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

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

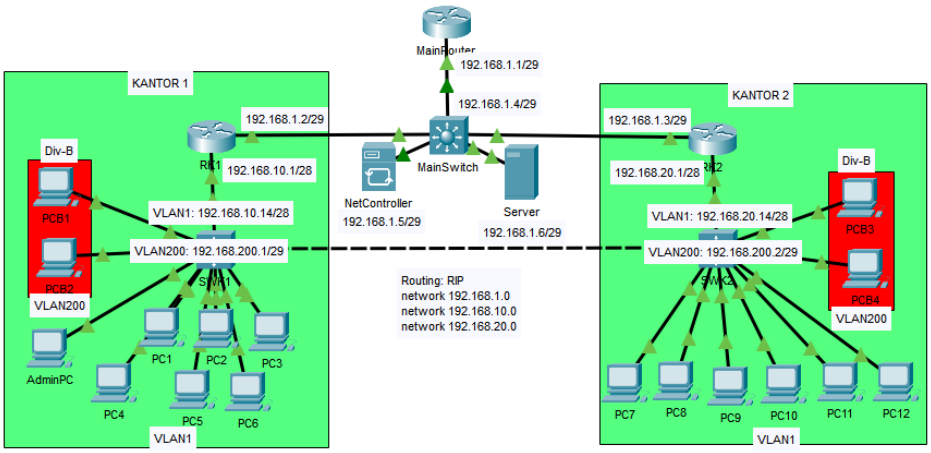
PEMBAHASAN

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. Pengalamatan

Tabel berikut memuat tentang konfigurasi pengalamatan untuk setiap elemen jaringan.

Device	Interface	IP Address
MainRouter	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
MainSwitch	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
Server	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

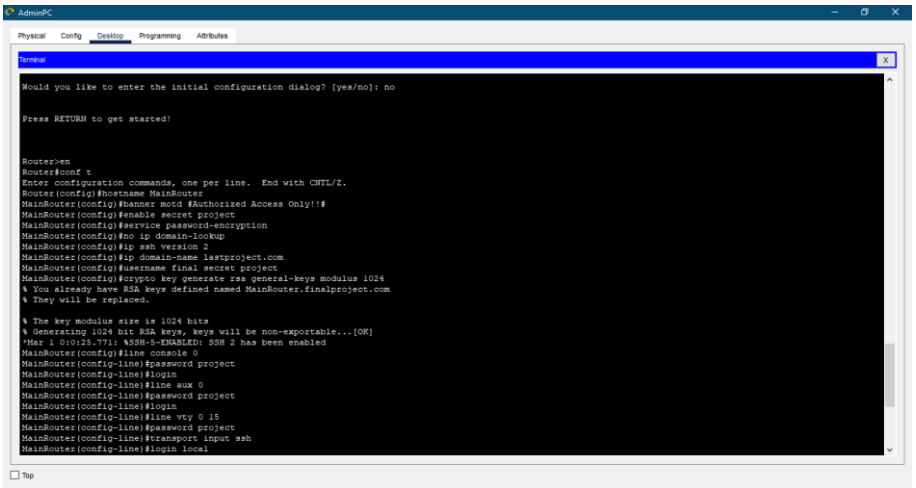
4. Konfigurasi Perangkat Jaringan

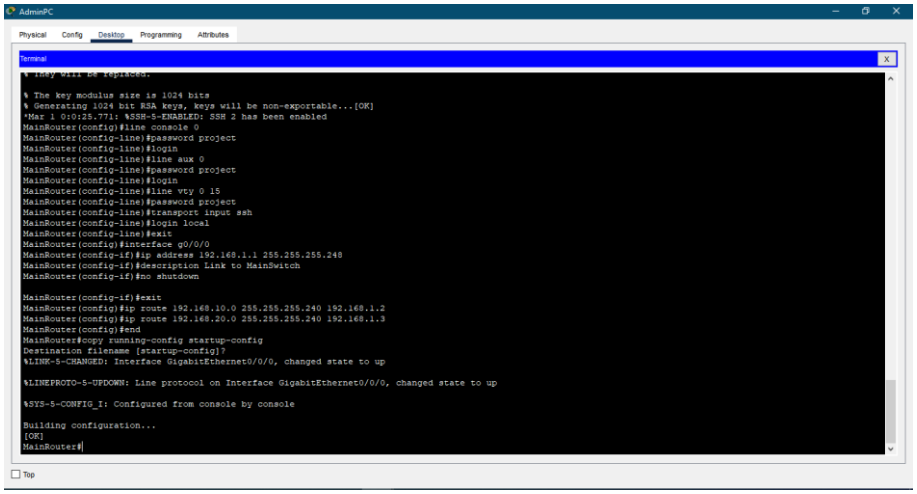
a. *Router* Utama (*MainRouter*)

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

Parameter	Keterangan
Konfgurasi Dasar	
*Host*name	*MainRouter*
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
Pengalamatan	
IP Address	Interface G0/0/0: 192.168.1.1/29
Routing (RIP)	• *Network*: 192.168.1.0 • *Network*: 192.168.10.0 • *Network*: 192.168.20.0

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





b. *Router* Kantor 1 (RK1)

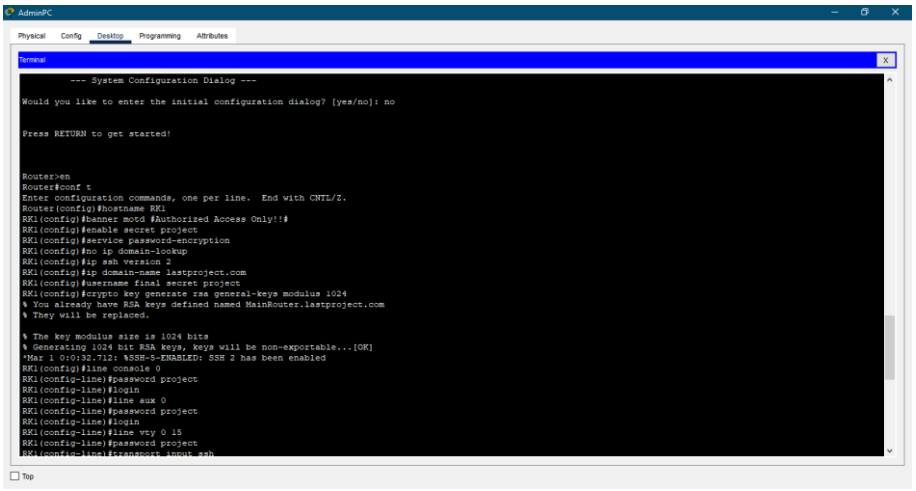
Pada *router* kantor 1, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *router*, dan konfigurasi pengalamatan. Selain pengalamatan dasar, dikonfiguasi juga DHCP untuk melayani *host* di LAN Kantor 1.

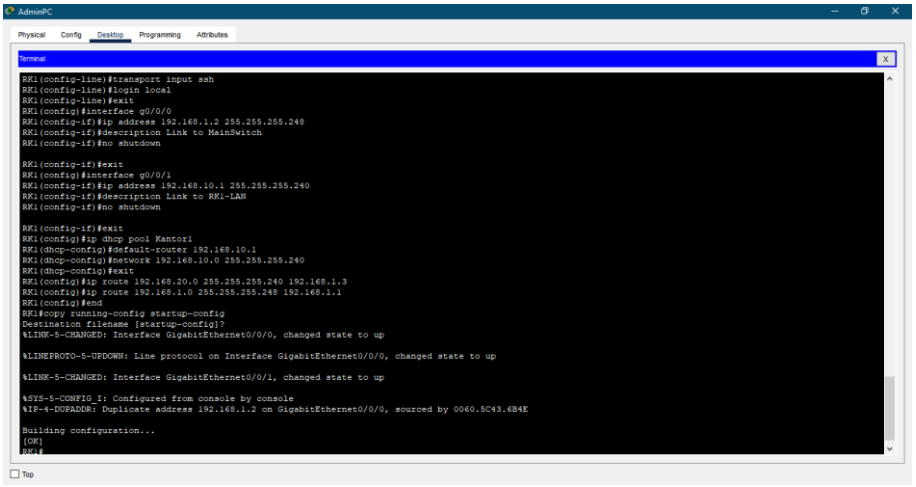
Tabel berikut ini menunjukan parameter konfigurasi yang dilakukan pada perangkat RK1.

Parameter	**Keterangan**
Konfgurasi Dasar	
*Host*name	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
Pengalamatan	
IP Address	• Interface G0/0/0: 192.168.1.2/29 • Interface G0/0/1: 192.168.10.1/28

Routing (RIP)	• *Network*: 192.168.1.0 • *Network*: 192.168.10.0 • *Network*: 192.168.20.0
DHCP	Pool: Kantor1 Default-*router*: 192.168.10.1 Network: 192.168.10.0/29

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





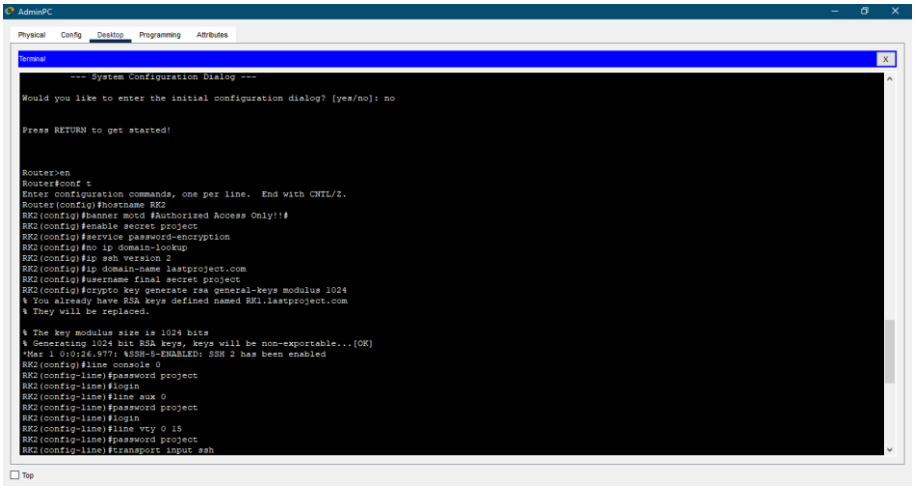
c. *Router* Kantor 2 (RK2)

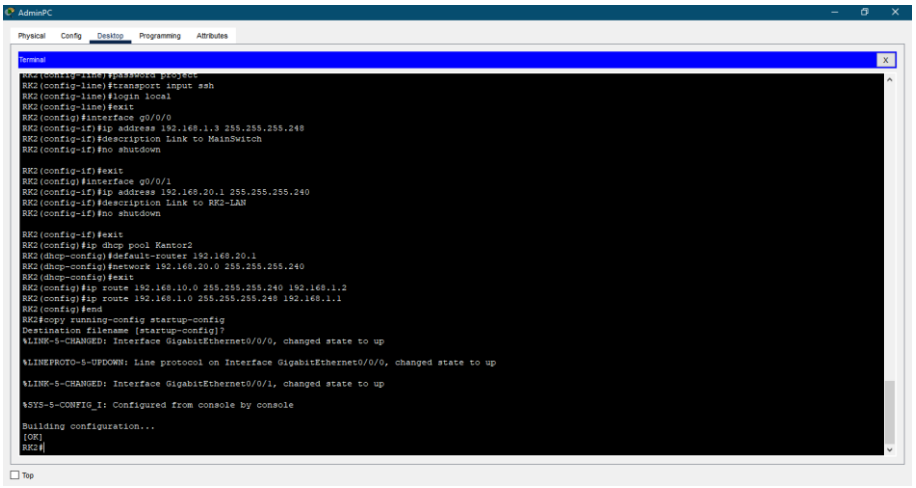
Pada *router* kantor 2, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *router*, dan konfigurasi pengalamatan. Selain pengalamatan dasar, dikonfiguasi juga DHCP untuk melayani *host* di LAN Kantor 2.

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

Parameter	Keterangan
Konfgurasi Dasar	
*Host*name	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
Pengalamatan	
IP Address	• Interface G0/0/0: 192.168.1.3/29 • Interface G0/0/1: 192.168.20.1/28
Routing (RIP)	• *Network*: 192.168.1.0 • *Network*: 192.168.10.0 • *Network*: 192.168.20.0
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.





