

**LAPORAN FINAL PROJECT**  
**Pemantauan dan Manajemen Jaringan Skala Kecil menggunakan**  
***Software Defined Networking (SDN) Controller***  
Diajukan untuk memenuhi persyaratan kelulusan  
Program MSIB MBKM



**Oleh:**

Nama : Ryan Nurfadilah  
NPM : 197002059  
Asal Universitas : Universitas Siliwangi

**Program Studi Teknik Elektro Universitas Siliwangi**  
**2022**

## KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah Swt. yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **“Pemantauan dan Manajemen Jaringan Skala Kecil menggunakan Software Defined Networking (SDN) Controller”** ini tepat pada waktunya.

Adapun tujuan dari penulisan dari laporan ini adalah untuk memenuhi tugas pada program Studi Independen Kampus Merdeka. Selain itu, laporan ini juga bertujuan untuk menambah wawasan tentang implementasi programabilitas jaringan melalui *SDN Controller* untuk automasi jaringan komputer skala kecil sebagai bagian dari tema final projectnya yaitu Desain dan Implementasi jaringan skala kecil (*Small Network*) yang menerapkan *Network Programmability*.

Penulis mengucapkan terima kasih kepada Bapak Viddi Mardiansyah, S.Si., M.T. selaku instruktur pembimbing dalam kegiatan Studi Independen Kampus Merdeka. Selain itu, penulis juga berterima kasih kepada seluruh pihak yang membantu dalam proses penyusunan tugas akhir ini.

Karena keterbatasan pengetahuan maupun pengalaman, penulis menyadari masih banyak kekurangan dalam laporan akhir ini. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempuraan laporan ini. Semoga laporan ini dapat berguna bagi para pembaca

## DAFTAR ISI

halaman

### HALAMAN JUDUL

KATA PENGANTAR .....	i
----------------------	---

DAFTAR ISI.....	ii
-----------------	----

### BAB I PENDAHULUAN

A. Latar Belakang.....	1
B. Rumusan Masalah.....	1
C. Tujuan Penulisan .....	1

### BAB II PEMBAHASAN

A. Desain dan Konfigurasi Jaringan.....	2
1. Rencana Kebutuhan Jaringan.....	2
2. Perangkat Jaringan yang Digunakan.....	2
3. Pengalamatan .....	2
4. Konfigurasi Perangkat Jaringan .....	3
5. Tes Konektivitas Jaringan.....	13
B. Implementasi programabilitas jaringan melalui SDN Controller.....	14
1. Konfigurasi <i>Network controller</i> .....	14
2. Fitur Automasi yang diimplementasikan .....	16

### BAB III PENUTUP

A. Kesimpulan .....	37
B. Saran.....	37

LAMPIRAN .....	38
----------------	----

## **BAB I**

### **PENDAHULUAN**

#### A. Latar Belakang

Jumlah perangkat dalam jaringan terus meningkat. Metode manual yang digunakan untuk konfigurasi dan memantau peralatan jaringan memakan waktu, dengan mempertimbangkan juga pengetahuan khusus vendor yang dibutuhkan. Konsep *Software Defined Networking* (SDN) bisa menjadi jawaban untuk masalah tersebut.

Automasi jaringan adalah solusi untuk penghematan biaya operasional, meningkatkan tidak hanya waktu yang dihabiskan untuk mengkonfigurasi perangkat jaringan, tetapi juga efisiensi pemeliharaan jaringan melalui prosedur yang lebih mudah diikuti dan diterapkan dalam skala besar. Semua vendor besar, termasuk Cisco, mulai mempromosikan konfigurasi perangkat lunak jaringan (misalnya konsep Cisco DevNet yang mempromosikan penciptaan komunitas open source untuk programabilitas jaringan. Semua implementasi otomatisasi baru didasarkan pada metode pemrograman generik (python, java) dan antarmuka standar (*Secure Shell* SSH atau bahkan layanan web RESTful).

Tujuan utama dari laporan akhir ini adalah untuk mendemonstrasikan efisiensi *scripting* dalam automasi jaringan menggunakan REST-API pada *Network controller* yang ditulis dalam sintaks *Python* untuk memantau perangkat jaringan.

#### B. Rumusan Masalah

1. Bagaimana desain dan konfigurasi jaringan yang dibangun
2. Bagaimana implementasi programabilitas jaringan melalui SDN Controller pada jaringan yang dibangun?

#### C. Tujuan Penulisan

1. Membuat desain dan mengonfigurasi jaringan skala kecil (*small network*)
2. Mengimplementasikan programabilitas jaringan melalui fitur REST-API pada SDN Controller

## BAB II

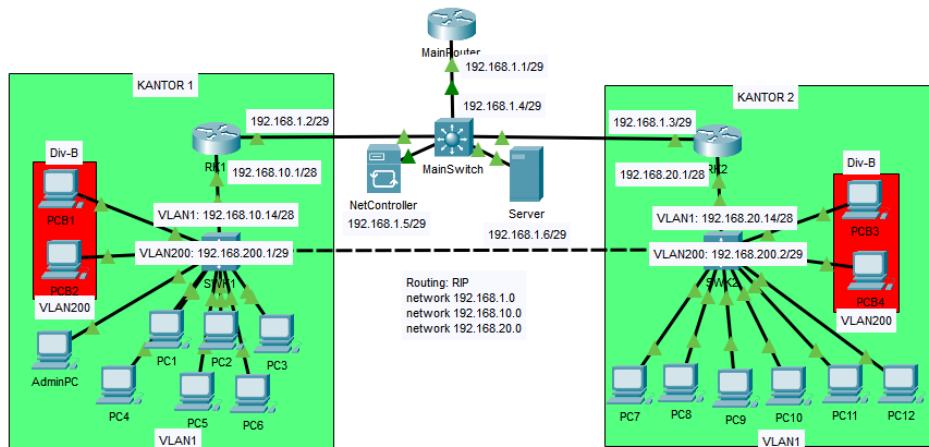
### PEMBAHASAN

#### A. Desain dan Konfigurasi Jaringan

##### 1. Rencana Kebutuhan Jaringan

Jaringan komputer yang dibangun berupa jaringan komputer skala kecil yang menghubungkan dua kantor, dengan pengguna di setiap kantor sebanyak 6-12 pengguna. Di setiap kantor terdapat *router* dan *switch*nya masing-masing, dan dua kantor tersebut dihubungkan dengan *router* utama melalui *switch* utama. Dua kantor tersebut juga membutuhkan jalur langsung antar kantor yang jaringannya privat dan tidak bisa diakses oleh pengguna di luar jaringan privat tersebut. Terdapat satu *server* dan *network controller* yang terhubung langsung ke jalur utama (*switch* utama).

Gambar berikut menunjukkan topologi desain jaringan yang dibangun.



##### 2. Perangkat Jaringan yang digunakan

Untuk memenuhi kebutuhan jaringan yang dibangun, digunakan perangkat jaringan sebagai berikut:

- 3 (tiga) unit *Router* Cisco 4331,
- 2 (dua) unit *Switch* Cisco 3650,
- 17 (tujuh belas) unit PC,
- 1 (satu) *Server*,
- 1 (satu) *Network controller*.

##### 3. Pengalaman

Tabel berikut memuat tentang konfigurasi pengalamatan untuk setiap elemen jaringan.

Device	Interface	IP Address
<i>MainRouter</i>	G0/0/0	192.168.1.1/29
RK1	G0/0/0	192.168.1.2/29
	G0/0/1	192.168.10.1/28
RK2	G0/0/0	192.168.1.3/29
	G0/0/1	192.168.20.1/28
<i>MainSwitch</i>	VLAN1	192.16.1.4/29
SWK1	VLAN1	192.168.10.14/28
	VLAN200	192.168.200.1/29
SWK2	VLAN1	192.168.20.14/28
	VLAN200	192.168.200.2/29
NetController	NIC	192.168.1.5/29
<i>Server</i>	NIC	192.168.1.6/29
AdminPC	NIC	192.168.10.13/28
PC1	NIC	DHCP
PC2	NIC	DHCP
PC3	NIC	DHCP
PC4	NIC	DHCP
PC5	NIC	DHCP
PC6	NIC	DHCP
PC7	NIC	DHCP
PC8	NIC	DHCP
PC9	NIC	DHCP
PC10	NIC	DHCP
PC11	NIC	DHCP
PC12	NIC	DHCP
PC-B1	NIC	192.168.200.3/29
PC-B2	NIC	192.168.200.4/29
PC-B3	NIC	192.168.200.5/29
PC-B4	NIC	192.168.200.6/29

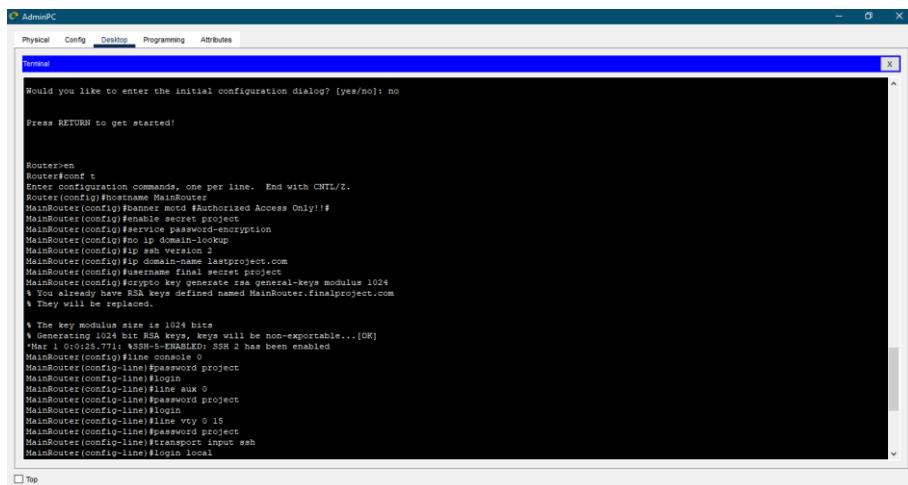
#### 4. Konfigurasi Perangkat Jaringan

##### a. Router Utama (*MainRouter*)

Pada *router* utama, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *router*, dan konfigurasi pengalamatan. Tabel berikut ini menunjukkan parameter konfigurasi dan keterangan tambahan yang dilakukan pada perangkat *Main Router*.

Parameter	Keterangan
<b>Konfigurasi Dasar</b>	
Hostname	<i>MainRouter</i>
Banner motd	Authorized Access Only!
Remote Access	SSH only (versi 2) Username: final Secret: project
User EXEC mode password	project (terenkripsi)
Privileged EXEC mode password	project (terenkripsi)
Akses langsung	Console, Auxiliary
<b>Pengalamatan</b>	
IP Address	Interface G0/0/0: 192.168.1.1/29
Routing ( <i>RIP</i> )	<ul style="list-style-type: none"> <li>Network: 192.168.1.0</li> <li>Network: 192.168.10.0</li> <li>Network: 192.168.20.0</li> </ul>

Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *router*. Gambar berikut ini menunjukan proses konfigurasi *MainRouter*



```

Admin@PC
Physical Config Desktop Programming Attributes
Would you like to enter the initial configuration dialog? [yes/no]: no
Press RETURN to get started!

Router>en
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname MainRouter
MainRouter(config)#banner motd #Authorized Access Only!#
MainRouter(config)#enable secret project
MainRouter(config)#enable password project
MainRouter(config)#no ip domain-lookup
MainRouter(config)#ip ssh version 2
MainRouter(config)#ip domain-name lastproject.com
MainRouter(config)#ip domain-list lastproject.com
MainRouter(config)#crypto key generate rsa general-keys modulus 1024
% You already have RSA keys defined named MainRouter.finalproject.com
% They will be replaced.
% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
% Mar 1 0:0:25.771: %SSH-5-ENABLED: SSH 2 has been enabled
MainRouter(config)#line vty 0 4
MainRouter(config-line)#password project
MainRouter(config-line)#login
MainRouter(config-line)#line aux 0
MainRouter(config-line)#password project
MainRouter(config-line)#login
MainRouter(config-line)#line vty 0 15
MainRouter(config-line)#password project
MainRouter(config-line)#transport input ssh
MainRouter(config-line)#login

```

```
AdminPC

Physical Config Desktop Programming Attributes

Terminal
% Ready. Will be replaced.

% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
# Mer 3 0:02:25,771; %SSH-5-ENABLED: SSH 3 has been enabled
MainRouter(config)#password secret
MainRouter(config-line)#password project
MainRouter(config-line)#login
MainRouter(config-line)#line aux 0
MainRouter(config-line)#line vty 0 15
MainRouter(config-line)#password project
MainRouter(config-line)#login
MainRouter(config-line)#line vty 0 15
MainRouter(config-line)#password project
MainRouter(config-line)#password project
MainRouter(config-line)#username admin
MainRouter(config-line)#password admin
MainRouter(config-line)#login local
MainRouter(config-line)#exit
MainRouter(config)#interface g0/0/0
MainRouter(config-if)#ip address 192.168.1.1 255.255.255.248
MainRouter(config-if)#description Link to MainSwitch
MainRouter(config-if)#no shutdown

MainRouter(config)#exit
MainRouter(config)#ip route 192.168.10.0 255.255.255.240 192.168.1.2
MainRouter(config)#ip route 192.168.20.0 255.255.255.240 192.168.1.3
MainRouter(config)#no shutdown

MainRouter#copy running-config startup-config
Destination filename [startup-config]?
%LINK-5-CHANGED: Interface GigabitEthernet0/0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to up

%SYS-5-CONFIG_I: Configured from console by console

Building configuration...
[OK]
MainRouter#
```

b. *Router Kantor 1 (RK1)*

Pada *router* kantor 1, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *router*, dan konfigurasi pengalamatan. Selain pengalamatan dasar, dikonfigurasikan juga DHCP untuk melayani *host* di LAN Kantor 1.

Tabel berikut ini menunjukkan parameter konfigurasi yang dilakukan pada perangkat RK1.

Parameter	Keterangan
<b>Konfigurasi Dasar</b>	
Hostname	RK1
Banner motd	Authorized Access Only!
Remote Access	SSH only (versi 2) Username: final Password: project
User EXEC mode password	project (terenkripsi)
Privileged EXEC mode password	project (terenkripsi)
Akses langsung	Console, Auxiliary
<b>Pengalamatan</b>	
IP Address	<ul style="list-style-type: none"> <li>• Interface G0/0/0: 192.168.1.2/29</li> <li>• Interface G0/0/1: 192.168.10.1/28</li> </ul>

<i>Routing (RIP)</i>	<ul style="list-style-type: none"> <li>• <i>Network:</i> 192.168.1.0</li> <li>• <i>Network:</i> 192.168.10.0</li> <li>• <i>Network:</i> 192.168.20.0</li> </ul>
DHCP	Pool: Kantor1 <i>Default-router:</i> 192.168.10.1 <i>Network:</i> 192.168.10.0/29

Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *router*. Gambar dibawah ini menunjukan proses konfigurasi RK1.

AdminPC

Physical Config Desktop Programming Attributes

Terminal

```
--- System Configuration Dialog ---  
Would you like to enter the initial configuration dialog? [yes/no]: no  
  
Press RETURN to get started!  
  
Router#  
Router>conf t  
Enter configuration commands, one per line. End with CNTL/Z.  
Router(config)#hostname R1  
R1(config)#username admin secret R1-Authorized Access Only!!#  
R1(config)#enable secret project  
R1(config)#service password-encryption  
R1(config)#ip domain-lookup  
R1(config)#ip domain-name R1Project.com  
R1(config)#username final secret project  
R1(config)#crypto key generate rsa general-keys modulus 1024  
! You already have RSA keys defined named MainRouter.lastproject.com  
They will be replaced.  
  
! The key modulus size is 1024 bits  
! The key is stored in non-volatile memory. It will be non-exportable...[OK]  
*Mar 1 0:00:32.121: %SSH-5-ENABLED: SSH 2 has been enabled  
R1(config)#line console 0  
R1(config-line)#password project  
R1(config-line)#exit  
R1(config)#line aux 0  
R1(config-line)#password project  
R1(config-line)#login  
R1(config-line)#line vty 0 15  
R1(config-line)#password project  
R1(config-line)#transport input ssh  
  
Top
```

AdminPC

Physical Config Desktop Programming Attributes

Terminal

```
R1(config-line)#transport input ssh  
R1(config-line)#login local  
R1(config-line)#exit  
R1(config)#interface g0/0/0  
R1(config-if)#ip address 192.168.1.2 255.255.255.248  
R1(config-if)#description Link to MainSwitch  
R1(config-if)#no shutdown  
  
R1(config-if)#exit  
R1(config)#interface g0/0/1  
R1(config-if)#ip address 192.168.10.1 255.255.255.240  
R1(config-if)#description Link to R1-LAN  
R1(config-if)#no shutdown  
  
R1(config)#exit  
R1(config)#ip pool Kantori  
R1(config)#default-router 192.168.10.1  
R1(dhcp-config)#network 192.168.10.0 255.255.255.240  
R1(dhcp-config)#exit  
R1(config)#ip route 0.0.0.0 0.0.0.0 192.168.20.0 255.255.255.250  
R1(config)#ip route 192.168.1.0 255.255.255.255 192.168.1.1  
R1(config)#end  
R1(config)#copy running-config startup-config  
Destination filename? [status= config]?  
!LINEPROTO-5-CHANGED: Interface GigabitEthernet0/0/0, changed state to up  
!LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to up  
!LINK-5-CHANGED: Interface GigabitEthernet0/0/1, changed state to up  
  
%SYS-5-CONFIG_I: Configured from console by console  
%IPV4-DUPADDR: Duplicate address 192.168.1.2 on GigabitEthernet0/0/0, sourced by 0006.5C49.6B4E  
  
Building configuration...  
[OK]  
R1#
```

c. Router Kantor 2 (RK2)

Pada *router* kantor 2, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *router*, dan konfigurasi pengalamanan. Selain pengalamanan dasar, dikonfigurasikan juga DHCP untuk melayani *host* di LAN Kantor 2.

Tabel berikut ini menunjukkan parameter konfigurasi yang dilakukan pada perangkat RK2.

Parameter	Keterangan
<b>Konfgurasi Dasar</b>	
Hostname	RK2
Banner motd	Authorized Access Only!
Remote Access	SSH only (versi 2) Username: final Secret: project
User EXEC mode password	project (terenkripsi)
Privileged EXEC mode password	project (terenkripsi)
Akses langsung	Console, Auxiliary
<b>Pengalamatan</b>	
IP Address	<ul style="list-style-type: none"> <li>Interface G0/0/0: 192.168.1.3/29</li> <li>Interface G0/0/1: 192.168.20.1/28</li> </ul>
Routing ( <i>RIP</i> )	<ul style="list-style-type: none"> <li>Network: 192.168.1.0</li> <li>Network: 192.168.10.0</li> <li>Network: 192.168.20.0</li> </ul>
DHCP	Pool: Kantor1 Default-router: 192.168.20.1 Network: 192.168.20.0/29

Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *router*. Gambar dibawah ini menunjukan proses konfigurasi RK2.

```

Administrator: ~
Physical Config Desktop Programming Attributes
System Configuration Dialog
Would you like to enter the initial configuration dialog? [yes/no]: no
Press RETURN to get started!

Router#en
Router>conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname RK2
RK2(config)#banner motd "Authorized Access Only!!"
RK2(config)#username final secret project
RK2(config)#service password-encryption
RK2(config)#ip domain-lookup
RK2(config)#ip ssh version 2
RK2(config)#ip ssh server lastproject.com
RK2(config)#username final secret project
RK2(config)#crypto key generate rsa general-keys modulus 1024
% You already have RSA keys defined named RK1.lastproject.com
% They will be replaced.

% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
% RSA key generation completed. RSA is ENABLED: SSH 2 has been enabled
RK2(config)#line console 0
RK2(config-line)#password project
RK2(config-line)#login
RK2(config-line)#line aux 0
RK2(config-line)#password project
RK2(config-line)#login
RK2(config-line)#line vty 0 15
RK2(config-line)#password project
RK2(config-line)#transport input ssh

```

```

Administrator: ~
Physical Config Desktop Programming Attributes
System Configuration Dialog
RK2(config)#line password project
RK2(config-line)#transport input ssh
RK2(config-line)#login
RK2(config-line)#line aux 0
RK2(config-line)#password project
RK2(config-line)#login
RK2(config-line)#line vty 0 15
RK2(config-line)#password project
RK2(config-line)#no shutdown
RK2(config-line)#exit
RK2(config)#interface g0/0/0
RK2(config-if)#ip address 192.168.1.3 255.255.255.248
RK2(config-if)#description Link to MainSwitch
RK2(config-if)#no shutdown
RK2(config-if)#exit
RK2(config)#interface g0/0/1
RK2(config-if)#ip address 192.168.20.1 255.255.255.240
RK2(config-if)#description Link to RK2-LAN
RK2(config-if)#no shutdown
RK2(config-if)#exit
RK2(config)#ip dhcp pool Kantor2
RK2(dhcp-config)#default-router 192.168.20.1
RK2(dhcp-config)#network 192.168.20.0 255.255.255.240
RK2(dhcp-config)#exit
RK2(config)#ip route 192.168.10.0 255.255.255.240 192.168.1.2
RK2(config)#ip route 192.168.1.0 255.255.255.248 192.168.1.2
RK2(config)#end
RK2(config)#copy running-config startup-config
Destination filename [startup-config]?
!LINK-5-CHANGED: Interface GigabitEthernet0/0/0, changed state to up
!LINKPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to up
!LINK-5-CHANGED: Interface GigabitEthernet0/0/1, changed state to up
!SYS-5-CONFIFO_1: Configured from console by console
Building configuration...
[OK]
RK2#

```

#### d. Switch Utama (*Main Switch*)

Pada *switch* utama, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *switch*, dan konfigurasi pengalamanan untuk remote akses SSH. Tabel berikut ini menunjukkan parameter konfigurasi yang dilakukan pada perangkat *MainSwitch*.

Parameter	Keterangan
<b>Konfigurasi Dasar</b>	
Hostname	<i>MainSwitch</i>
Banner motd	Authorized Access Only!
Remote Access	SSH only (versi 2) Username: final Secret: project
User EXEC mode password	project (terenkripsi)

Privileged EXEC mode password	project (terenkripsi)
Akses langsung	Console
<b>Pengalamatan</b>	
IP Address	Interface VLAN1: 192.168.1.4/29 Default-gateway: 192.168.1.1
<b>VLAN Database</b>	
Interface	G1/0/1-24, G1/1/1-4: VLAN1

Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *switch*. Gambar dibawah ini menunjukan proses konfigurasi *MainSwitch*.

```

--- System Configuration Dialog ---
Would you like to enter the initial configuration dialog? [yes/no]: n
Press RETURN to get started!

Switch#>
Switch#>conf t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#hostname MainSwitch
MainSwitch#>enable secret project
MainSwitch(config)#enable secret project
MainSwitch(config)#service password-encryption
MainSwitch(config)#no ip domain-lookup
MainSwitch(config)#ip domain-name lastproject.com
Please create RSA keys (of at least 768 bits size) to enable SSH v2.
MainSwitch(config)#ip domain-name lastproject.com
MainSwitch(config)#username final secret project
MainSwitch(config)#crypto key generate rsa general-keys modulus 1024
The name for the keys will be: MainSwitch.lastproject.com

The key modulus size is 1024 bits
RSA keys will be non-exportable...[OK]
* Mar 1 01:14:47.71: %SSH-3-ENABLED: SSH 2 has been enabled
MainSwitch(config)#line console 0
MainSwitch(config-line)#password project
MainSwitch(config-line)#login
MainSwitch(config-line)#line vty 0 15
MainSwitch(config-line)#password project
MainSwitch(config-line)#transport input ssh
MainSwitch(config-line)#login local
MainSwitch(config-line)#exit
MainSwitch(config)#int vlan 1
MainSwitch(config-if)#no shutdown

MainSwitch(config-if)#ip address 192.168.1.4 255.255.255.240
MainSwitch(config-if)#exit
MainSwitch(config)#ip default-gateway 192.168.1.1
MainSwitch(config)#end
MainSwitch#>copy running-config startup-config
Destination filename [startup-config]?
*LINK-5-CHANGED: Interface Vlan1, changed state to up
SYS-5-CONFIG_I: Configured from console by console
Building configuration...
[OK]
MainSwitch#

```

#### e. Switch Kantor 1 (SWK1)

Pada *switch* kantor 1, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *switch*, dan konfigurasi pengalamatan untuk

remote akses SSH. Dikonfigurasi juga VLAN untuk akses privat antar kantor. Tabel berikut ini menunjukkan parameter konfigurasi yang dilakukan pada perangkat SWK1.