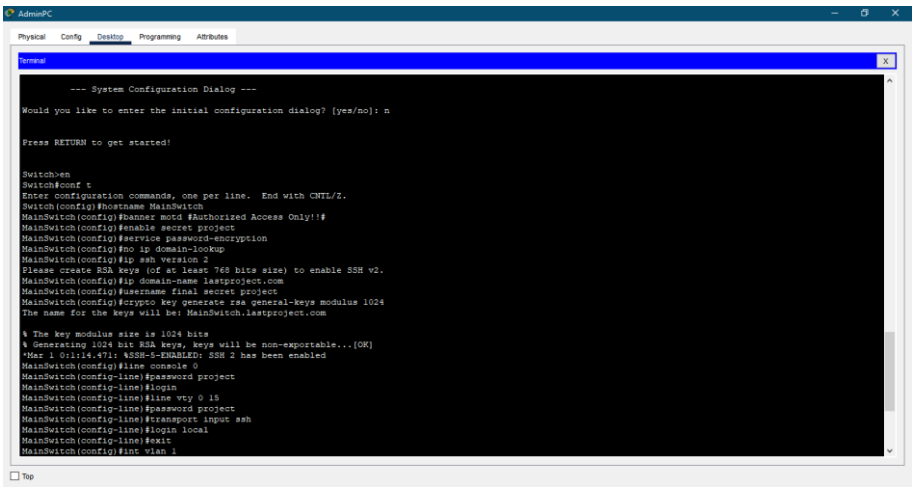
d. *Switch* Utama (*Main Switch*)

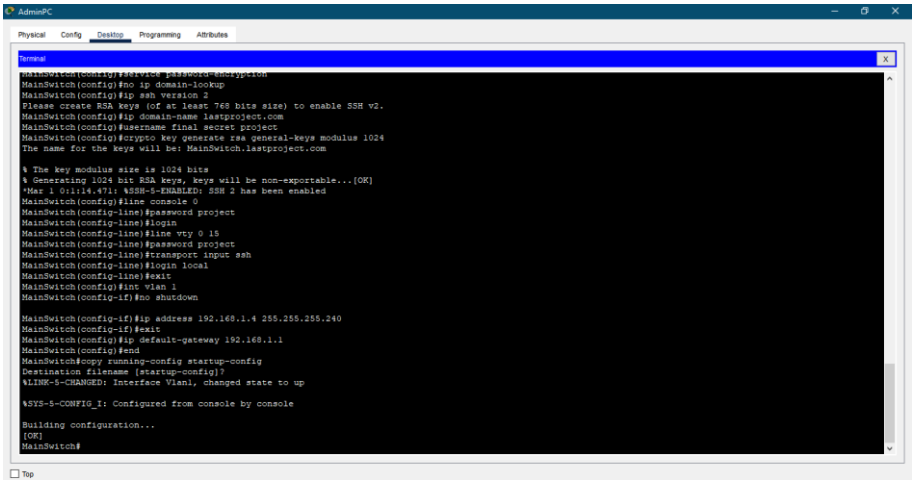
Pada *switch* utama, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *switch*, dan konfigurasi pengalamatan untuk remote akses SSH. Tabel berikut ini menunjukan parameter konfigurasi yang dilakukan pada perangkat *MainSwitch*.

Parameter	**Keterangan**
Konfgurasi Dasar	
*Host*name	*MainSwitch*
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
Pengalamatan	
IP Address	Interface VLAN1: 192.168.1.4/29 Default-gateway: 192.168.1.1
VLAN Database	
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*.





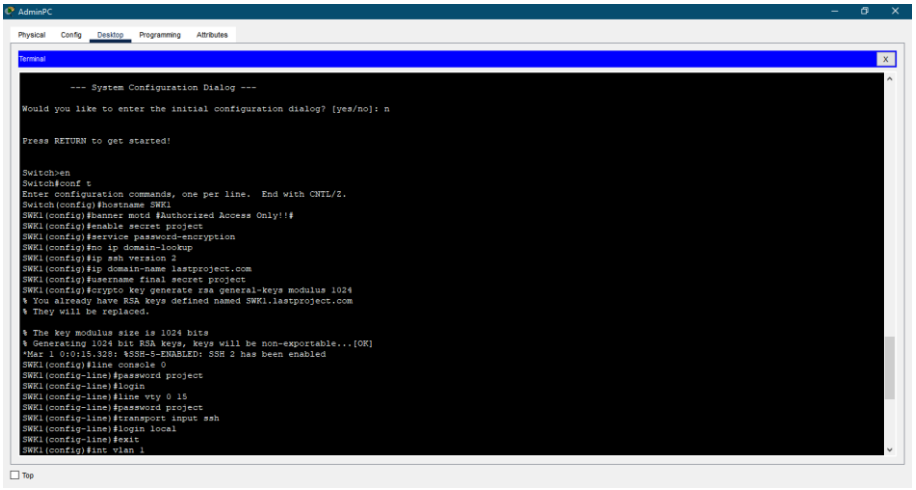
e. *Switch* Kantor 1 (SWK1)

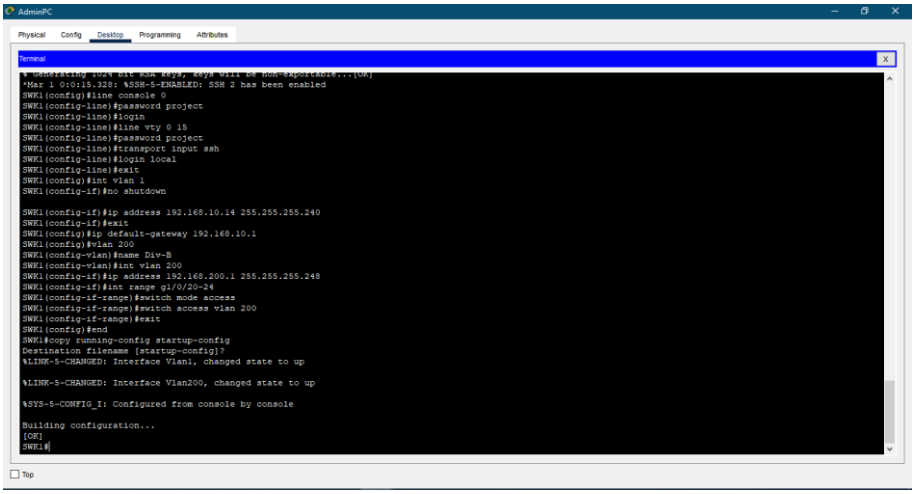
Pada *switch* kantor 1, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *switch*, dan konfigurasi pengalamatan untuk

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

Parameter	Keterangan
Konfgurasi Dasar	
*Host*name	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
Pengalamatan	
IP Address	• Interface VLAN1: 192.168.10.14/29 • Interface VLAN200: 192.168.200.1/29 Default-gateway: 192.168.10.1
VLAN Database	
Interface (nama)	• VLAN1 (default) G1/0/1-19, G1/1/1-4 • VLAN200 (Div-B) G1/0/20-24 Mode Access

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





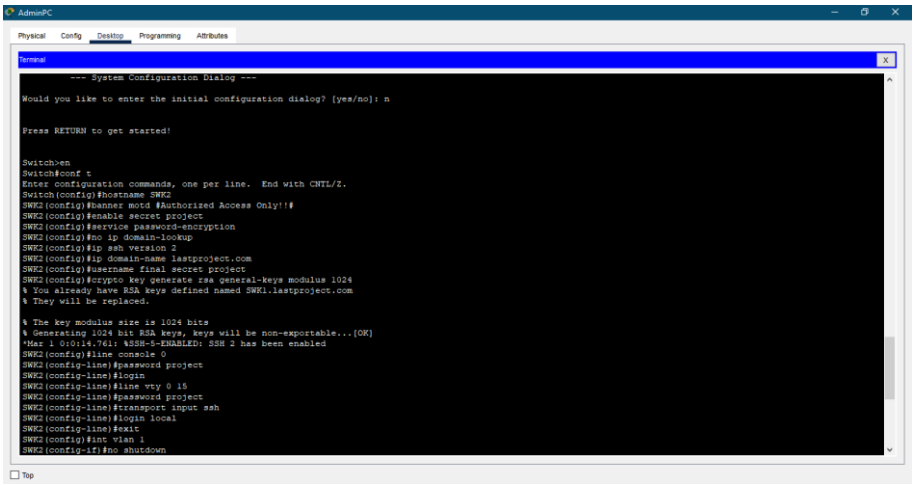
f. *Switch* Kantor 2 (SWK2)

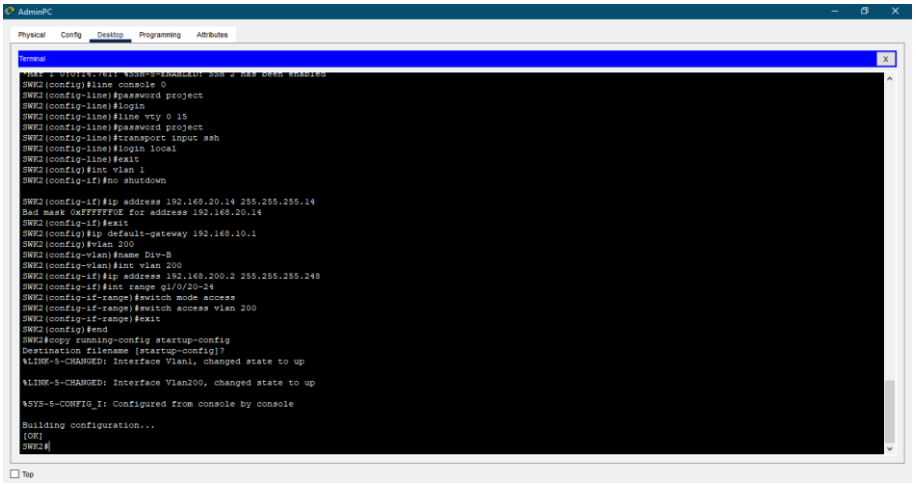
Pada *switch* kantor 2, konfigurasi yang dilakukan meliputi konfigurasi dasar seperti keamanan *switch*, dan konfigurasi pengalamatan untuk remote akses SSH. Dikonfigurasi juga vlan untuk akses privat antar kantor. Tabel berikut ini menunjukan parameter konfigurasi yang dilakukan pada perangkat SWK2.

Parameter	**Keterangan**
Konfgurasi Dasar	
*Host*name	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
Pengalamatan	
IP Address	• Interface VLAN1: 192.168.20.14/29 • Interface VLAN200: 192.168.200.1/29 Default-gateway: 192.168.20.1
VLAN Database	
Interface (nama)	• VLAN1 (default) G1/0/1-19, G1/1/1-4 • VLAN200 (Div-B) G1/0/20-24 Mode Access

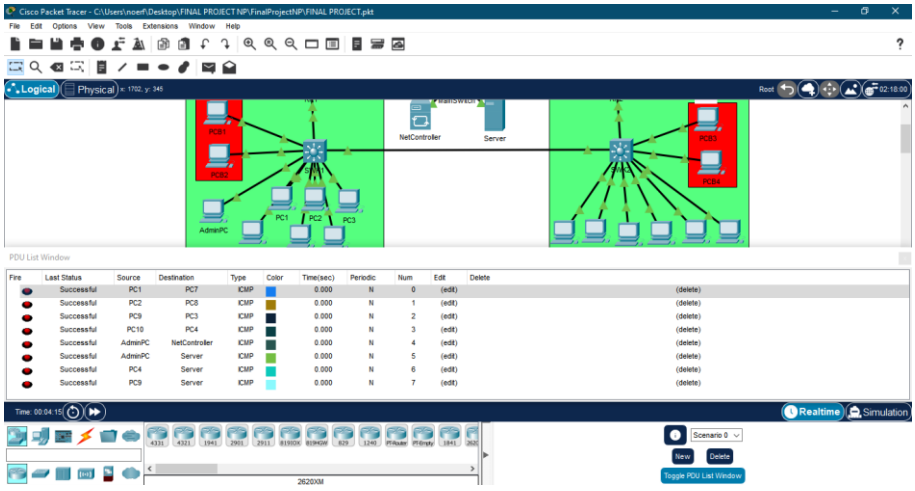
Konfigurasi awal dilakukan melalui terminal PC dengan metode CLI melalui console *switch*. Gambar dibawah ini menunjukan proses konfigurasi SWK2.



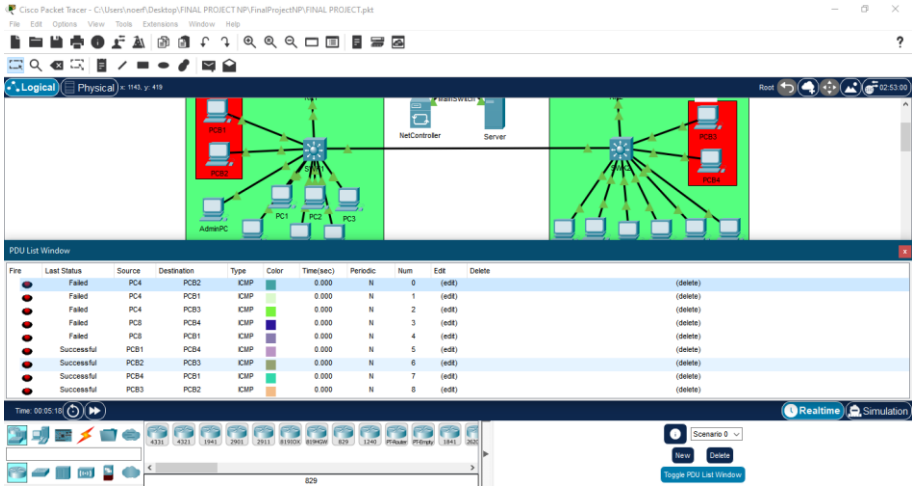


5. Tes Konektivitas Jaringan

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



Dan untuk gambar di bawah ini menunjukan, 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



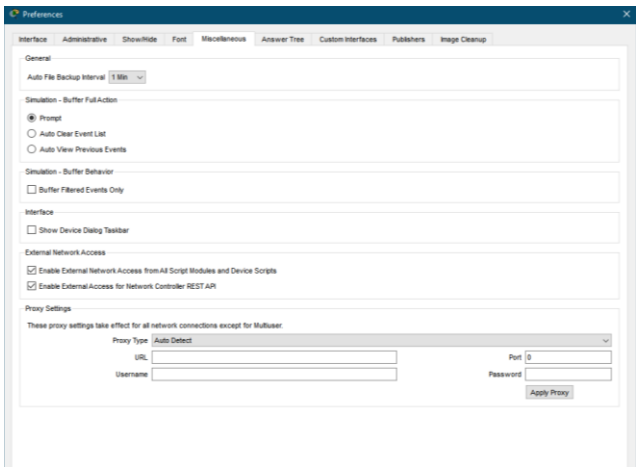
B. Implementasi Programabilitas Jaringan melalui SDN Controller

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).

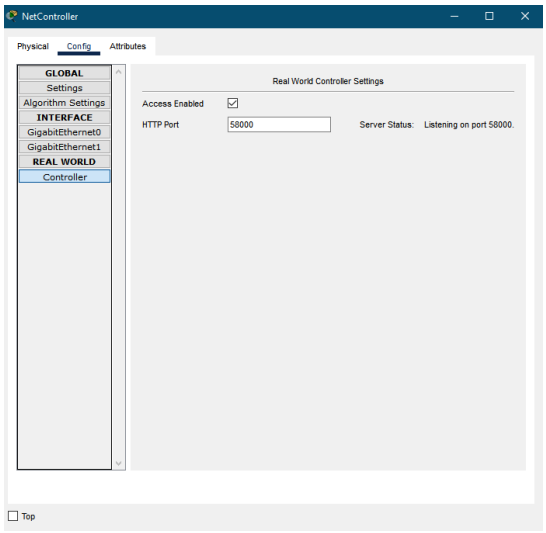
Adapun untuk cakupan automasi yang dilakukan dalam tugas akhir ini meliputi monitoring dan manajemen perangkat jaringan melalui *network controller* yang terhubung ke jaringan.

1. Konfigurasi *Network Controller*

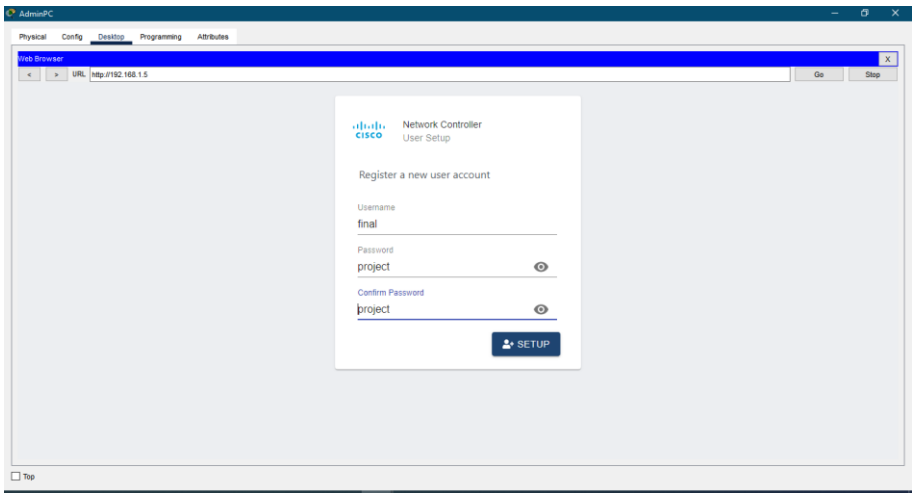
Untuk mengakses *network controller*, centang opsi pada External Network Access dari menu Options > Preferences > Miscellaneous.



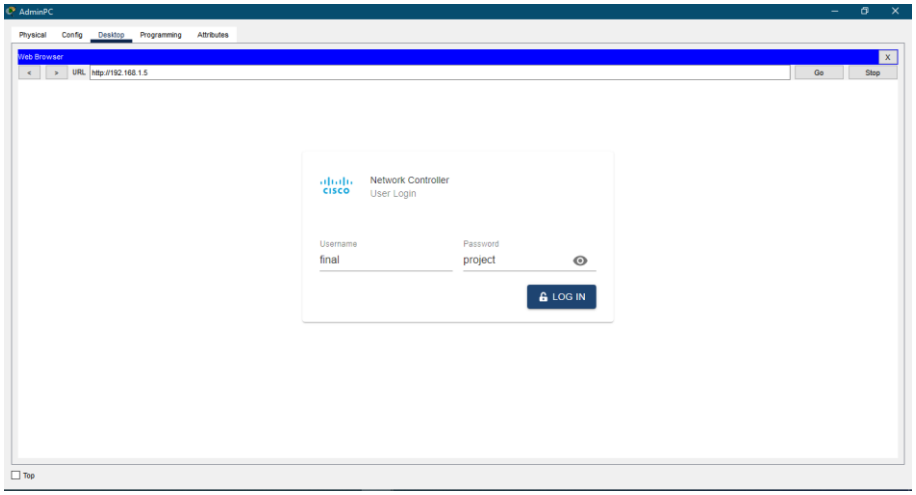
Dari perangkat *network controller*, aktifkan Access Enabled pada Config > REAL WORLD > Controller.



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-fiturya, *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*.

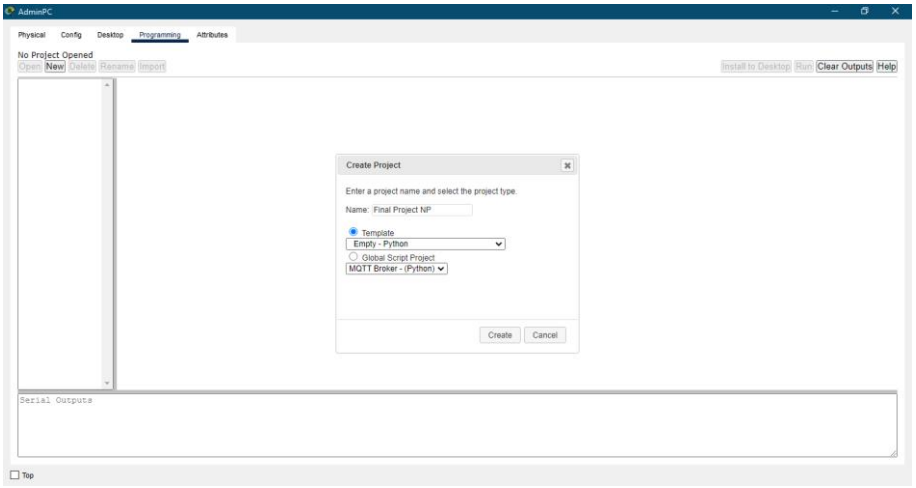
2. Fitur Automasi yang diimplementasikan

   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:

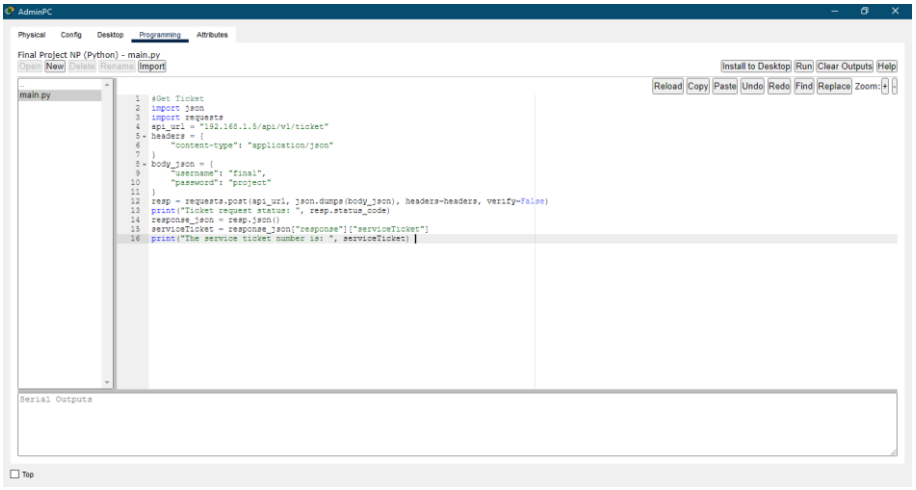
   a. Mendapatkan Service Ticket (Get Ticket)

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

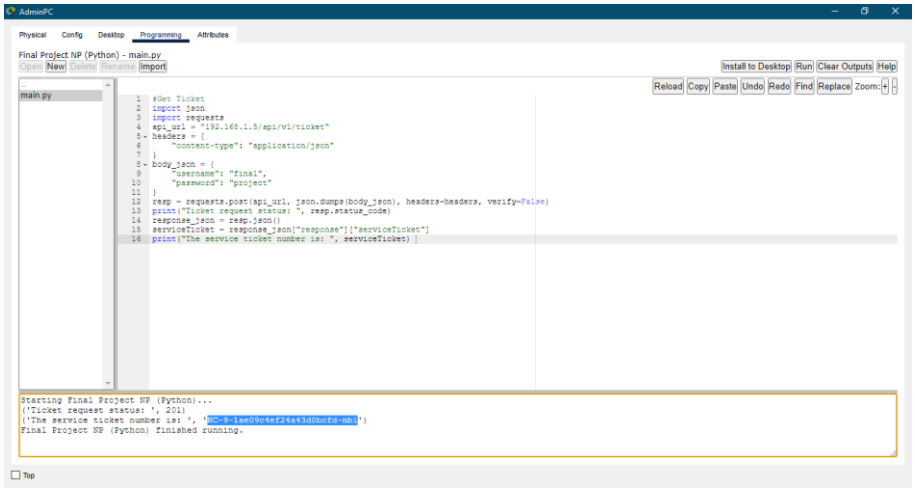
      Pada AdminPC, di bagian programming, buat program untuk mengakses dan meminta service ticket API *Network controller*.



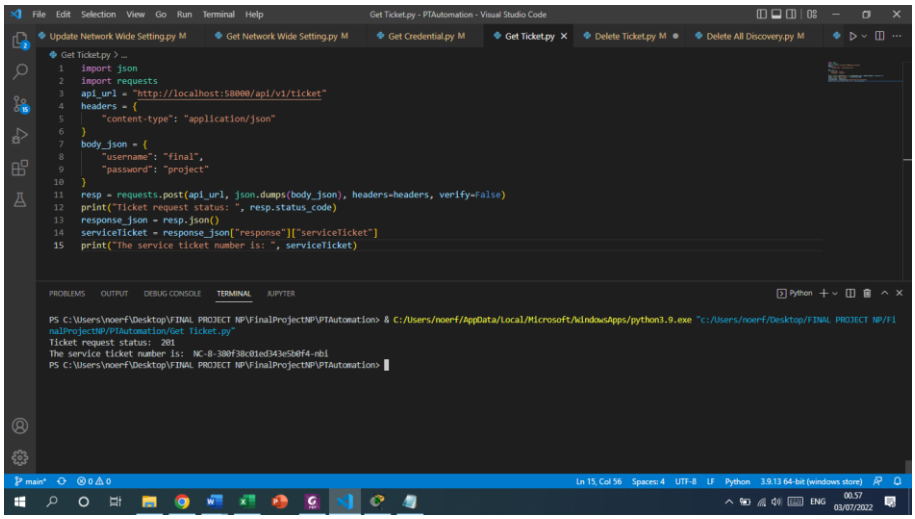
Pada programnya, isi api_url sesuai API documentation (lihat lampiran), dan pada body_json isi parameter username dan password *network controller* yang sudah dibuat sebelumnya, lalu jalankan program.