Parameter	Keterangan
<b>Konfgurasi Dasar</b>	
Hostname	SRK1
Banner motd	Authorized Access Only!
Remote Access	SSH only (versi 2) Username: final Secret: project
User EXEC mode password	project (terenkripsi)
Privileged EXEC mode password	project (terenkripsi)
Akses langsung	Console
<b>Pengalamatan</b>	
IP Address	<ul style="list-style-type: none"> <li>• Interface VLAN1: 192.168.10.14/29</li> <li>• Interface VLAN200: 192.168.200.1/29</li> </ul> Default-gateway: 192.168.10.1
<b>VLAN Database</b>	
Interface (nama)	<ul style="list-style-type: none"> <li>• VLAN1 (default) G1/0/1-19, G1/1/1-4</li> <li>• VLAN200 (Div-B) G1/0/20-24 Mode Access</li> </ul>

Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *switch*. Gambar dibawah ini menunjukkan proses konfigurasi SWK1.

```

AdminPC
Physical Config Desktop Programming Attributes
Terminal
--- System Configuration Dialog ---
Would you like to enter the initial configuration dialog? [yes/no]: n
Press RETURN to get started!

Switch# 
Switch#config t
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)#hostname SWK1
SWK1(config)#banner motd #Authorized Access Only!#
SWK1(config)#service ssh
SWK1(config)#service password-encryption
SWK1(config)#no ip domain-lookup
SWK1(config)#ip ssh version 2
SWK1(config)#crypto key generate rsa lastproject.com
SWK1(config)#username final secret project
SWK1(config)#crypto key generate rsa general-keys modulus 1024
% You already have RSA keys defined named SWK1.lastproject.com
% They will be replaced.

% The key modulus size is 1024 bits
Generating 1024 bit RSA keys, keys will be non-exportable...[OK]
%SSH-2-ENABLED: SSH 2 has been enabled
SWK1(config)#line console 0
SWK1(config-line)#password project
SWK1(config-line)#login
SWK1(config-line)#line vty 0 15
SWK1(config-line)#password project
SWK1(config-line)#transport input ssh
SWK1(config-line)#login local
SWK1(config-line)#exit
SWK1(config)#line vty 1
SWK1(config-line)#exit

Mar 1 0:0:18.328: %SSH-5-ENABLED: SSH 2 has been enabled
SWK1(config)#line console 0
SWK1(config-line)#password project
SWK1(config-line)#login
SWK1(config-line)#line vty 0 15
SWK1(config-line)#password project
SWK1(config-line)#transport input ssh
SWK1(config-line)#login local
SWK1(config-line)#exit
SWK1(config)#line vlan 1
SWK1(config-line)#shutdown
SWK1(config-if)#ip address 192.168.10.14 255.255.255.240
SWK1(config-if)#exit
SWK1(config)#ip default-gateway 192.168.10.1
SWK1(config)#int vlan 200
SWK1(config-vlan)#name Div-B
SWK1(config-vlan)#int vlan 200
SWK1(config-if)#ip address 192.168.200.1 255.255.255.240
SWK1(config-if)#switch mode access
SWK1(config-if-range)#switch access vlan 200
SWK1(config-if-range)#exit
SWK1(config)#end
SWK1#configure config startup-config
Destination filename [startup-config]?
%LINK-5-CHANGED: Interface Vlan1, changed state to up
%SYS-5-CONFIG_I: Configured from console by console
Building configuration...
[OK]
SWK1#

```

#### f. Switch Kantor 2 (SWK2)

Pada *switch* kantor 2, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *switch*, dan konfigurasi pengalamanan untuk remote akses SSH. Dikonfigurasi juga *vlan* untuk akses privat antar kantor. Tabel berikut ini menunjukan parameter konfigurasi yang dilakukan pada perangkat SWK2.

Parameter	Keterangan
<b>Konfgurasi Dasar</b>	
Hostname	SRK2
Banner motd	Authorized Access Only!
Remote Access	SSH only (versi 2) Username: final Secret: project

User EXEC mode password	project (terenkripsi)
Privileged EXEC mode password	project (terenkripsi)
Akses langsung	Console
<b>Pengalamatan</b>	
IP Address	<ul style="list-style-type: none"> <li>Interface VLAN1: 192.168.20.14/29</li> <li>Interface VLAN200: 192.168.200.1/29</li> </ul> Default-gateway: 192.168.20.1
<b>VLAN Database</b>	
Interface (nama)	<ul style="list-style-type: none"> <li>VLAN1 (default) G1/0/1-19, G1/1/1-4</li> <li>VLAN200 (Div-B) G1/0/20-24</li> </ul> Mode Access

Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *switch*. Gambar dibawah ini menunjukan proses konfigurasi SWK2.

```
Administrator: ~
```

Physical Config Desktop Programming Attributes

Terminal X

```
--- System Configuration Dialog ---
```

Would you like to enter the initial configuration dialog? (yes/no): n

Press RETURN to get started!

Switch>en

Switch#conf t

Enter configuration commands, one per line. End with CNTL/Z.

Switch(config)#hostname SWK2

SWK2(config)#banner motd #Authorized Access Only!#

SWK2(config)#enable secret project

SWK2(config)#enable password project

SWK2(config)#no ip domain-lookup

SWK2(config)#ip ssh version 2

SWK2(config)#ip domain-name lastproject.com

SWK2(config)#crypto ipsec general secret project

SWK2(config)#crypto key generate rsa general-keys modulus 1024

% You already have RSA keys defined named SWK1.lastproject.com

% They will be replaced.

% The key modulus size is 1024 bits

% Generating 1024 bit RSA keys, keys will be non-exportable...(OK)

HWIC-0/0/1#0/1/1<0>%00000000000000000000000000000000: SSH 2 has been enabled

SWK2(config-line)#password console 0

SWK2(config-line)#password project

SWK2(config-line)#login

SWK2(config-line)#line vty 0 15

SWK2(config-line)#password project

SWK2(config-line)#transport input ssh

SWK2(config-line)#login local

SWK2(config-line)#exit

SWK2(config-line)1

SWK2(config-1)#no shutdown

```

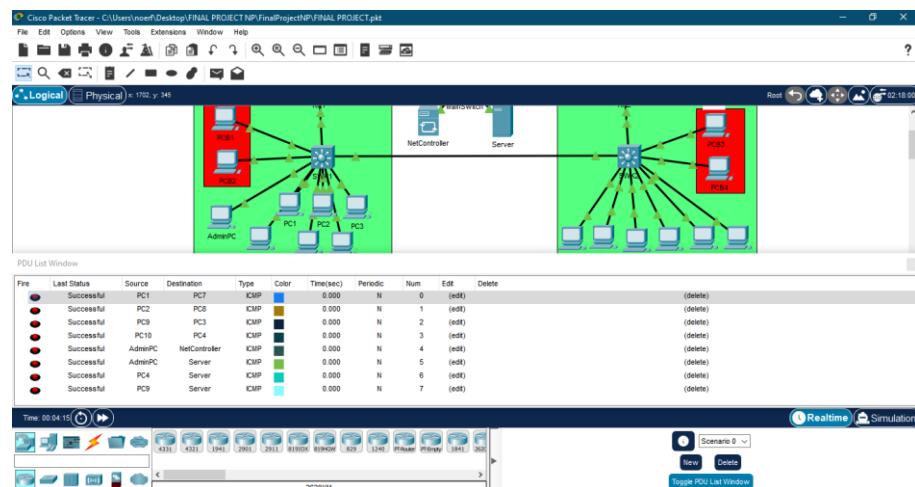
AdminPC
Physical Config Desktop Programming Attributes
Terminal
SWK1# configure terminal
SWK1# line console 0
SWK1(config-line)#password project
SWK1(config-line)#login
SWK1(config-line)#secret 0 test
SWK1(config-line)#password project
SWK1(config-line)#transport input ssh
SWK1(config-line)#login local
SWK1(config-line)#exit
SWK1(config-if)#line vlan 1
SWK1(config-if)#no shutdown

SWK1(config-if)#ip address 192.168.20.14 255.255.255.14
Bad mask 0xFFFFFFF for address 192.168.20.14
SWK1(config-if)#exit
SWK1(config)#vlan 200
SWK1(config-vlan)#name Div-B
SWK1(config-vlan)#exit
SWK1(config-if)#ip address 192.168.200.2 255.255.255.248
SWK1(config-if)#ip range 192.168.200.2 255.255.255.248
SWK1(config-if-range)#switch mode access
SWK1(config-if-range)#switch access vlan 200
SWK1(config-if-range)#exit
SWK1(config)#end
Building configuration...
[OK]
SWK1#

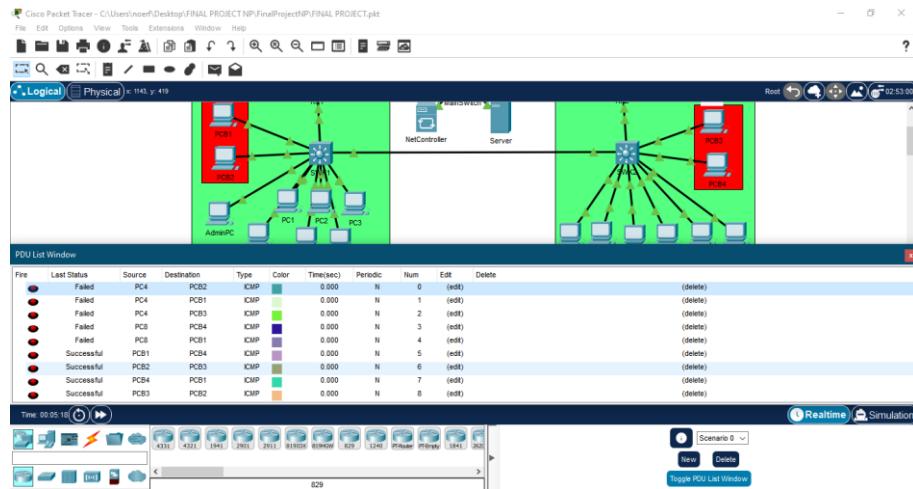
```

## 5. Tes Konektivitas Jaringan

Tes konektivitas dilakukan untuk memvalidasi koneksi pada jaringan antar kantor, gambar di bawah ini menunjukkan kedua kantor bisa berkomunikasi satu sama lain (diluar jaringan privat).



Dan untuk gambar di bawah ini menunjukkan, konektivitas pada jaringan privat yang dibangun oleh SWK1 dan SWK2 yang tidak bisa diakses oleh *host* di luar jaringan itu, begitupun *host* yang ada di jaringan tersebut tidak bisa mengakses jaringan luar. Namun untuk *host* yang berada pada jaringan yang sama, bisa saling berkomunikasi



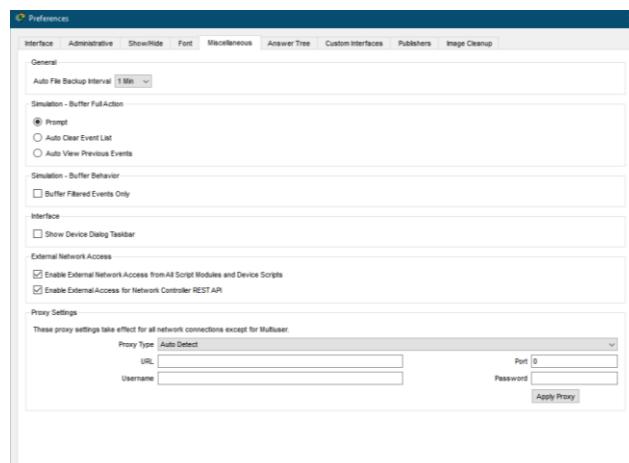
## B. Implementasi Programabilitas Jaringan melalui SDN Controller

Pada tugas akhir ini, jaringan dirancang menggunakan Network Simulation Tool yaitu Cisco Packet Tracer 8.1.1. Proses implementasi programabilitas jaringan berupa automasi diimplementasikan dengan 2 cara, yaitu melalui programming AdminPC pada simulatornya, dan melalui aplikasi pihak ketiga, yaitu Visual Studio Code. File-file program disimpan dalam repositori lokal yang dikelola oleh aplikasi Git, juga tersedia repositori daringnya pada laman Github (<https://github.com/noerfadil26/FinalProjectNP>).

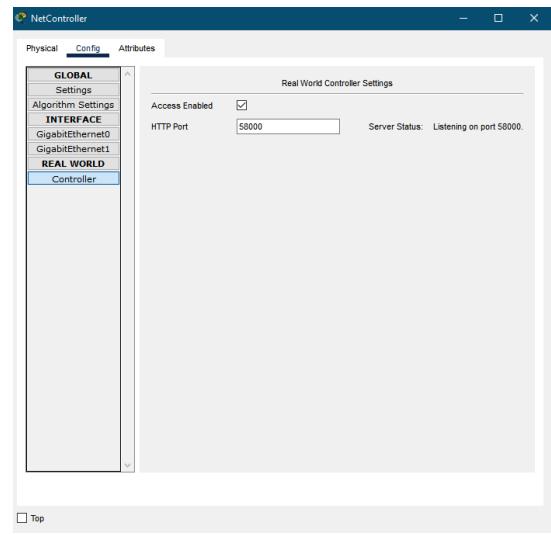
Adapun untuk cakupan automasi yang dilakukan dalam tugas akhir ini meliputi monitoring dan manajemen perangkat jaringan melalui *network controller* yang terhubung ke jaringan.

### 1. Konfigurasi *Network Controller*

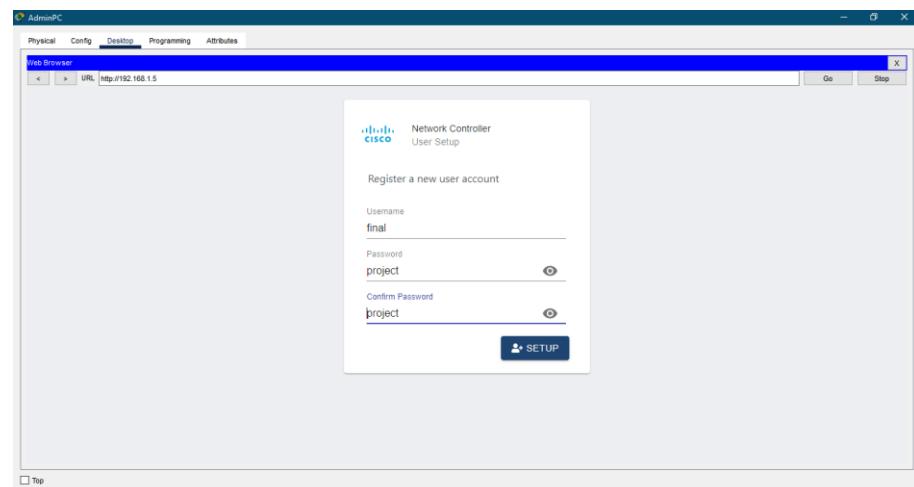
Untuk mengakses *network controller*, centang opsi pada External Network Access dari menu Options > Preferences > Miscellaneous.