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



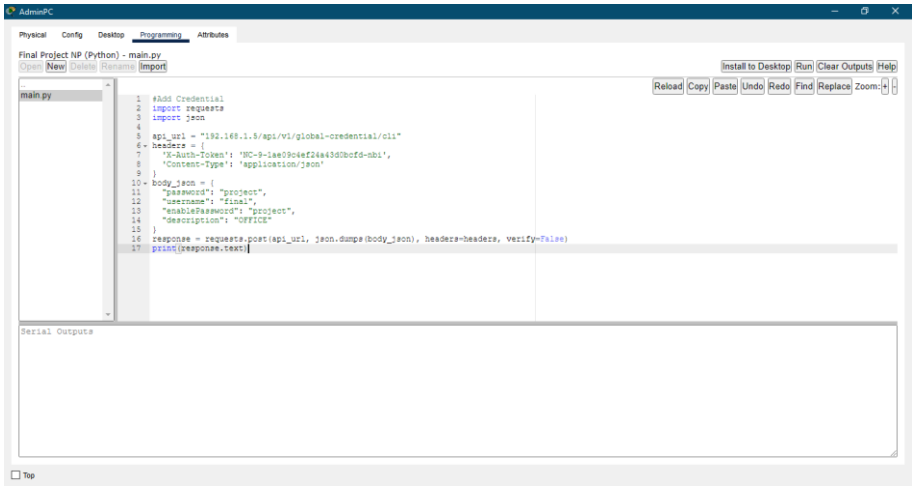
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



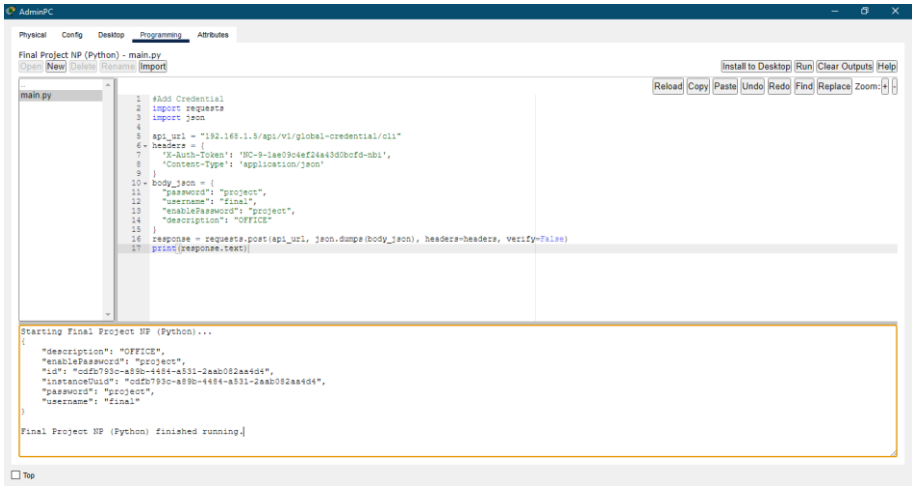
b. Menambahkan Kredensial (Add Credential)

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*.

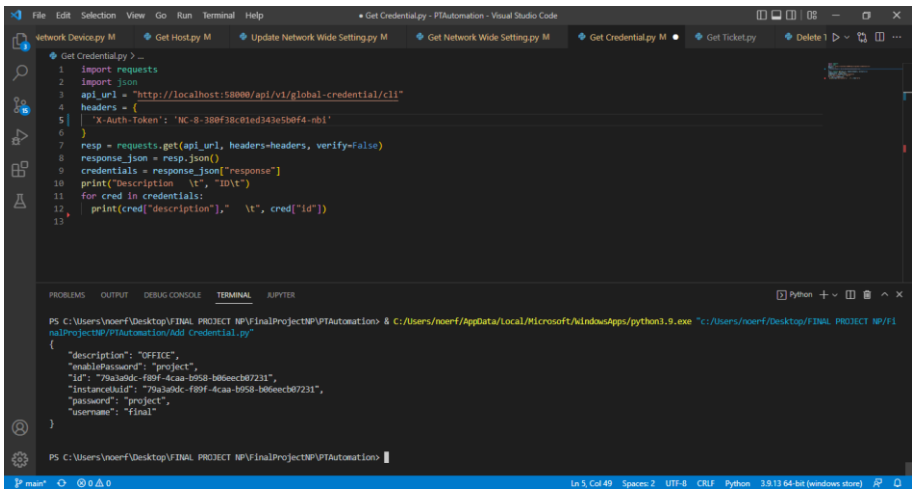
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Pada bagian body_json parameter yang harus diisi adalah username, password, enablePassword sesuai dengan akun SSH pada perangkat jaringan, dan isikan nama kredensialnya pada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



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

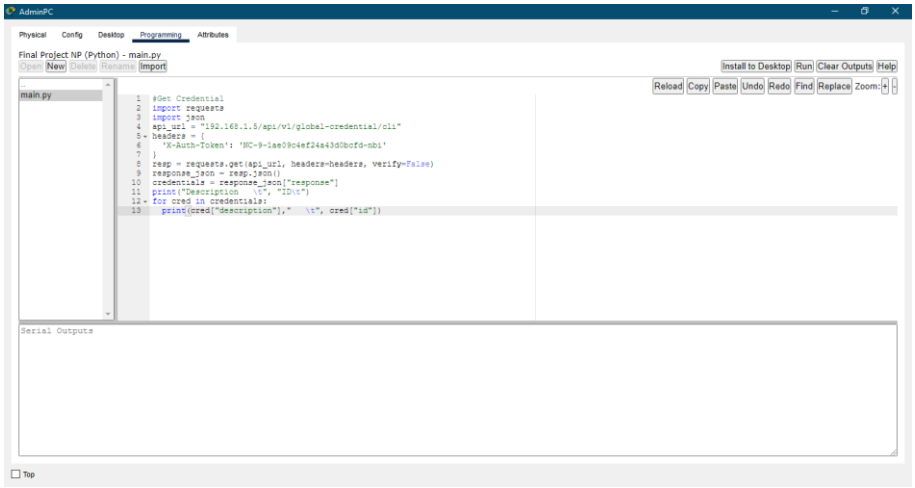


Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

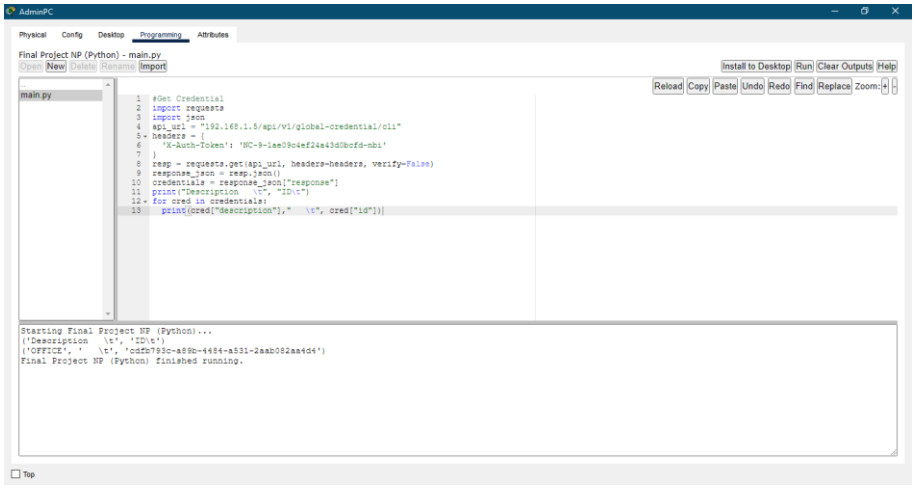


c. Mendapatkan Kredensial (Get Credential)

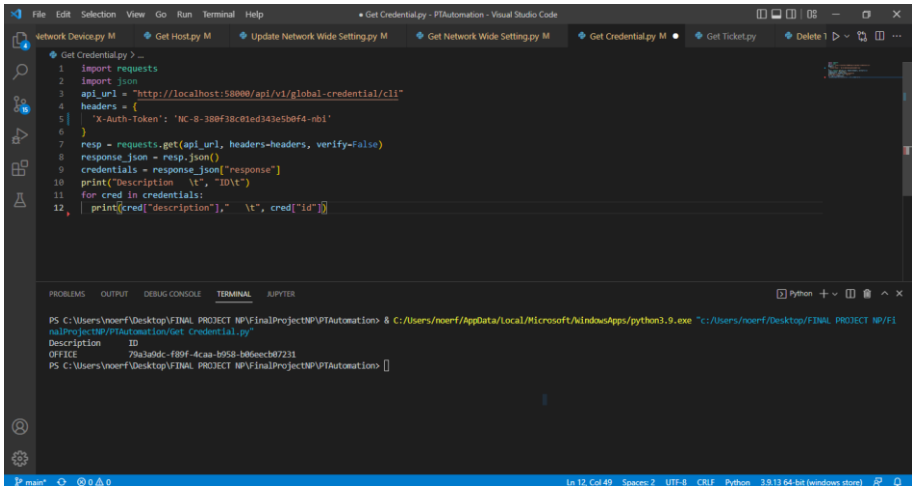
Apabila ingin mengetahui apa saja kredensial yang ada di *network controller*, bisa digunakan program get credential seperti pada gambar dibawah ini. Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket.



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



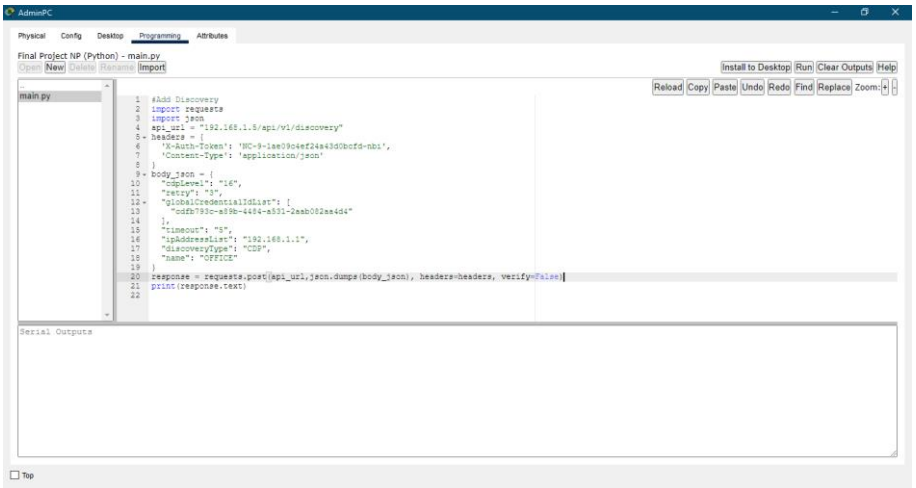
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



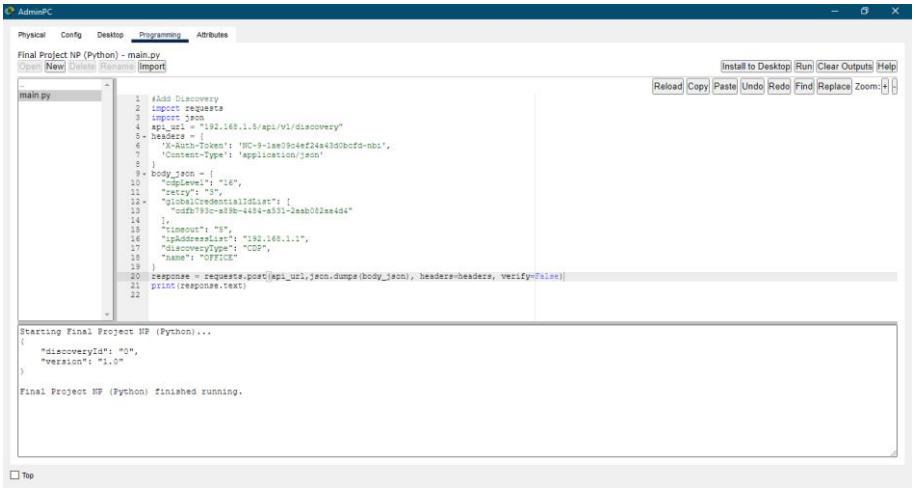
d. Menambahkan Discovery (Add Discovery)