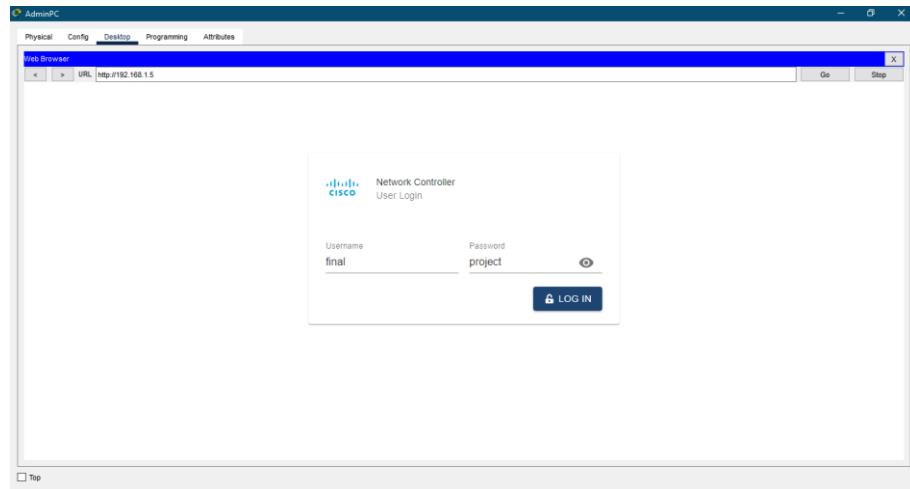
Dari perangkat *network controller*, aktifkan Access Enabled pada Config > REAL WORLD > Controller.



Dari AdminPC, akses *network controller* melalui web browser dengan mengunjungi alamat IP *network controller*, lalu buat akun baru.



Setelah membuat akun baru, login ke *network controller* dengan akun yang sudah dibuat.



Setelah login, akan muncul dashboard dari *network controller* dan fitur-fiturnya, *Network controller* ini bisa dioperasikan melalui laman web GUI, atau melalui scripting menggunakan REST-API-nya. Dengan REST-API yang ada pada *network controller*, memungkinkan administrator untuk memonitor dan megoperasikan jaringan melalui *network controller* secara scripting.

Fungsi automasi jaringan *network controller* diakses melalui fitur REST-API yang dijadikan program yang ditulis menggunakan bahasa Python. Aturan penulisan program dan parameter-parameternya mengikuti API documentation (lihat lampiran) yang ada pada *Network controller*.

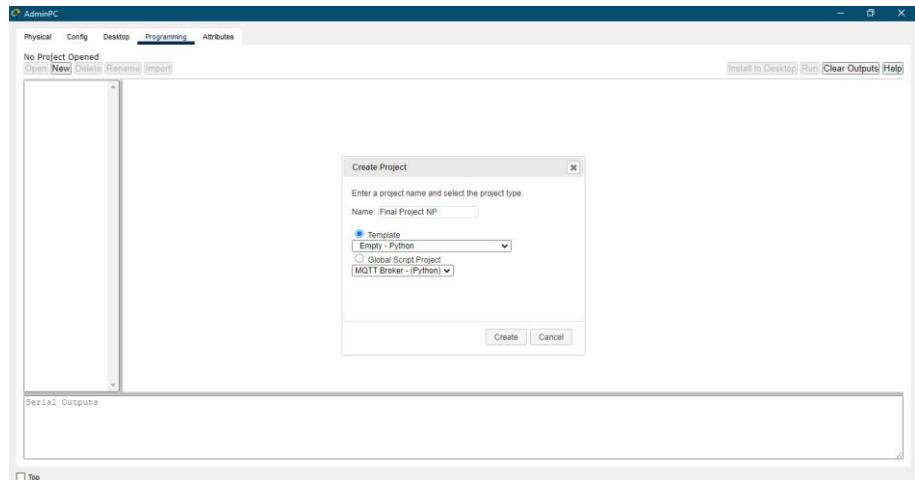
## 2. Fitur Automasi yang diimplementasikan

Pada tugas akhir ini, sudah dibuat beberapa program untuk implementasi network programmability pada jaringan. Adapun program-program automasi yang dilakukan pada tugas akhir ini meliputi program-program berikut:

### a. Mendapatkan Service Ticket (Get Ticket)

Untuk mengakses *Network controller* melalui scripting, dibutuhkan API-Ticket dari *Network controller*nya.

Pada AdminPC, di bagian programming, buat program untuk mengakses dan meminta service ticket API *Network controller*.



Pada programnya, isi `api_url` sesuai API documentation (lihat lampiran), dan pada `body_json` isi parameter `username` dan `password` *network controller* yang sudah dibuat sebelumnya, lalu jalankan program.

```
1 #Get Ticket
2 import json
3 import requests
4 api_url = "192.168.1.8/api/v1/ticket"
5 headers = {
6     "Content-Type": "application/json"
7 }
8 body_json = {
9     "username": "final",
10    "password": "project"
11 }
12 resp = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)
13 print("Status code: ", resp.status_code)
14 response_json = resp.json()
15 serviceTicket = response_json["response"]["serviceTicket"]
16 print("The service ticket number is: ", serviceTicket)
```

Setelah program dijalankan, akan muncul output service ticket APInya, service ticket tersebutlah yang akan digunakan untuk akses ke API *network controller*.

```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
Install to Desktop Run Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: 100%
main.py
1 #Get Ticket
2 import json
3 import requests
4 api_url = "http://192.168.1.5/api/v1/ticket"
5 headers = {
6     "Content-Type": "application/json"
7 }
8 body_json = {
9     "username": "final",
10    "password": "project"
11 }
12 resp = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)
13 print("Ticket request status: ", resp.status_code)
14 response_json = resp.json()
15 serviceTicket = response_json["response"]["serviceTicket"]
16 print("The service ticket number is: ", serviceTicket)
17
Starting Final Project NP (Python)...
('Ticket request status: ', 201)
('The service ticket number is: ', 'NC-8-380f38c0led343e58ef4-nbl')
Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

File Edit Selection View Go Run Terminal Help Get Ticket.py - PTAutomation - Visual Studio Code
Update Network Wide Setting.py M Get Network Wide Setting.py M Get Credential.py M Get Ticket.py X Delete Ticket.py M Delete All Discovery.py M D ...
Get Ticket.py
1 #Get Ticket
2 import json
3 import requests
4 api_url = "http://localhost:58000/api/v1/ticket"
5 headers = {
6     "Content-Type": "application/json"
7 }
8 body_json = {
9     "username": "final",
10    "password": "project"
11 }
12 resp = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)
13 print("Ticket request status: ", resp.status_code)
14 response_json = resp.json()
15 serviceTicket = response_json["response"]["serviceTicket"]
16 print("The service ticket number is: ", serviceTicket)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
PS C:\Users\noorf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation> python Get Ticket.py
Ticket request status: 201
The service ticket number is: NC-8-380f38c0led343e58ef4-nbl
PS C:\Users\noorf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation>

```

### b. Menambahkan Kredensial (Add Credential)

Pada *network controller*, dibutuhkan sebuah kredensial untuk bisa mengakses perangkat jaringan, pada hal ini kredensial tersebut adalah username dan password remote akses SSH dari setiap perangkat *Router* dan *Switch*.

Pada programnya, isi `api_url` sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Pada bagian `body_json` parameter yang harus diisi adalah username, password, enablePassword sesuai dengan akun SSH pada perangkat jaringan, dan isikan nama kredensialnya pada bagian description.

```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
main.py
1 #Add Credential
2 import requests
3 import json
4
5 api_url = "192.168.1.5/api/v1/global-credential/cli"
6 headers = {
7     'X-Auth-Token': 'NC-9-lae09c4ef2aa43d0bcfd-nbi',
8     'Content-Type': 'application/json'
9 }
10 body_json = [
11     {
12         "password": "project",
13         "username": "final",
14         "enablePassword": "project",
15         "description": "OFFICE"
16     }
17 ]
18 response = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)
19 print(response.text)

```

Serial Outputs

Lalu jalankan program. Setelah program dijalankan, output akan menampilkan kredensial yang sudah dibuat.

```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
main.py
1 #Add Credential
2 import requests
3 import json
4
5 api_url = "192.168.1.5/api/v1/global-credential/cli"
6 headers = {
7     'X-Auth-Token': 'NC-9-lae09c4ef2aa43d0bcfd-nbi',
8     'Content-Type': 'application/json'
9 }
10 body_json = [
11     {
12         "password": "project",
13         "username": "final",
14         "enablePassword": "project",
15         "description": "OFFICE"
16     }
17 ]
18 response = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)
19 print(response.text)

Starting Final Project NP (Python)...
{
    "description": "OFFICE",
    "enablePassword": "project",
    "id": "ccdfb793c-a89c-4484-a531-2aab082aa4d4",
    "instanceUuid": "ccdfb793c-a89c-4484-a531-2aab082aa4d4",
    "password": "project",
    "username": "final"
}
Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

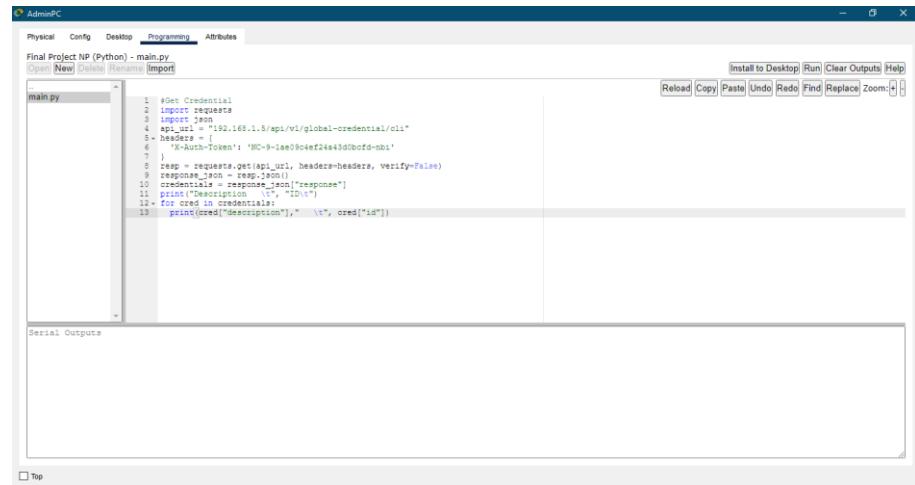
```

File Edit Selection View Go Run Terminal Help • Get Credential.py - PTAutomation - Visual Studio Code
Get Credential.py M Get Ticket.py M Delete 1 ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
PS C:\Users\noerf\Desktop\FINAL PROJECT NP\PTAutomation> & C:/Users/noerf/AppData/Local/Microsoft/WindowsApps/python3.9.exe "C:/Users/noerf/Desktop/FINAL PROJECT NP/PTAutomation/add_Credential.py"
(
    "description": "OFFICE",
    "enablePassword": "project",
    "id": "79a3abdc-f89f-4caa-b558-b66eech07231",
    "instanceUuid": "79a3abdc-f89f-4caa-b558-b66eech07231",
    "password": "project",
    "username": "final"
)

```

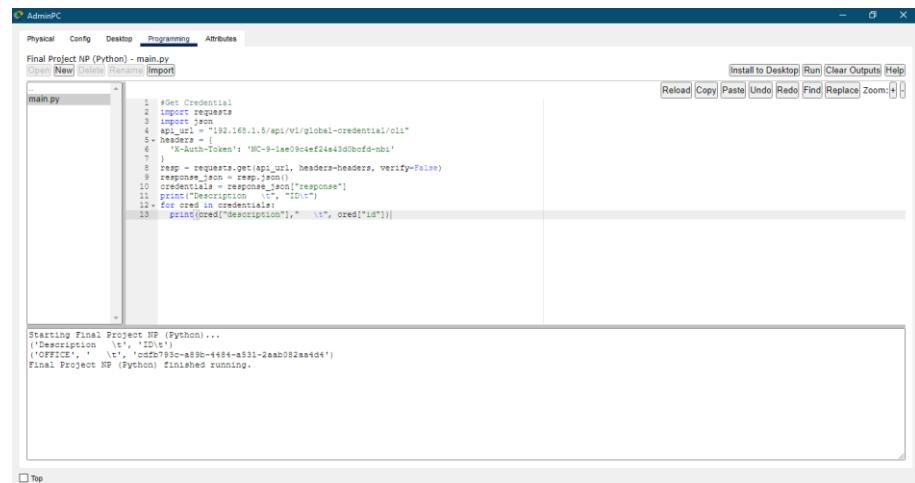
### c. Mendapatkan Kredensial (Get Credential)

Apabila ingin mengetahui apa saja kredensial yang ada di *network controller*, bisa digunakan program get credential seperti pada gambar dibawah ini. Pada programnya, isi api\_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket.



```
#Get Credential
import requests
import json
api_url = "192.168.1.5/api/v1/global-credential/cli"
headers = {"X-Auth-Token": "NC9-9-lae90e4ef24a43d0bcfd9-nhi"}
response = requests.get(api_url, headers=headers, verify=False)
response_json = response.json()
credentials = response_json["response"]
for cred in credentials:
    print("Description\t", "ID\t")
    for cred in credentials:
        print(cred["description"], "\t", cred["id"])
```

Setelah program dijalankan, output akan menampilkan kredensial yang ada pada *network controller*.



```
Starting Final Project NP (Python)...
("Description\t", "ID")
("OFFICE", "cd2b793c-a5b8-4484-a531-2a8b082aa4d4")
Final Project NP (Python) finished running.
```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

File Edit Selection View Go Run Terminal Help
Get Credential.py - PTAutomation - Visual Studio Code
Network Device.py Get Host.py M Update Network Wide Setting.py M Get Network Wide Setting.py M Get Credential.py M Get Ticket.py Delete 1 ...
Get Credential.py > ...
1 import requests
2 import json
3 api_url = "http://localhost:58000/api/v1/global-credential/cli"
4 headers = {
5     'X-Auth-Token': 'NC-9-3ae0f3c01e1d343ebf4-nbl'
6 }
7 resp = requests.get(api_url, headers=headers, verify=False)
8 response_json = resp.json()
9 credentials = response_json["response"]
10 print("Description\tID\t")
11 for cred in credentials:
12     | print(f'{cred["description"]}\t{cred["id"]}')

```

TERMINAL

```

PS C:\Users\noerf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation> & C:/Users/noerf/AppData/Local/Microsoft/WindowsApps/python3.9.exe "C:/Users/noerf/Desktop/FINAL PROJECT NP/FinalProjectNP\PTAutomation\Get_Credential.py"
Description ID
OFFICE 7983a9dc-f90f-4ca9-b958-bbfeecb97231
PS C:\Users\noerf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation>

```

#### d. Menambahkan Discovery (Add Discovery)

Agar bisa melakukan pemantauan, harus ditambahkan discovery pada *network controller*. Discovery ini yang akan melakukan scanning pada jaringan yang terhubung dengan *network controller*.

Pada programnya, isi `api_url` sesuai dengan API documentation (lihat lampiran), dan isi `X-Auth-Token` nya dengan service ticket. Pada bagian `body_json` ada beberapa parameter yang harus diisi, yaitu `cdpLevel`, `globalCredentialIdlist`, `timeout`, `retry`, `ipAddressList`, `discoveryType`, dan nama discoverynya. Isi setiap parameter tersebut lalu jalankan program.

```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
Install to Desktop Run Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: 100%
main.py
1 #Add Discovery
2 import requests
3 import json
4 api_url = "192.168.1.8/api/v1/discovery"
5 headers = {
6     'X-Auth-Token': 'NC-9-3ae0f3c01e1d343ebf4-nbl',
7     'Content-Type': 'application/json'
8 }
9 body_json = [
10     {
11         "cdpLevel": "16",
12         "globalCredentialIdlist": [
13             "cdfb793c-a39b-4444-a551-2aab022eadd4"
14         ],
15         "timeout": "3",
16         "ipAddressList": "192.168.1.1",
17         "discoveryType": "CDP",
18         "name": "OFFICE"
19     }
20 ]
21 response = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)
22 print(response.text)

```

Serial Outputs

Apabila sudah dijalankan, output program akan menampilkan id dari discovery yang sudah dibuat.

```

1 #Add Discovery
2 import requests
3 import json
4 api_url = "http://192.168.1.8/api/v1/discovery"
5 headers = {
6     'X-Auth-Token': 'NC-9-1ae09c4ef2443d0bfcd-nbi',
7     'Content-type': 'application/json'
8 }
9 body_json = {
10     "cdpLevel": "16",
11     "retry": "3",
12     "globalCredentialIdList": [
13         "cdtb73c9e-44f4-8531-2aab02aa4d4"
14     ],
15     "ipAddressList": "192.168.1.1",
16     "discoverType": "CDP",
17     "name": "OFFICE"
18 }
19 response = requests.post(api_url,json.dumps(body_json), headers=headers, verify=False)
20 print(response.text)
21
22

```

Starting Final Project NP (Python)...

```

{
    "discoveryId": "",
    "version": "1.0"
}

Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

File Edit Selection View Go Run Terminal Help Add Discovery.py - PTAutomation - Visual Studio Code
File Edit Selection View Go Run Terminal Help Add Discovery.py - PTAutomation - Visual Studio Code
1 import requests
2 import json
3 api_url = "http://localhost:58000/api/v1/discovery"
4 headers = {
5     'X-Auth-Token': 'NC-B-3B0F8C01e3d3e5b0f4-nbi',
6     'Content-type': 'application/json'
7 }
8 body_json = {
9     "cdpLevel": "16",
10    "retry": "3",
11    "globalCredentialIdList": [
12        "79a3a0dc-f89f-4caa-b958-bb6eecc0e7231"
13    ],
14    "timeout": "5",
15    "ipAddressList": "192.168.1.1",
16    "discoverType": "CDP",
17    "name": "OFFICE"
18 }
19 response = requests.post(api_url,json.dumps(body_json), headers=headers, verify=False)
20 print(response.text)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
PS C:\Users\noerf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation> & C:\Users\noerf\AppData\Local\Microsoft\WindowsApps\python3.9.exe "c:/Users/noerf/Desktop/FINAL_PROJECT_NP/FinalProjectNP\main.py"
{
    "discoveryId": "",
    "version": "1.0"
}

```

#### e. Mendapatkan Discovery (Get Discovery)

Untuk mengetahui discovery pada *network controller*, bisa digunakan program berikut. Pada programnya, isi `api_url` sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.

```

#Get Discovery
import requests
api_url = "http://192.168.1.8/api/v1/discovery"
headers = {
    'X-Auth-Token': 'NC-9-iae9cief24ax3dbcrd-nbi'
}
resp = requests.get(api_url, headers=headers)
discoveries = resp.json()["response"]
for disc in discoveries:
    print("Status\t", "ID\t", "Condition\t", "Name")
    print(disc["discoveryStatus"], "\t", disc["id"], "\t", disc["discoveryCondition"], "\t", disc["name"])

```

Setelah program dijalankan, output akan menampilkan discovery yang ada pada *network controller*, dan status serta id nya.

```

Starting Final Project NP (Python)...
('Status', '\t', 'ID\t', 'Condition\t', 'Name')
('Inactive', '\t', '0', '\t', 'Complete', '\t', 'OFFICE')
('Inactive', '\t', '1', '\t', 'Complete', '\t', 'new_device_detection')
Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

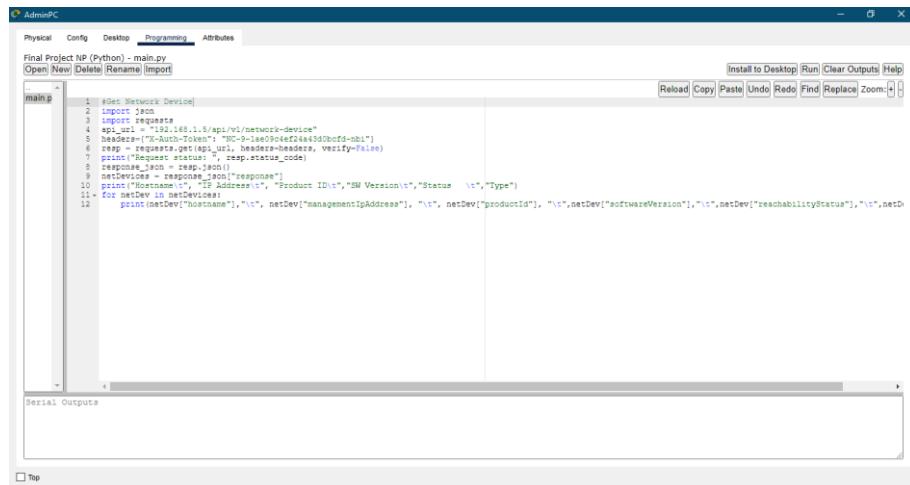
PS C:\Users\neerf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation> & C:/Users/neerf/AppData/Local/Microsoft/WindowsApps/python3.9.exe <:./Users/neerf/Desktop/FINAL PROJECT NP/FinalProjectNP/PTAutomation/GetDiscovery.py
Starting Final Project NP (Python)...
('Status', '\t', 'ID\t', 'Condition\t', 'Name')
('Inactive', '\t', '0', '\t', 'Complete', '\t', 'OFFICE')
('Inactive', '\t', '1', '\t', 'Complete', '\t', 'new_device_detection')
Final Project NP (Python) finished running.

```

## f. Mendapatkan Perangkat Jaringan (Get Network Device)

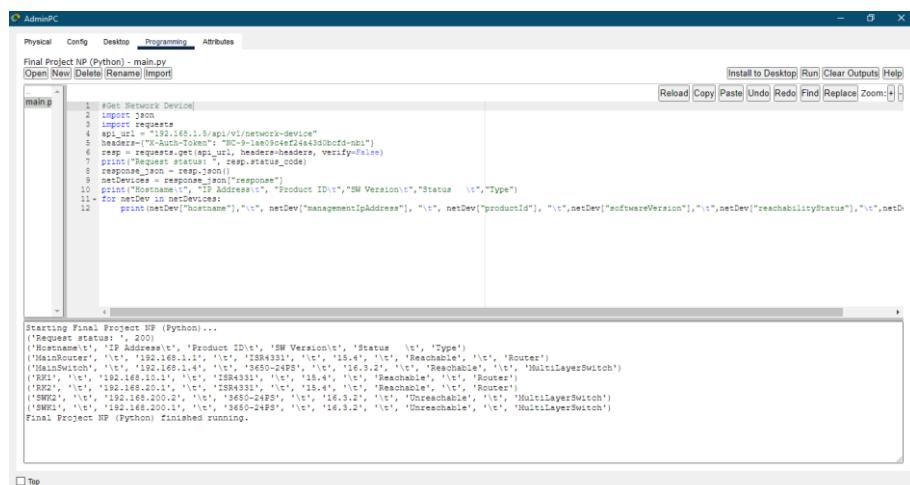
Apabila status discovery sudah complete, artinya *network controller* sudah selesai memindai keseluruhan jaringan. Selanjutnya bisa didapatkan data *network device* yang terhubung ke jaringan.

Pada programnya, isi `api_url` sesuai dengan API documentation (lihat lampiran), dan isi `X-Auth-Token` nya dengan service ticket. Lalu jalankan program.



```
Administrator: C:\Users\Public\Documents\Mininet>python Final Project NP (Python) - main.py
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
main.py
1 #Get Network Device
2 import json
3 import requests
4 api_url = "192.168.1.5/api/v1/network-device"
5 headers = {"X-Auth-Token": "NC9-9-ae0984e72443d0bcd-nbi"}
6 resp = requests.get(api_url, headers=headers, verify=False)
7 print("Request status: ", resp.status_code)
8 response_json = resp.json()
9 netDevices = response_json["response"]
10 print("Hostname\t", "IP Address\t", "Product ID\t", "SM Version\t", "Status\t", "Type")
11 for netDev in netDevices:
12     print(netDev["hostname"], "\t", netDev["managementIpAddress"], "\t", netDev["productId"], "\t", netDev["softwareVersion"], "\t", netDev["reachabilityStatus"], "\t", netD
Serial Output
Top
```

Output program akan menampilkan semua *network device* yang terpindai oleh network discovery dari *network controller*.



```
Administrator: C:\Users\Public\Documents\Mininet>python Final Project NP (Python) - main.py
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
main.py
1 #Get Network Device
2 import json
3 import requests
4 api_url = "192.168.1.5/api/v1/network-device"
5 headers = {"X-Auth-Token": "NC9-9-ae0984e72443d0bcd-nbi"}
6 resp = requests.get(api_url, headers=headers, verify=False)
7 print("Request status: ", resp.status_code)
8 response_json = resp.json()
9 netDevices = response_json["response"]
10 print("Hostname\t", "IP Address\t", "Product ID\t", "SM Version\t", "Status\t", "Type")
11 for netDev in netDevices:
12     print(netDev["hostname"], "\t", netDev["managementIpAddress"], "\t", netDev["productId"], "\t", netDev["softwareVersion"], "\t", netDev["reachabilityStatus"], "\t", netD
Starting Final Project NP (Python)...
('Request status: ', 200)
('Hostname', 'IP Address', 'Product ID', 'SM Version', 'Status', 'Type')
('MainSwitch', '\t', '192.168.1.1', '\t', '3650-24PS', '\t', '16.3.2', '\t', 'Reachable', '\t', 'Router')
('MainSwitch', '\t', '192.168.1.4', '\t', '3650-24PS', '\t', '16.3.2', '\t', 'Reachable', '\t', 'MultilayerSwitch')
('RBL1', '\t', '192.168.10.11', '\t', '1988331', '\t', '15.4', '\t', 'Reachable', '\t', 'Router')
('RBL2', '\t', '192.168.10.12', '\t', '1988331', '\t', '15.4', '\t', 'Reachable', '\t', 'Router')
('SMB1', '\t', '192.168.200.27', '\t', '3650-24PS', '\t', '16.3.2', '\t', 'Unreachable', '\t', 'MultilayerSwitch')
('SMB2', '\t', '192.168.200.1', '\t', '3650-24PS', '\t', '16.3.2', '\t', 'Unreachable', '\t', 'MultilayerSwitch')
Final Project NP (Python) finished running.
```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