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

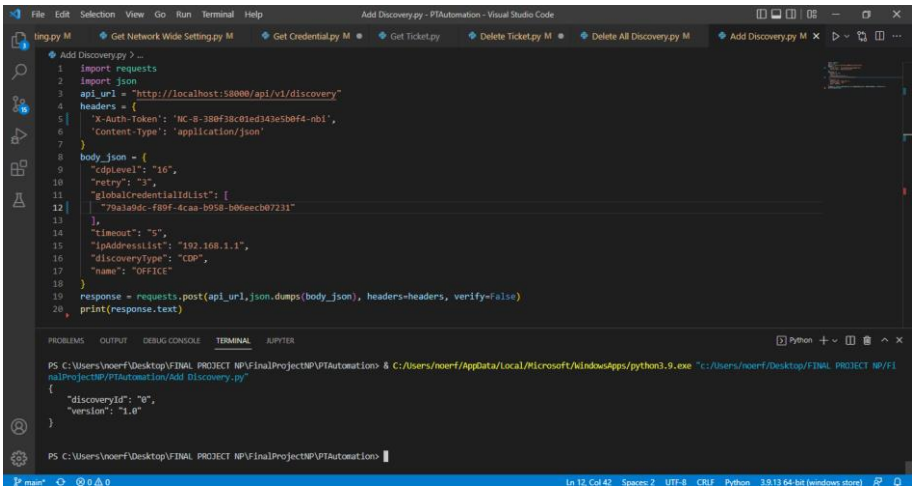
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Pada bagian body_json ada beberapa parameter yang harus diisi, yaitu cdpLevel, globalCredentialIdlist, timeout, retry, ipAddressList, discoveryType, dan nama discoverynya. Isi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



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

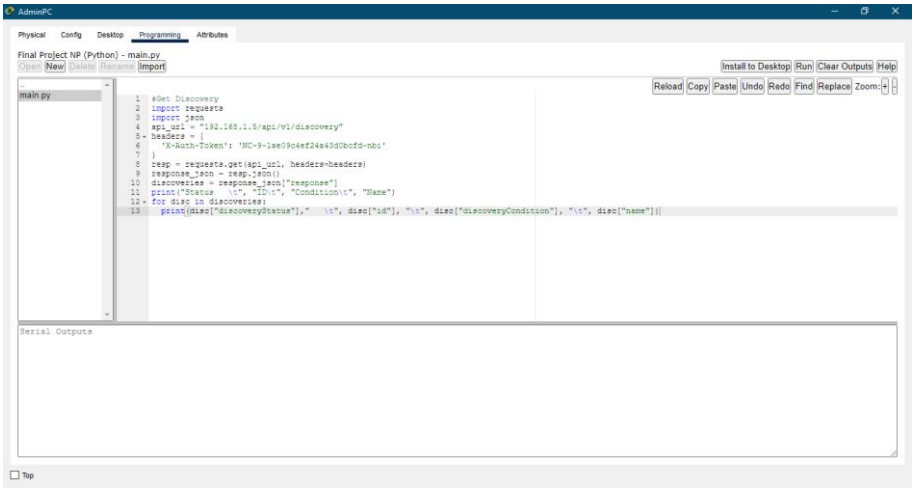


Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.

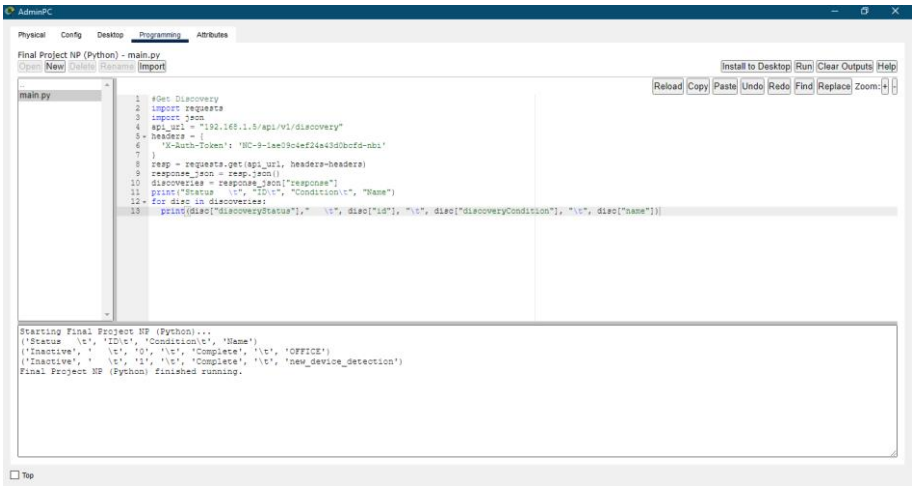


e. Mendapatkan Discovery (Get Discovery)

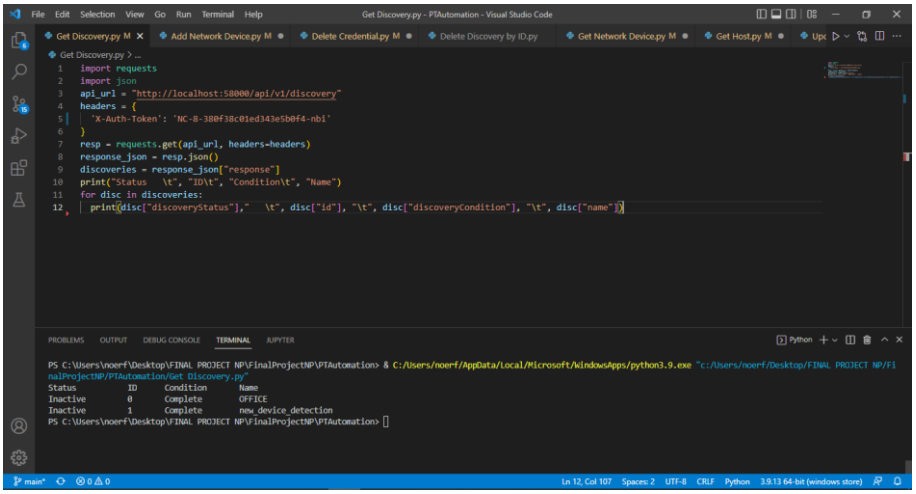
Untuk mengetahui discovery pada *network controller*, bisa digunakan program berikut. Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.



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



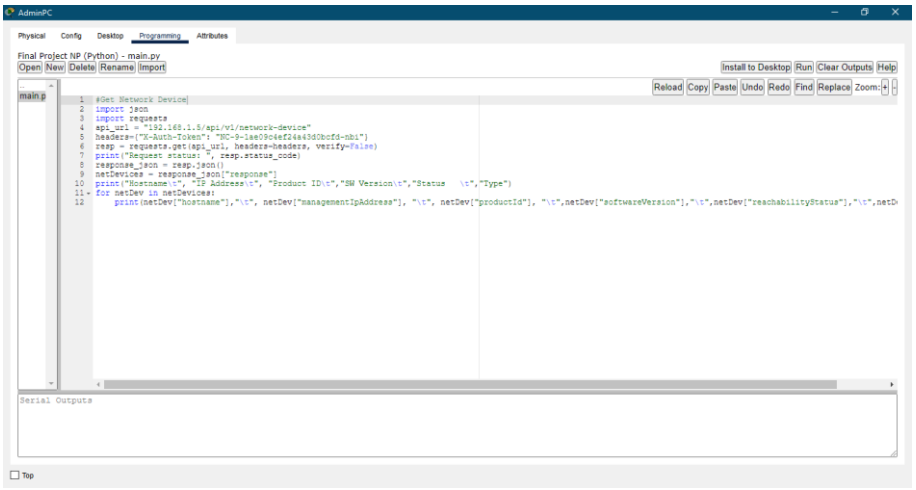
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



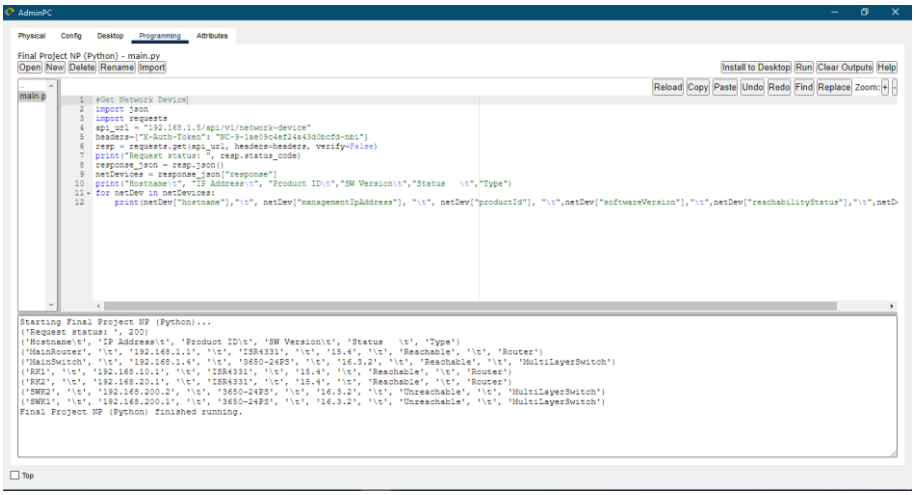
f. Mendapatkan Perangkat Jaringan (Get *Network Device*)

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

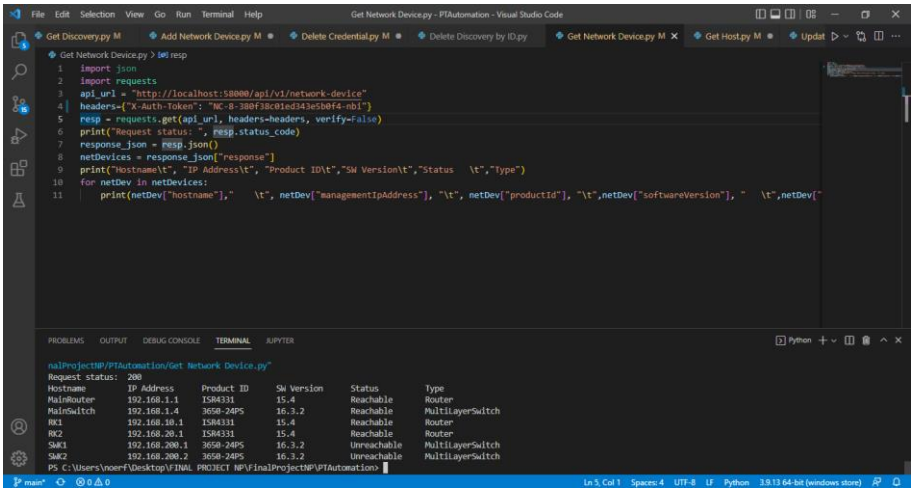
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.



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



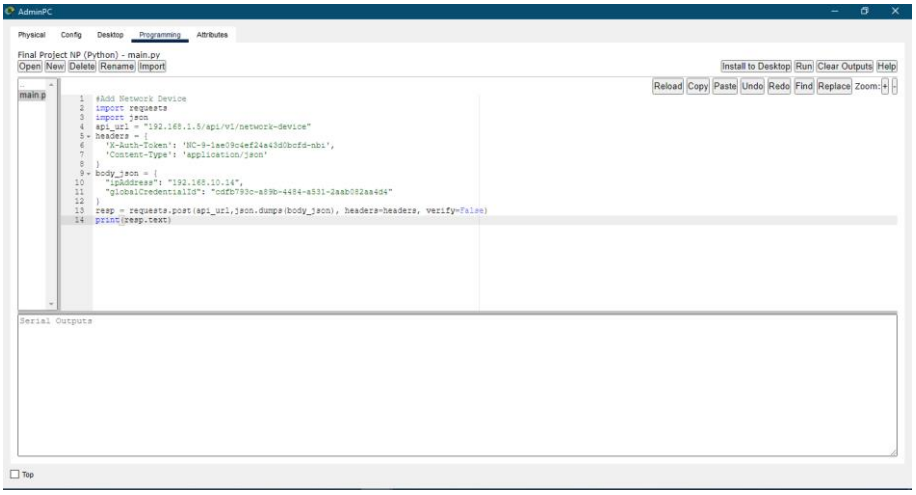
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



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

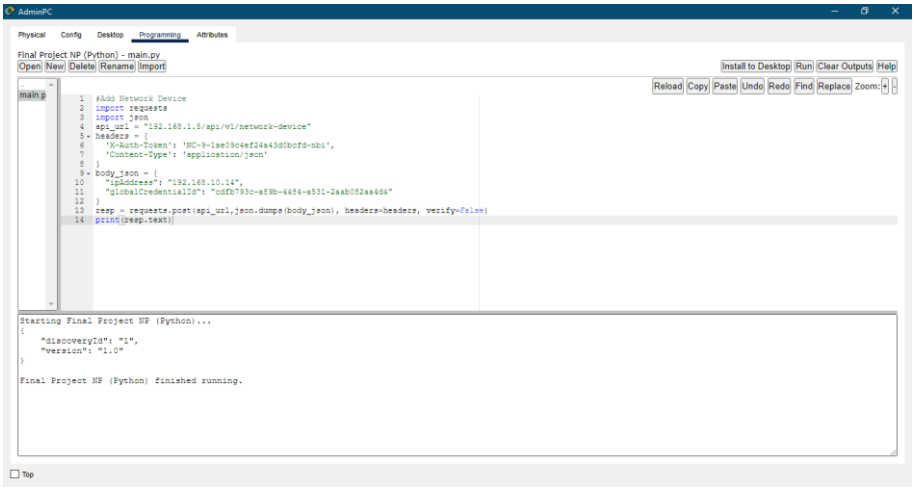
Apabila dirasa ada *network device* yang belum terpindai, atau tidak sesuai, pada kasus ini SWK1 dan SWK2 yang terpindai adalah IP VLAN200 nya, bukan IP VLAN 1 yang terhubung ke *router*nya. Maka kita bisa menambahkan *network device* secara manual.