File Edit Selection View Go Run Terminal Help Get Network Device.py - PTAutomation - Visual Studio Code
Get Network Device.py > [Run]
Get Network Device.py M Add Network Device.py M Delete Credential M Delete Discovery by ID M Get Network Device.py X Get Host.py M Update > V ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Request status: 200
Hostname IP Address Product ID SW Version Status Type
MainRouter 192.168.1.1 ISW4311 15.4 Reachable Router
RouterSwitch 192.168.1.4 3650-24PS 16.3.2 Reachable MultiLayerSwitch
RIC1 192.168.10.1 ISW4311 15.4 Reachable Router
RIC2 192.168.20.1 ISW4311 15.4 Reachable Router
SWK1 192.168.200.1 3650-24PS 16.3.2 Unreachable MultiLayerSwitch
SWK2 192.168.200.2 3650-24PS 16.3.2 Unreachable MultiLayerSwitch

```

### g. Menambahkan Perangkat Jaringan (Add Network Device)

Apabila dirasa ada *network device* yang belum terpindai, atau tidak sesuai, pada kasus ini SWK1 dan SWK2 yang terpindai adalah IP VLAN200 nya, bukan IP VLAN 1 yang terhubung ke *routernya*. Maka kita bisa menambahkan *network device* secara manual.

Pada programnya, isi `api_url` sesuai dengan API documentation (lihat lampiran), dan isi `X-Auth-Token` nya dengan service ticket. Pada bagian `json_body` isi parameter `ipAddress` dengan IP device yang ingin ditambahkan, dan isi `globalCredentialId` nya dengan ID credential yang ingin kita gunakan. Lalu jalankan program.

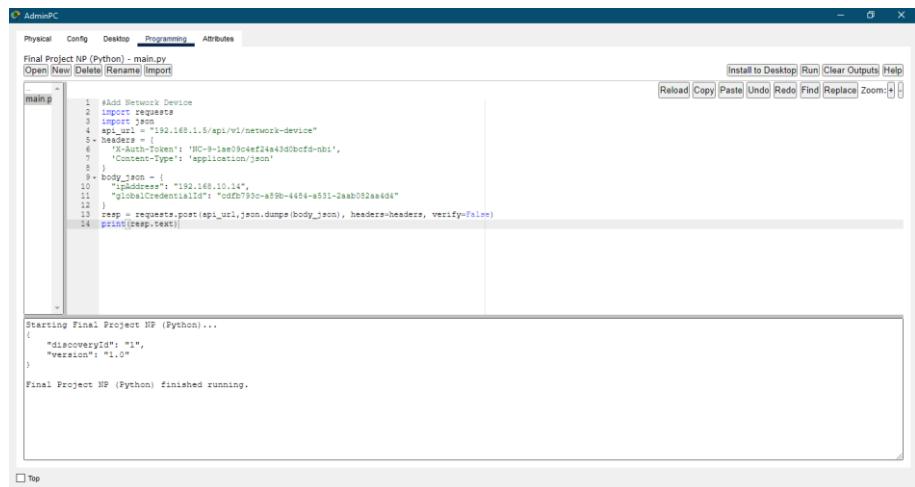
```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
main.py
1 #Add Network Device
2 import requests
3 import json
4 api_url = "192.168.1.5/api/v1/network-device"
5 headers = {
6     'X-Auth-Token': 'NC-3-1am59ceff24a43d0bcfd-nbi',
7     'Content-Type': 'application/json'
8 }
9 body_json = {
10     "ipAddress": "192.168.10.247",
11     "globalCredentialId": "c0fb0793c-a890-4484-a531-2aa8b02aa424"
12 }
13 resp = requests.post(api_url,json.dumps(body_json), headers=headers, verify=False)
14 print(resp.text)

```

Output program akan menampilkan status discovery id apabila device berhasil ditambahkan.

Untuk menambahkan SWK2, lakukan prosedur yang sama.



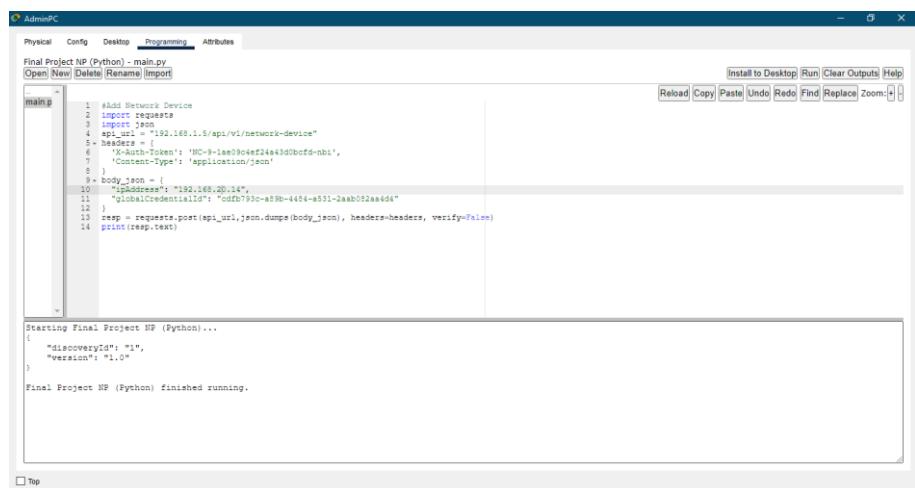
```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
Install to Desktop Run Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: 100%
main.py
1 #Add Network Device
2 import requests
3 import json
4 api_url = "192.168.1.5/api/v1/network-device"
5 headers = {
6     'X-Auth-Token': 'NC-9-lae9c0eff24a43d0bcfd-nbi',
7     'Content-Type': 'application/json'
8 }
9 body_json = {
10     "ipAddress": "192.168.10.14",
11     "globalCredentialId": "ccfd793c-a59b-4484-a531-2ab02aa6d4"
12 }
13 resp = requests.post(api_url,json.dumps(body_json), headers=headers, verify=False)
14 print(resp.text)

Starting Final Project NP (Python)...
{
    "discoveryId": "1",
    "version": "1.0"
}
Final Project NP (Python) finished running.

```

Output program akan menampilkan status discovery id apabila device berhasil ditambahkan.



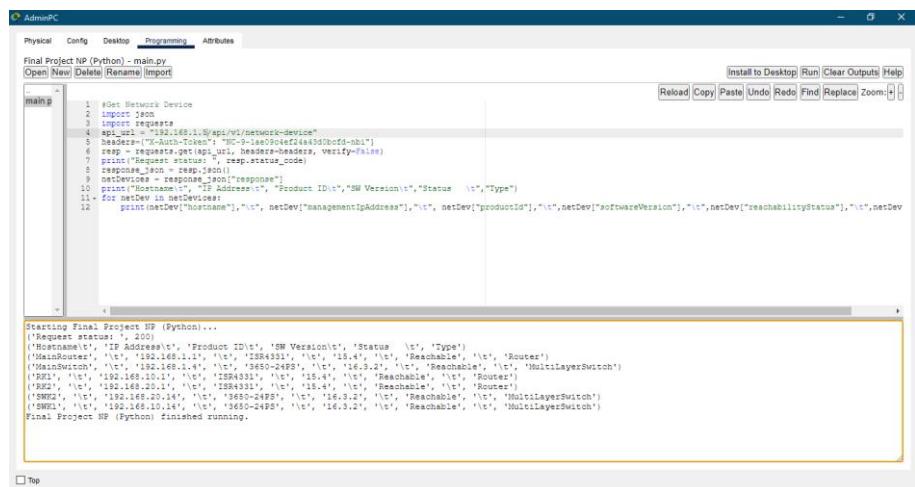
```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
Install to Desktop Run Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: 100%
main.py
1 #Add Network Device
2 import requests
3 import json
4 api_url = "192.168.1.5/api/v1/network-device"
5 headers = {
6     'X-Auth-Token': 'NC-9-lae9c0eff24a43d0bcfd-nbi',
7     'Content-Type': 'application/json'
8 }
9 body_json = {
10     "ipAddress": "192.168.10.14",
11     "globalCredentialId": "ccfd793c-a59b-4484-a531-2ab02aa6d4"
12 }
13 resp = requests.post(api_url,json.dumps(body_json), headers=headers, verify=False)
14 print(resp.text)

Starting Final Project NP (Python)...
{
    "discoveryId": "1",
    "version": "1.0"
}
Final Project NP (Python) finished running.

```

Untuk memvalidasi, bisa kita menjalankan program get *network device* sekali lagi



```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
Open New Delete Rename Import
Install to Desktop Run Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: 100%
main.py
1 #Get Network Device
2 import json
3 import requests
4 api_url = "192.168.1.5/api/v1/network-device"
5 headers = {
6     'X-Auth-Token': 'NC-9-lae9c0eff24a43d0bcfd-nbi'
7 }
8 resp = requests.get(api_url, headers=headers, verify=False)
9 print(json.dumps(resp.json(), indent=4))
10 response_json = resp.json()
11 netDevices = response_json["response"]
12 for netDev in netDevices:
13     print("Hostname: ", netDev["hostAddress"], "- Product ID: ", netDev["productID"], "- SW Version: ", netDev["softwareVersion"], "- Status: ", netDev["reachabilityStatus"])
14     print("Management IP Address: ", netDev["managementIpAddress"], "- Product ID: ", netDev["productID"], "- SW Version: ", netDev["softwareVersion"], "- Status: ", netDev["reachabilityStatus"])

Starting Final Project NP (Python)...
{'Request status': 200}
('Hostname', 'IP Address', 'Product ID', 'SW Version', 'Status', '\n', 'Type')
('SWK2', '192.168.10.14', '3680-24P8', '\n', '15.4.2', '\n', 'Reachable', '\n', 'Router')
('MSwitch', '\n', '192.168.1.47', '\n', '3680-24P8', '\n', '16.3.2', '\n', 'Reachable', '\n', 'MultilayerSwitch')
('RBL', '\n', '192.168.10.1', '\n', '1SRA331', '\n', '15.4', '\n', 'Reachable', '\n', 'Router')
('RBL', '\n', '192.168.20.1', '\n', '1SRA331', '\n', '15.4', '\n', 'Reachable', '\n', 'Router')
('MSW1', '\n', '192.168.10.24', '\n', '3680-24P8', '\n', '16.3.2', '\n', 'Reachable', '\n', 'MultilayerSwitch')
('MSW1', '\n', '192.168.10.14', '\n', '3680-24P8', '\n', '16.3.2', '\n', 'Reachable', '\n', 'MultilayerSwitch')
Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

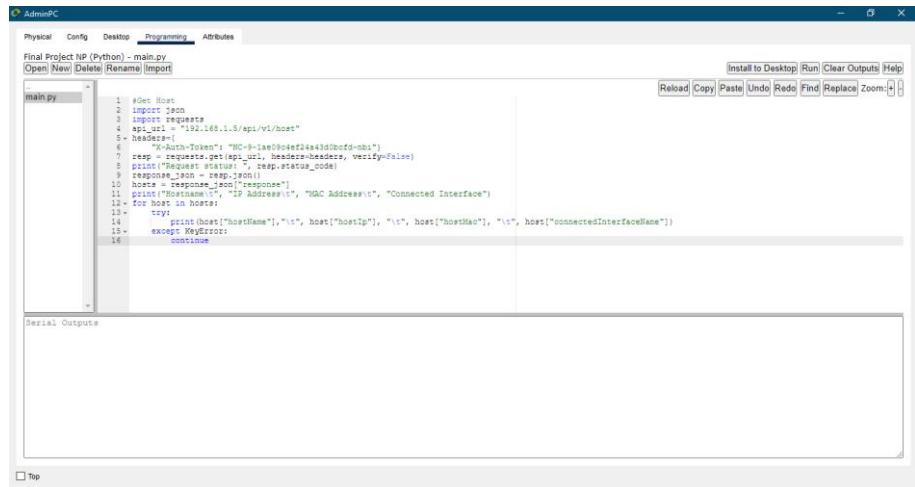
The image consists of three vertically stacked screenshots of the Visual Studio Code interface, each displaying a different Python script related to network automation:

- Screenshot 1:** Shows the `Add Network Device.py` script. The code uses the `requests` library to post data to a local API endpoint. The posted JSON includes an IP address (192.168.20.14) and a global credential ID (28e73115-c45e-41ea-bf47-85fice868c4e).
- Screenshot 2:** Shows the `Get Network Device.py` script. This script uses the `requests` library to get data from a local API endpoint. It prints out network device details such as Hostname, IP Address, Product ID, SW Version, Status, and Type.
- Screenshot 3:** Shows the `Get Host.py` script. This script also uses the `requests` library to get data from a local API endpoint. It prints out host details including management IP address, product ID, software version, and type.

#### h. Mendapatkan Host (Get Host)

Selain dari *network device*, *network controller* bisa juga memonitor *host* yang terhubung.

Pada programnya, isi api\_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.

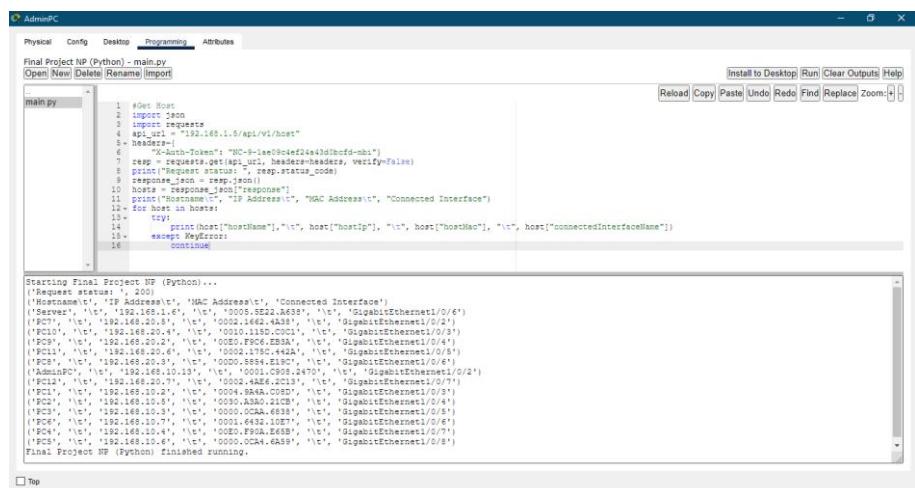


```

Administrator: Final Project NP (Python) - main.py
Physical Config Desktop Programming Attributes
File Project New Delete Rename Import
main.py
1 #Get Host
2 import json
3 import requests
4 api_url = "192.168.1.5/api/v1/host"
5 headers = {
6     "X-Auth-Token": "NC-4-iae30cf143d0cfd-n8z"
7 }
8 resp = requests.get(api_url, headers=headers, verify=False)
9 print("Request status: ", resp.status_code)
10 response_json = resp.json()
11 print("Hostname:", "IP Address", "MAC Address", "Connected Interface")
12 for host in hosts:
13     try:
14         print(host["hostName"], host["hostIp"], host["hostMac"], host["connectedInterfaceName"])
15     except KeyError:
16         continue

```

Output program akan menampilkan data dari *host* yang terhubung ke jaringan.



```

Administrator: Final Project NP (Python) - main.py
Physical Config Desktop Programming Attributes
File Project New Delete Rename Import
main.py
1 #Get Host
2 import json
3 import requests
4 api_url = "192.168.1.5/api/v1/host"
5 headers = {
6     "X-Auth-Token": "NC-4-iae30cf143d0cfd-n8z"
7 }
8 resp = requests.get(api_url, headers=headers, verify=False)
9 print("Request status: ", resp.status_code)
10 response_json = resp.json()
11 print("Hostname", "IP Address", "MAC Address", "Connected Interface")
12 for host in hosts:
13     try:
14         print(host["hostName"], host["hostIp"], host["hostMac"], host["connectedInterfaceName"])
15     except KeyError:
16         continue

```

Starting Final Project NP (Python)...
('Request status: ', 200)
('Hostname', 'IP Address', 'MAC Address', 'Connected Interface')
('hostName1', '192.168.20.1', '0000.5522.6633', '1', 'GigabitEthernet1/0/4')
('FCT1', '192.168.20.1', '0000.1162.4433', '1', 'GigabitEthernet1/0/2')
('FCT2', '192.168.20.4', '0010.1180.CC01', '1', 'GigabitEthernet1/0/3')
('FCT3', '192.168.20.2', '0000.1162.4433', '1', 'GigabitEthernet1/0/1')
('FCT4', '192.168.20.6', '0000.1162.4421', '1', 'GigabitEthernet1/0/5')
('FCT5', '192.168.20.9', '0000.5584.EE8C', '1', 'GigabitEthernet1/0/6')
('AdminPC', '192.168.10.131', '0000.C900.2470', '1', 'GigabitEthernet1/0/2')
('FCT6', '192.168.10.1', '0000.AA8B.CC01', '1', 'GigabitEthernet1/0/3')
('FCT7', '192.168.10.9', '0000.AA8B.CC02', '1', 'GigabitEthernet1/0/4')
('FCT8', '192.168.10.8', '0000.AA8B.CC03', '1', 'GigabitEthernet1/0/5')
('FCT9', '192.168.10.7', '0000.6432.10E7', '1', 'GigabitEthernet1/0/6')
('FCT10', '192.168.10.4', '0000.F900.EE8B', '1', 'GigabitEthernet1/0/7')
('FCT11', '192.168.10.6', '0000.5C48.6A59', '1', 'GigabitEthernet1/0/8')
Final Project NP (Python) finished running.

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

#### i. Memperbarui Wide Setting Jaringan (Update Network Wide Setting)

Dari *network controller*, bisa juga melakukan manajemen perangkat jaringan seperti menambahkan beberapa layanan, misalnya DNS, dan NTP. Disini akan diupdate konfigurasi untuk DNS pada seluruh *network device*, dan menambahkan layanan NTP.

Sebelumnya, cek terlebih dahulu keadaan konfigurasi DNS dan NTPnya. Gambar dibawah ini menunjukan konfigurasi DNS pada *MainRouter* sebelum dilakukan update konfigurasi.

```
AdminPC

Physical Config Desktop Programming Attributes

Command Prompt X

interface Vlan1
no ip address
shutdown
!
ip classless
ip route 192.168.10.0 255.255.255.240 192.168.1.2
ip route 192.168.20.0 255.255.255.240 192.168.1.3
!
ip flow-export version 9
!
!
banner motd ^Authorized Access Only!^C
!
!
!
line con 0
password 7 08315E41031C0603
login
!
line aux 0
password 7 08315E41031C0603

MainRouter# show hosts
Default Domain is lastproject.com
Name/address lookup uses Domain service
Name servers are 255.255.255.255

Codes: UN - unknown, EX - expired, OK - OK, ?? - revalidate
      temp - temporary, perm - permanent
      Na - Not Applicable None - Not defined

Host          Port Flags    Age Type   Address(es)
MainRouter#
```

Gambar dibawah ini menunjukan konfigurasi NTP pada *MainRouter* sebelum dilakukan update konfigurasi.

```

AdminPC
Physical Config Desktop Programming Attributes
Command Prompt
[...]
lmcn com 0
Password 7 08315E41031C0603
login
lmcn aux 0
Password 7 08315E41031C0603
login
lmcn vty 0 4
Password 7 08315E41031C0603
login local
transport input ssh
lmcn vty 5 6
Password 7 08315E41031C0603
login local
transport input ssh
!
ntp update-calendar
!
end

MainRouter#sh ntp status
Clock is unsynchronized, stratum 16, no reference clock
Nominal freq is 10.000000 Hz, actual freq is 9.999950 Hz, precision is 2**24
reference time is 00000000.00000000 (0010:010.000 UTC Mon Jan 1 1990)
clock offset is 0.00 msec, root delay is 0.00 msec
root dispersion is 0.00 msec, peer dispersion is 0.00 msec.
RootDrift state is 'FSET' (Drift set from file), drift is - 0.000001193 s/s system poll interval is 4, never updated.
MainRouter#

```

Top

Untuk mengupdate konfigurasi DNS dan NTP pada semua *network device*, pada programnya isi `api_url` sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Pada `body_json`, isi parameter IP *server* NTP, name address, dan IP Adress DNS. Lalu jalankan program.

```

AdminPC
Physical Config Desktop Programming Attributes
Final Project NP (Python) - main.py
File New Open Recent Import
Install to Desktop Run Clear Outputs Help
Reload Copy Paste Undo Redo Find Replace Zoom: 100%
main.py
1 #Update Network Wide Setting
2 import requests
3 import json
4 headers = {
5     "X-Auth-Token": "NC-9-las9c94ef24a43d0bcd-nb1",
6     "Content-Type": "application/json"
7 }
8 api_url = "http://192.168.1.5/api/v1/wan/network-wide-setting"
9 body_json = {
10     "http": [
11         {
12             "serverIp": "192.168.1.6"
13         },
14         {
15             "name": "finalproject.com",
16             "ipAddress": "192.168.1.6"
17         }
18     ]
19 }
20 resp = requests.put(api_url, json.dumps(body_json), headers=headers, verify=False)
21 response_json = resp.json()
22 print("DNS IP Address:", response_json["settings"]["dns"]["ipAddress"])
23 print("DNS Name:", response_json["settings"]["dns"]["name"])
24 print("NTP Server IP:", response_json["ntp"]["serverIp"])
25
Serial Outputs

```

Top

Output program akan menampilkan konfigurasi yang diupdate ke *network device*.

```

main.py
1 #Update Network Wide Setting
2 import requests
3 import json
4 headers = {
5     'X-Auth-Token': 'NC-9-lae09c4ef24a43d0bcd-nb1',
6     'Content-Type': 'application/json'
7 }
8 api_url = "http://192.168.1.5/api/v1/wan/network-wide-setting"
9 body_json = [
10     {
11         "serverIp": "192.168.1.6"
12     },
13     {
14         "name": "finalproject.com",
15         "ipAddress": "192.168.1.6"
16     }
17 ]
18 resp = requests.put(api_url, json.dumps(body_json), headers=headers, verify=False)
19 response_json = resp.json()
20 setting=response_json['response']
21 dns_name=setting["dns"]["dns"][0]["name"]
22 print ("DNS Name:", setting["dns"][0]["name"])
23 print ("NTP Server IP:", setting["ntp"]["serverIp"])
24

```

Starting Final Project NP (Python)...
('DNS Address', '192.168.1.6')
('DNS Name', 'finalproject.com')
('NTP Server IP', '192.168.1.6')
Final Project NP (Python) finished running.

Bisa dicek kembali, DNS sudah terupdate di *network device*, pada gambar ini, yang terupdate adalah konfigurasi di *MainRouter*.

```

Authorized Access Only!
MainRouter>en
Password:
MainRouter#sh hosts
Default Domain is finalproject.com
Name/address lookup uses domain service
Name servers are 192.168.1.6

Codes: UN - unknown, EX - expired, OK - OK, ?? - revalidate
      temp - temporary, perm - permanent
      NA - Not Applicable None - Not defined

Host          Port  Flags    Age Type   Address(es)
MainRouter#sh hosts
Default Domain is finalproject.com
Name/address lookup uses domain service
Name servers are 192.168.1.6

Codes: UN - unknown, EX - expired, OK - OK, ?? - revalidate
      temp - temporary, perm - permanent
      NA - Not Applicable None - Not defined

Host          Port  Flags    Age Type   Address(es)
MainRouter#sh hosts
Default Domain is finalproject.com
Name/address lookup uses domain service
Name servers are 192.168.1.6

Codes: UN - unknown, EX - expired, OK - OK, ?? - revalidate
      temp - temporary, perm - permanent
      NA - Not Applicable None - Not defined

Host          Port  Flags    Age Type   Address(es)
MainRouter#

```

Dan NTP juga sudah terupdate di *network device*, pada gambar ini, yang terupdate adalah konfigurasi di *MainRouter*.

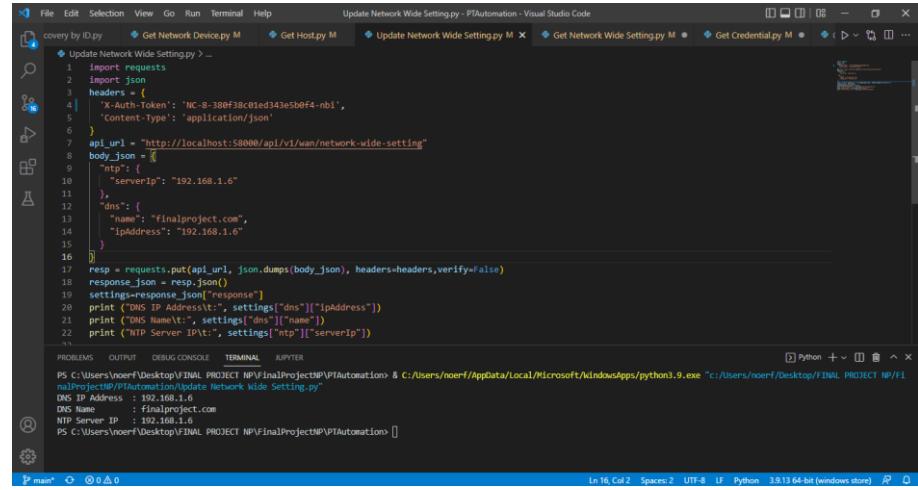
```

MainRouter>
:
:
line con 0
password 7 08315E41031C0603
login
:
line aux 0
password 7 08315E41031C0603
login
:
line vty 0 4
password 7 08315E41031C0603
login local
transport input ssh
line vty 5 15
password 7 08315E41031C0603
login local
transport input ssh
:
ntp server 192.168.1.6
ntp update-calendar
:
end

MainRouter#sh ntp status
Clock is synchronized, stratum 2, reference is 192.168.1.6
nominal freq is 250.0000 Hz, actual freq is 249.9990 Hz, precision is 2**24
reference time is E6415DCS.00000229 (3:38:49.761 UTC Sat Jul 2 2022)
clock offset is -3.00 msec, root delay is 1.00 msec
root dispersion is 174.36 msec, peer dispersion is 0.12 msec.
loop discipline state is 'CRL' (Normal Controlled Loop), drift is -0.000001193 s/m system poll interval is 4, last update was 3 sec ago.
MainRouter#

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