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 global credential id nya dengan ID credential yang ingin kita gunakan. Lalu jalankan program.

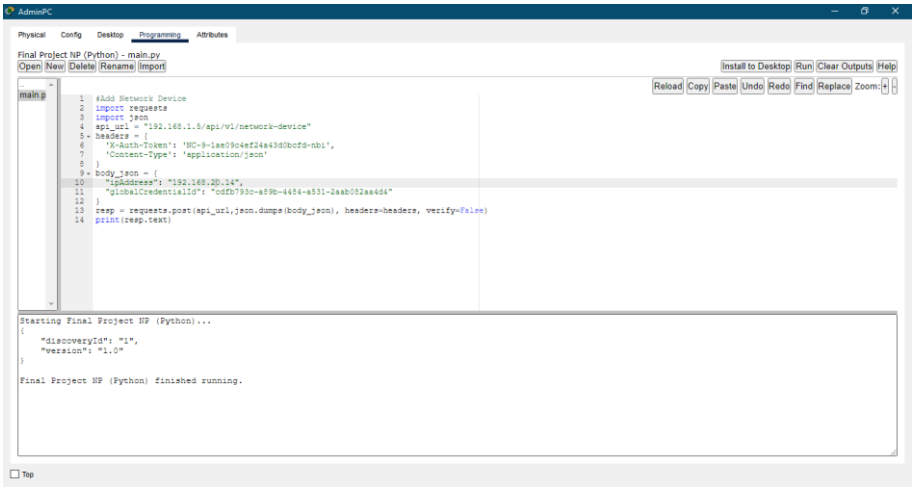


Output program akan menampilkan status discovery id apabila device berhasil ditambahkan.

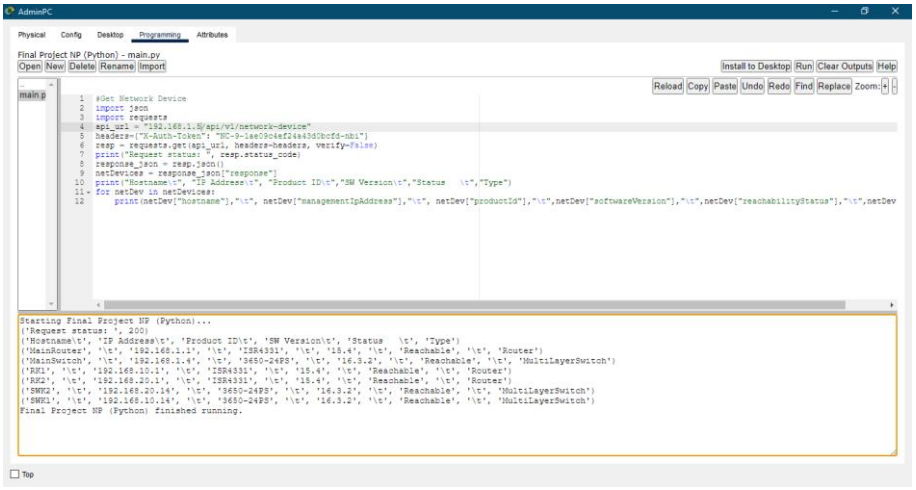
Untuk menambahkan SWK2, lakukan prosedur yang sama.



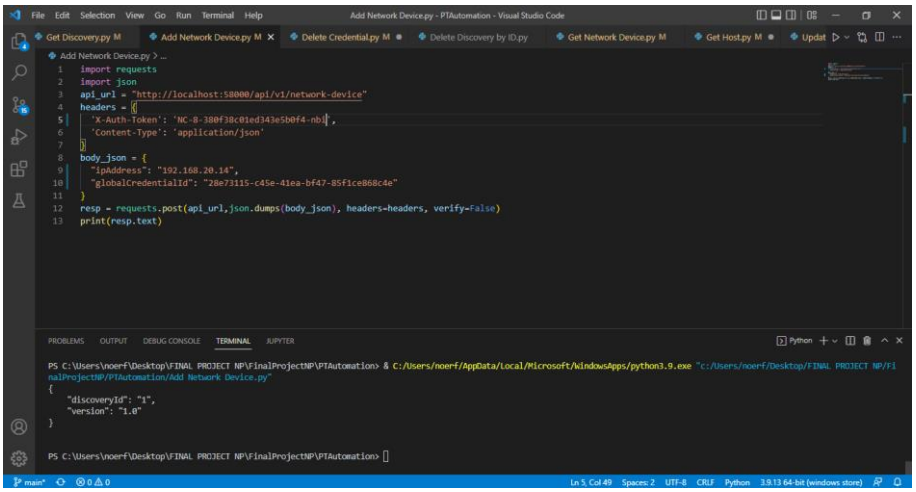
Output program akan menampilkan status discovery id apabila device berhasil ditambahkan.

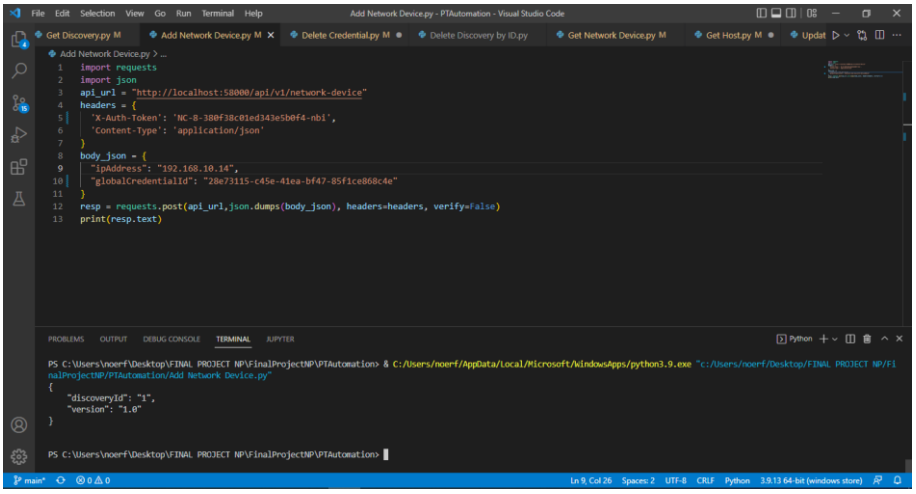


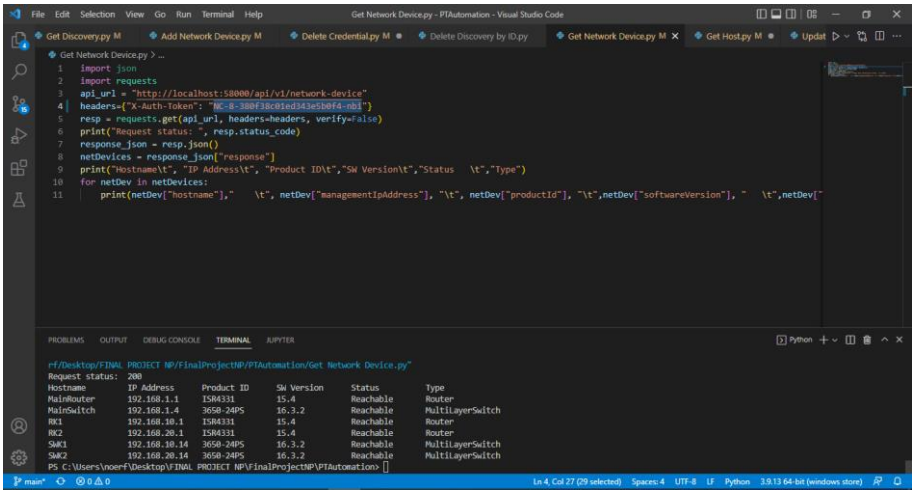
Untuk memvalidasi, bisa kita menjalankan program get *network device* sekali lagi



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



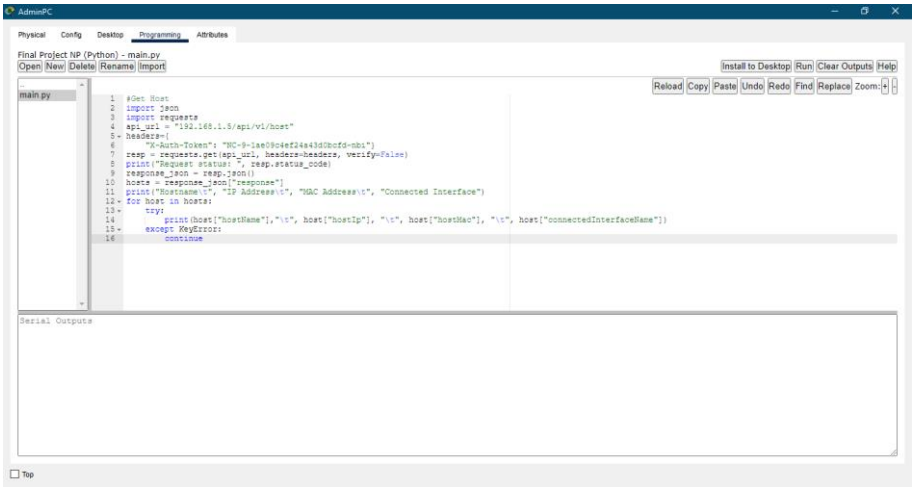




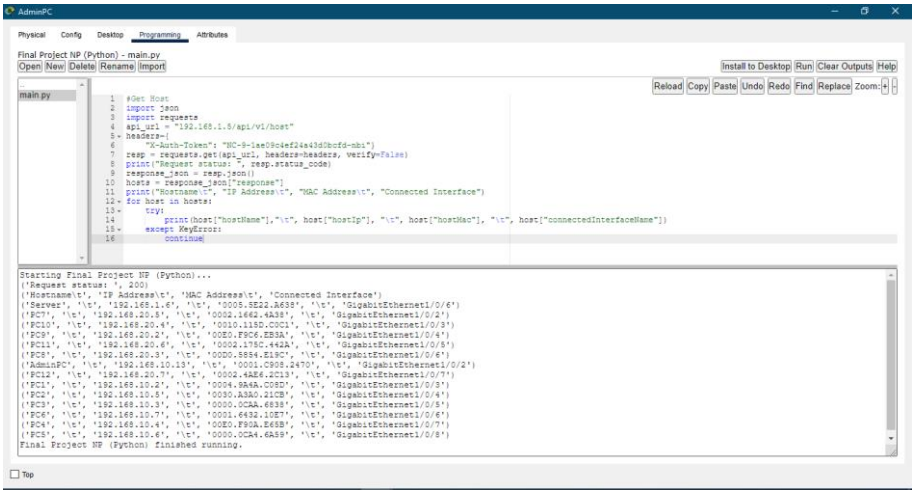
h. Mendapatkan *Host* (Get *Host*)

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

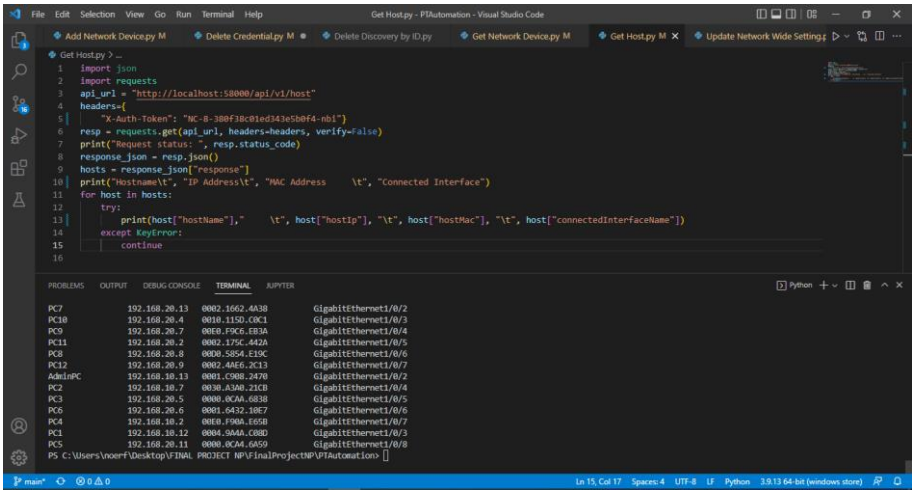
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.