```

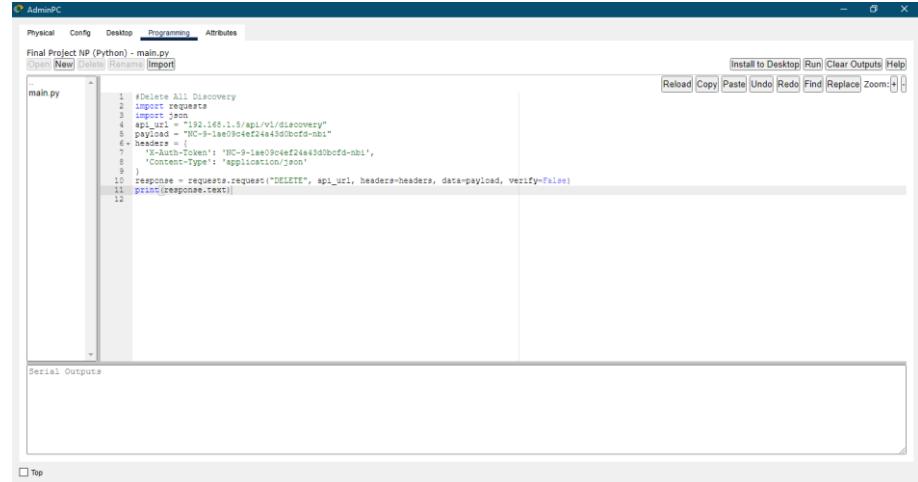
    1 import requests
    2 import json
    3 headers = {
    4     'X-Auth-Token': 'NC-8-380f38c0led343e5b0f4-nbi',
    5     'Content-Type': 'application/json'
    6 }
    7 api_url = "http://localhost:58000/api/v1/wan/network-wide-setting"
    8 body_json = []
    9     "ntp": {
    10         "serverIp": "192.168.1.6"
    11     },
    12     "dns": {
    13         "name": "finalproject.com",
    14         "ipAddress": "192.168.1.6"
    15     }
    16
    17 resp = requests.put(api_url, json.dumps(body_json), headers=headers, verify=False)
    18 response_json = resp.json()
    19 settings=response_json['response']
    20 print ("DNS IP Address:",settings["dns"]["ipAddress"])
    21 print ("DNS Name(s):", settings["dns"]["name"])
    22 print ("NTP Server IP:", settings["ntp"]["serverIp"])
    23

```

#### j. Menghapus Semua Discovery (Delete All Discovery)

Apabila dirasa tidak membutuhkan lagi semua discovery, discovery pada *network controller* bisa dihapus, pada program ini akan menghapus seluruh discovery yang ada pada *network controller*.

Pada programnya, isi api\_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.



```

    1 #Delete All Discovery
    2 import requests
    3 import json
    4 api_url = "http://localhost:58000/api/v1/wan/discovery"
    5 payload = "NC-9-1ae09c1ed24443d0bcfd-nbi"
    6 headers = {
    7     'X-Auth-Token': 'NC-9-1ae09c1ed24443d0bcfd-nbi',
    8     'Content-Type': 'application/json'
    9 }
    10 response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
    11 print(response.text)
    12

```

Output program akan menampilkan response true apabila sudah berhasil menghapus semua discovery.

```

main.py
1 #Delete All Discovery
2 import requests
3 import json
4 api_url = "http://192.168.1.8/api/v1/discovery"
5 payload = {"NC-9-lae09e1ef24a45d0bcfd-nbl"
6 - Headers
7 | 'X-Auth-Token': 'NC-9-lae09e1ef24a45d0bcfd-nbl',
8 | 'Content-Type': 'application/json'
9 }
10 response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
11 print(response.text)
12

Starting Final Project NP (Python)...
{
    "response": true,
    "version": "1.0"
}
Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

Delete All Discovery.py - PTAutomation - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Delete All Discovery.py M Get Network Wide Setting.py M Get Credential.py M Get Ticket.py M Delete Ticket.py M Delete All Discovery.py X Add Discovery.py M D v ... 

Delete All Discovery.py ...
1 import requests
2 import json
3 api_url = "http://localhost:58000/api/v1/discovery"
4 payload = {"NC-8-380f98c0fe343e5b0f4-nbl"
5 headers = {
6 | 'X-Auth-Token': 'NC-8-380f98c0fe343e5b0f4-nbl',
7 | 'Content-Type': 'application/json'
8 }
9 response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
10 print(response.text)
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Jupyter
PS C:\Users\noerF\Desktop\FINAL PROJECT NP\FinalProject\P\PTAutomation> python main.py
(
    "response": true,
    "version": "1.0"
)

PS C:\Users\noerF\Desktop\FINAL PROJECT NP\FinalProject\P\PTAutomation>

```

#### k. Menghapus Kredensial (Delete Credential)

Selain discovery, credential pada *network controller* juga bisa dihapus, pada program ini akan menghapus credential yang ada pada *network controller*.

Pada programnya, isi api\_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket, dan pada payload, isikan id credential yang ingin dihapus. Lalu jalankan program.

```

main.py
1 #Delete Credential
2 import requests
3 import json
4 api_url = "192.168.1.8/api/v1/global-credential/odfb793c-a89b-4484-a551-2aab02aa4d4"
5 payload = "odfb793c-a89b-4484-a551-2aab02aa4d4"
6 headers = {
7     'X-Auth-Token': 'NC-9-las09cef24a43d0bcfd-nbl',
8     'Content-Type': 'application/json'
9 }
10 response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
11 print(response.text)

```

Output program akan menampilkan response true apabila sudah berhasil menghapus discovery.

```

Starting Final Project NP (Python)...
{
    "response": true,
    "version": "1.0"
}
Final Project NP (Python) finished running.

```

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

```

Delete Credential.py - PIAutomation - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Delete Credential.py M × Delete Discovery by ID.py M × Get Network Device.py M × Get Host.py M × Update Network Wide Setting.py M × Get Network Wide Settin... ×

Delete Credential.py -
1 import requests
2 import json
3 api_url = "http://localhost:58000/api/v1/global-credential/79a3a9dc-f89f-4caa-b958-b0eecd07231"
4 payload = "79a3a9dc-f89f-4caa-b958-b0eecd07231"
5 headers = {
6     'X-Auth-Token': 'NC-9-380f38c0led343e5b0f4-nbl',
7     'Content-Type': 'application/json'
8 }
9 response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
10 print(response.text)
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL APIYER
PS C:\Users\noerF\Desktop\FINAL PROJECT NP\Final Project\NP\PIAutomation> python3.9 Delete Credential.py
(
    "response": true,
    "version": "1.0"
)

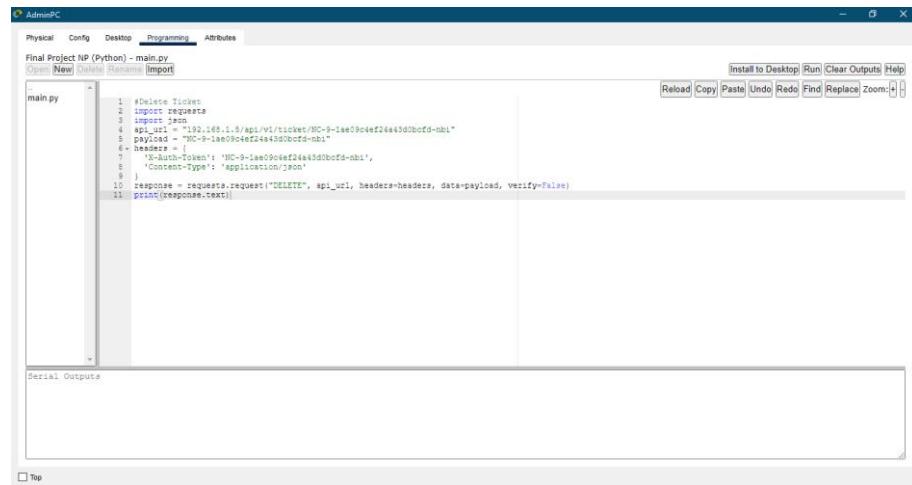
PS C:\Users\noerF\Desktop\FINAL PROJECT NP\Final Project\NP\PIAutomation>

```

## 1. Menghapus Service Ticket (Delete Ticket)

Apabila sudah tidak dibutuhkan, service ticket pada *network controller* juga bisa dihapus, pada program ini akan menghapus service ticket yang ada pada *network controller*.

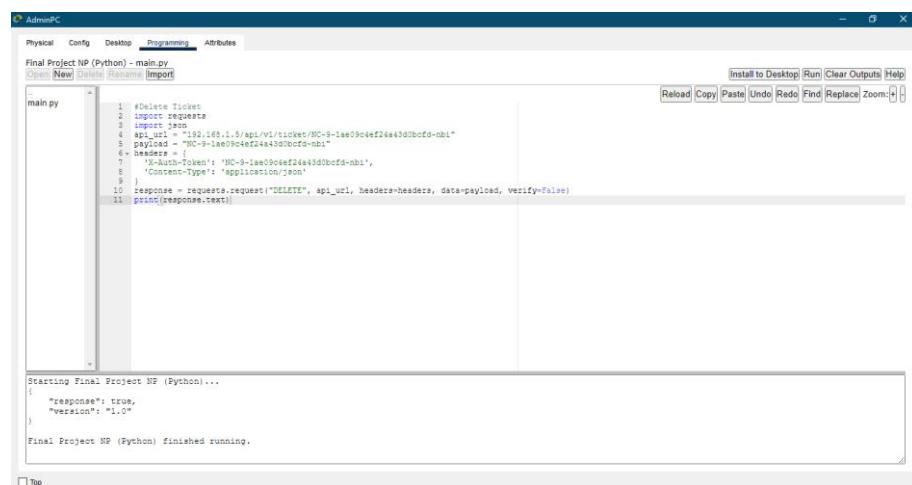
Pada programnya, isi `api_url` sesuai dengan API documentation (lihat lampiran), isi `X-Auth-Token` nya dengan service ticket, dan isi payload dengan nomor ticket yang akan dihapus. Lalu jalankan program.



The screenshot shows the AdminPC IDE interface. The menu bar includes Physical, Config, Desktop, Programming, and Attributes. The Programming tab is selected. A toolbar below the menu has options: Install to Desktop, Run, Clear Outputs, Help, Reload, Copy, Paste, Undo, Redo, Find, Replace, and Zoom. The main workspace displays a Python script named `main.py`. The code is as follows:

```
1 #Delete Ticket
2 import requests
3 import json
4 api_url = "192.168.1.8/api/v1/ticket/NC-9-lae09c4ef244a43d0bcfd-nbi"
5 payload = "NC-9-lae09c4ef244a43d0bcfd-nbi"
6 headers = {
7     'X-Auth-Token': 'NC-9-lae09c4ef244a43d0bcfd-nbi',
8     'Content-Type': 'application/json'
9 }
10 response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
11 print(response.text)
```

Output program akan menampilkan response true apabila ticket sudah berhasil terhapus.



The screenshot shows the AdminPC IDE interface after the script has been run. The status bar at the bottom indicates "Starting Final Project NP (Python)...". The output window shows the following text:

```
{ "response": true,
  "version": "1.0"
}
```

Final Project NP (Python) finished running.

Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `Update Network Wide Setting.py M`, `Get Network Wide Setting.py M`, `Get Credential.py M`, `Get Ticket.py M`, `Delete Ticket.py M`, and `Delete All Discovery.py M`.
- Code Editor:** Displays Python code for deleting a ticket. The code imports `requests`, defines an API URL, sets headers including an auth token, and sends a DELETE request.
- Terminal:** Shows the command `python Delete Ticket.py` being run, with the output indicating success: "{'response': true, 'version': '1.0'}".
- Status Bar:** Shows file statistics: Line 6, Col 49, Spaces: 2, UITF-8, CR/LF, Python, 3.9.13 64-bit (windows store).

## **BAB III**

### **PENUTUP**

#### A. Kesimpulan

Dari simulasi yang sudah dilakukan pada final project ini, dapat disimpulkan bahwa untuk melakukan manajemen dan pemantauan jaringan pada skala kecil bisa dilakukan secara scripting melalui REST\_API *network controller*.

Dengan mengimplementasikan programabilitas jaringan menggunakan fitur REST-API pada *network controller*, proses manajemen dan pemantauan lebih efektif, efisien, dan mudah dilakukan.

#### B. Saran

Pada final project ini, network simulation tool yang digunakan masih terbatas dalam hal implementasi pemrograman jaringan. Untuk mengeksplorasi kapabilitas dan fitur programabilitas jaringan secara lebih luas, bisa digunakan network simulation tool yang lain (seperti GNS3, Cisco VirtualLab, dan lainnya) dan tools-tools automasi lainnya yang tersedia (Ansible, Netmiko, dan lainnya).

## LAMPIRAN

Tautan api\_url merujuk pada API Documentation di *network controller*

<b>Nama Program</b>	<b>api_url</b>
Get Ticket	http://(netcontrollerIP)/api/v1/ticket
Add Credential	http://(netcontrollerIP)/api/v1/global-credential/cli
Get Credential	http://(netcontrollerIP)/api/v1/global-credential/cli
Add Discovery	http://(netcontrollerIP)/api/v1/discovery
Get Discovery	http://(netcontrollerIP)/api/v1/discovery
Get <i>Network Device</i>	http://(netcontrollerIP)/api/v1/network-device
Add Network Device	http://(netcontrollerIP)/api/v1/network-device
Get Host	http://(netcontrollerIP)/api/v1/host
Update Network Wide Setting	http://(netcontrollerIP)/api/v1/wan/network-wide-setting
Delete All Discovery	http://(netcontrollerIP)/api/v1/discovery
Delete Credential	http://(netcontrollerIP)/api/v1/global-credential/(credentialId)
Delete Ticket	http://(netcontrollerIP)/api/v1/ticket/(ticketNumber)