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



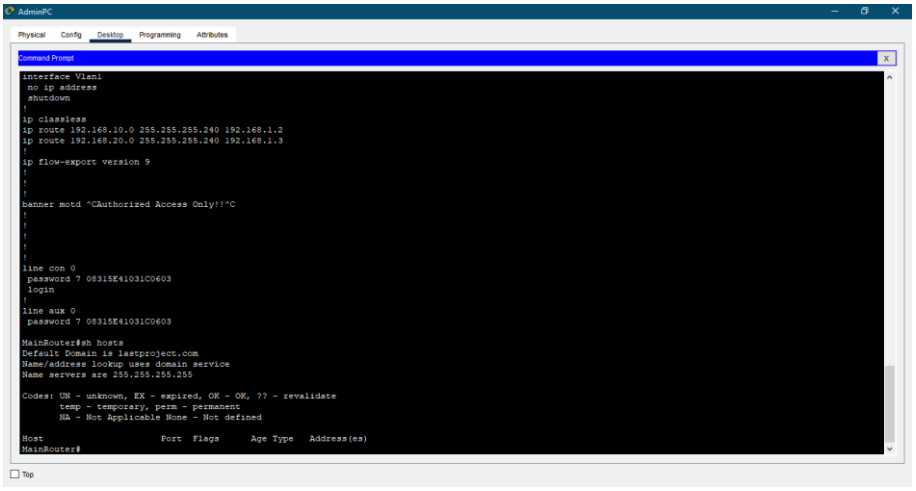
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



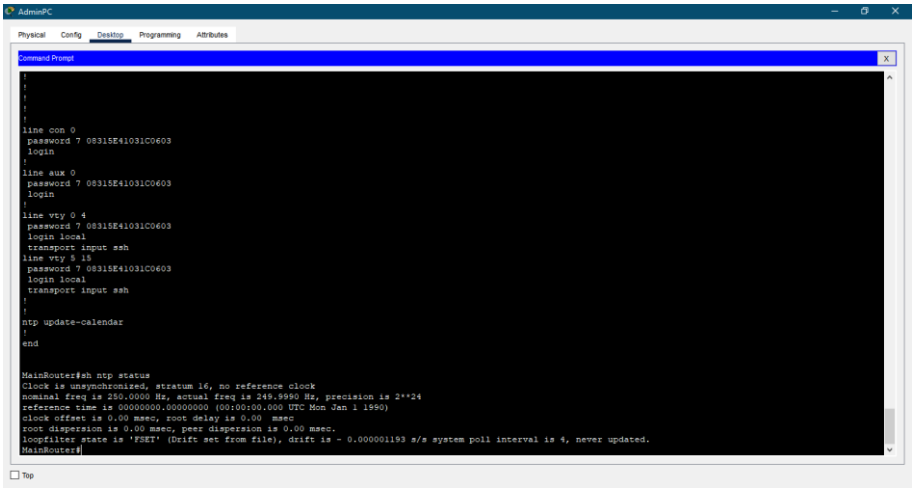
i. Memperbarui Wide Setting Jaringan (Update Network Wide Setting)

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.

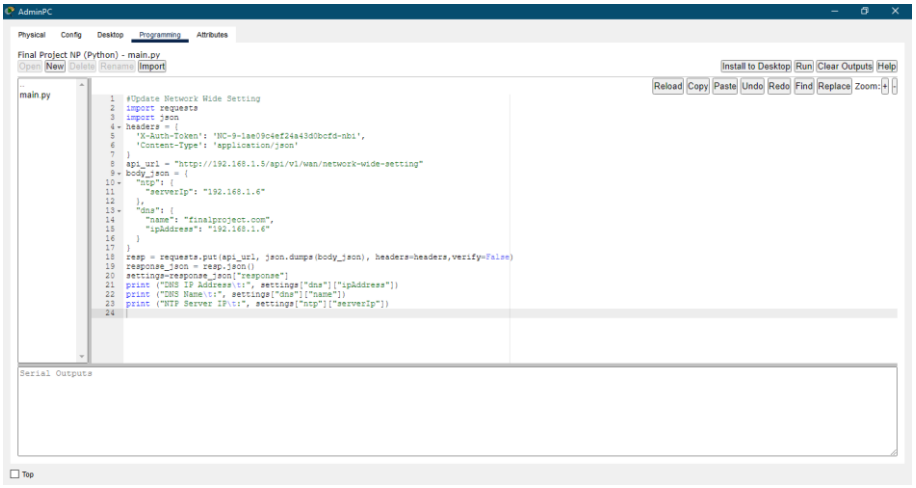
Sebelumnya, cek terlebih dahulu keadaan konfigurasi DNS dan NTPnya. Gambar dibawah ini menunjukan konfigurasi DNS pada *MainRouter* sebelum dilakukan update konfigurasi.



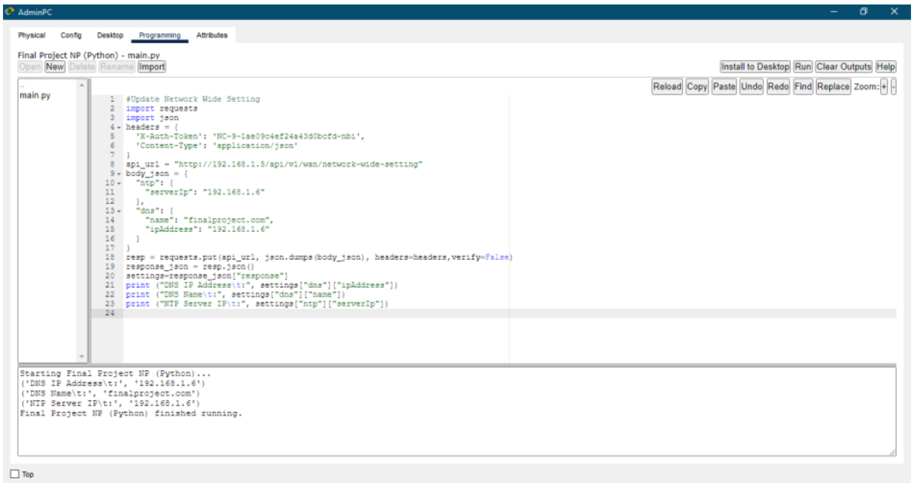
Gambar dibawah ini menunjukan konfigurasi NTP pada *MainRouter* sebelum dilakukan update konfigurasi.



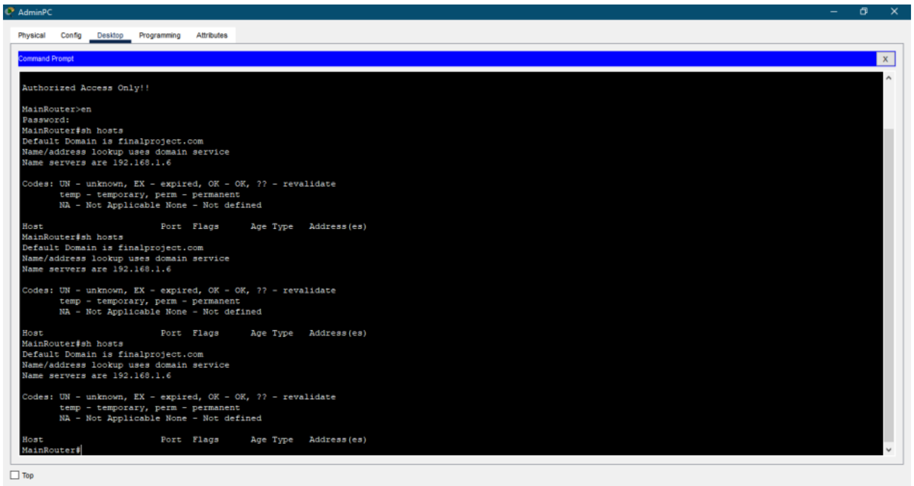
Untuk mengupdate konfigurasi DNS dan NTP pada semua *network device*, pada programnya isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Pada body_json, isi parameter IP *server* NTP, name address, dan IP Adress DNS. Lalu jalankan program.



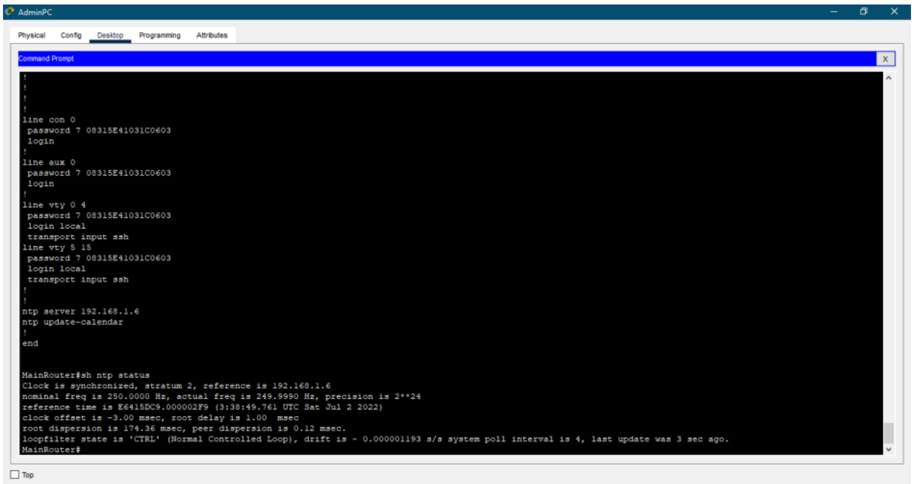
Output program akan menampilkan konfgurasi yang diupdate ke *network device*.



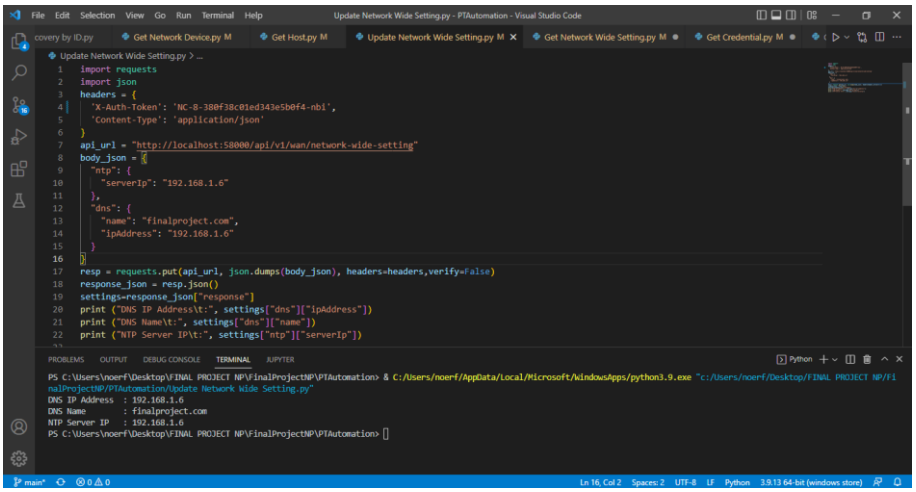
Bisa dicek kembali, DNS sudah terupdate di *network device*, pada gambar ini, yang terupdate adalah konfigurasi di *MainRouter*.



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



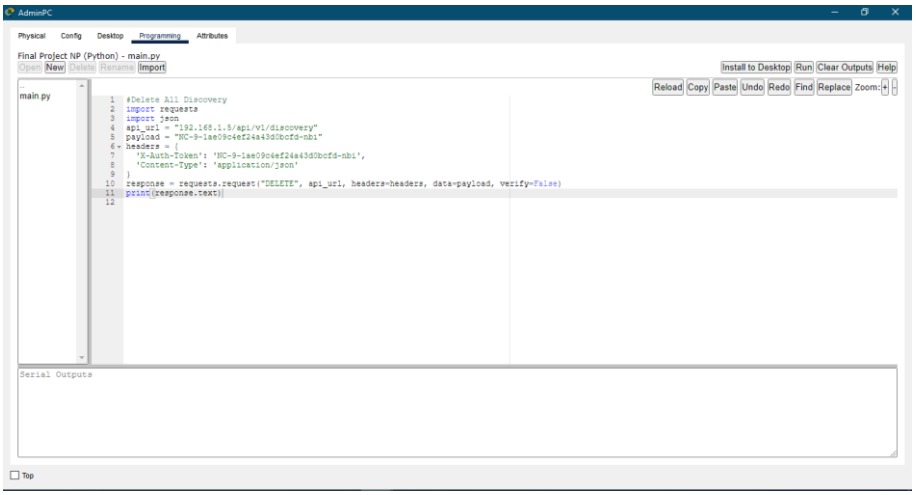
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



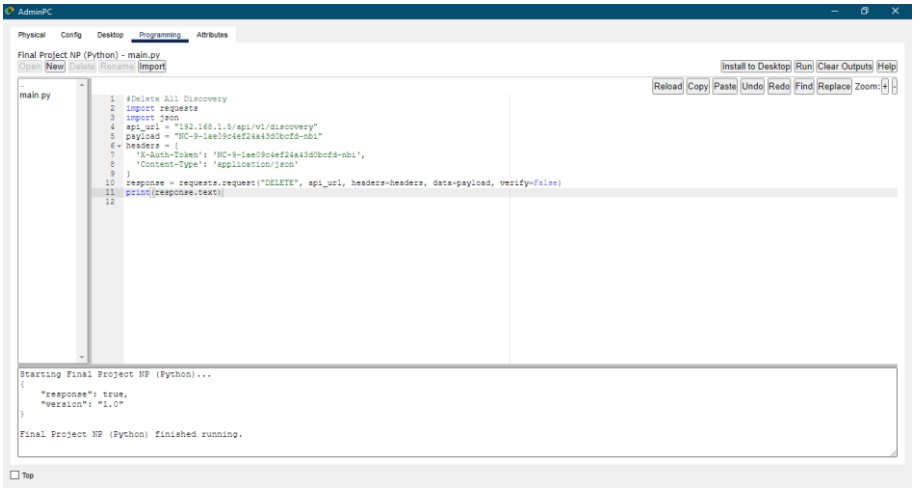
j. Menghapus Semua Discovery (Delete All Discovery)

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

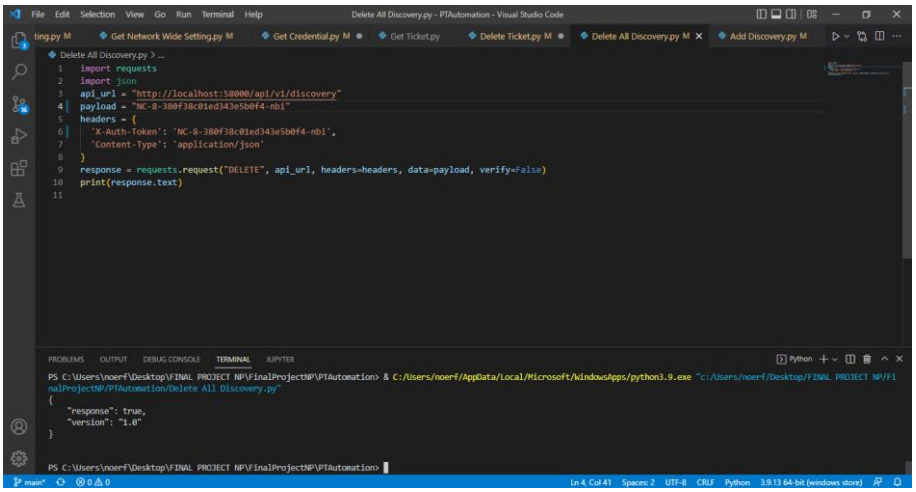
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket. Lalu jalankan program.



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



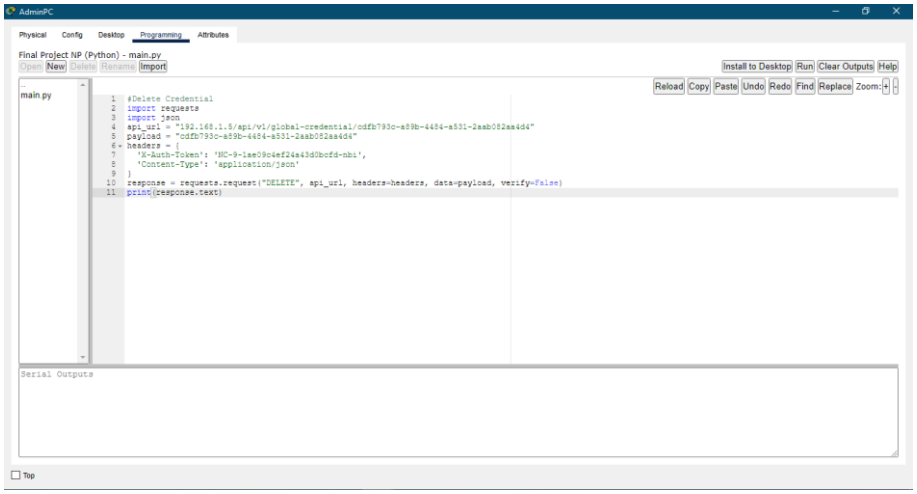
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



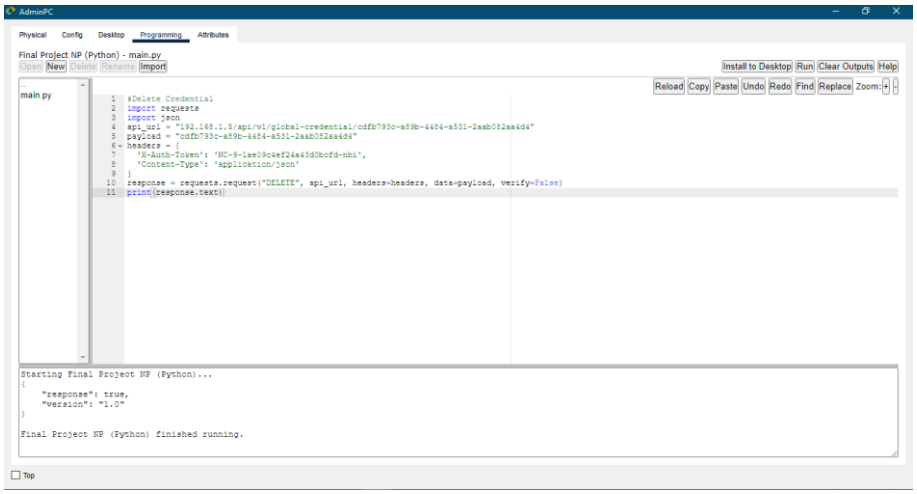
k. Menghapus Kredensial (Delete Credential)

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

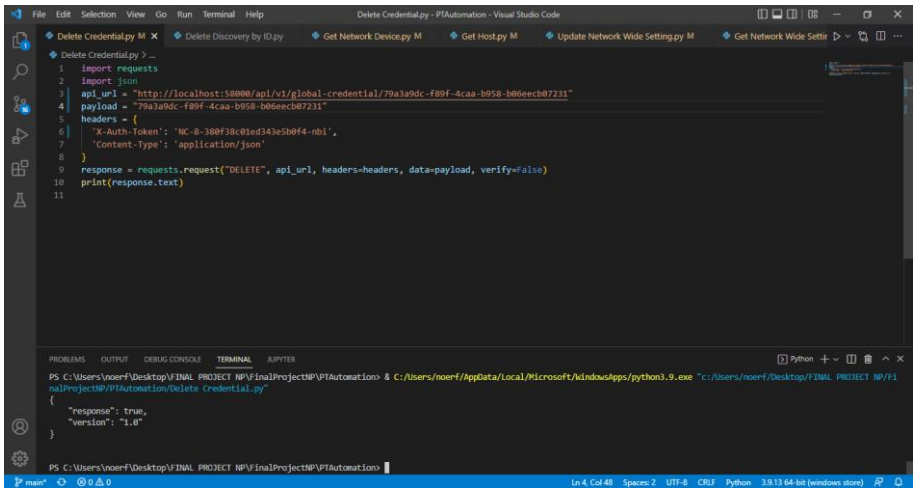
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), dan isi X-Auth-Token nya dengan service ticket, dan pada payload, isikan id credential yang ingin dihapus. Lalu jalankan program.



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



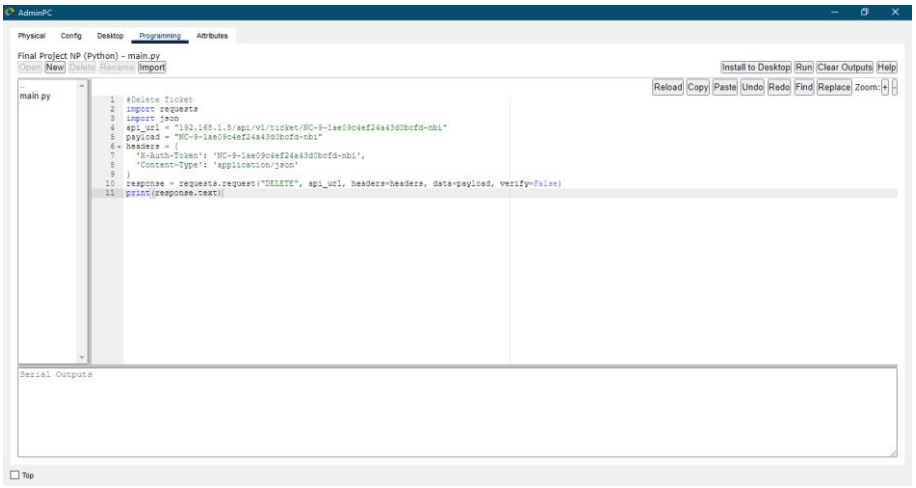
Program yang sama juga dijalankan melalui visual studio code, seperti pada gambar berikut.



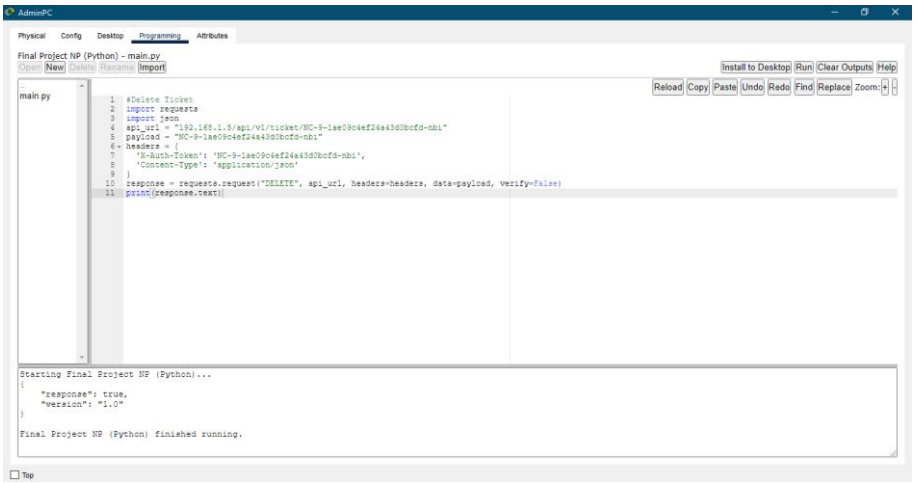
l. Menghapus Service Ticket (Delete Ticket)

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

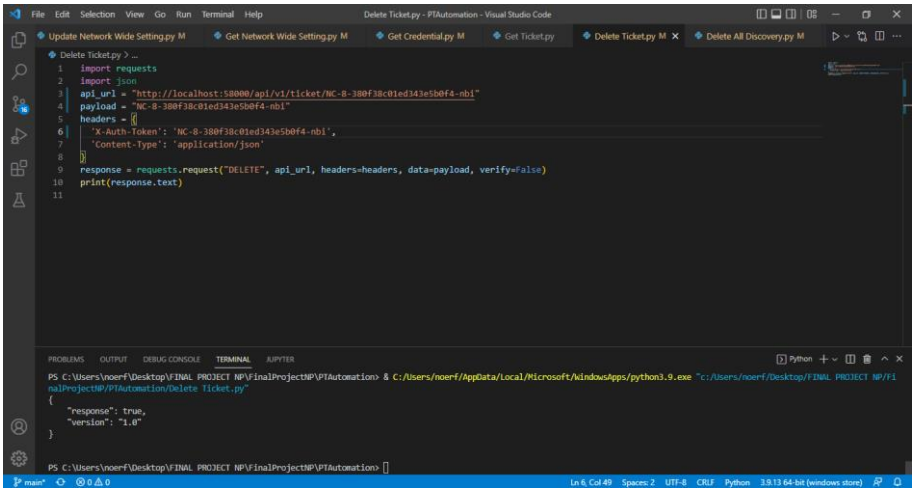
Pada programnya, isi api_url sesuai dengan API documentation (lihat lampiran), isi X-Auth-Token nya dengan service ticket, dan isi payload dengan nomor ticket yang akan dihapus. Lalu jalankan program.



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



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


File Edit Selection View Go Run Terminal Help
Delete 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
Delete Ticket.py M
Delete All Discovery.py M
Delete Ticket.py > ...
import requests
import json
api_url = "http://localhost:58000/api/v1/ticket/NC-8-380f38c01ed343e5b0f4-nbi"
payload = "NC-8-380f38c01ed343e5b0f4-nbi"
headers = {
'X-Auth-Token': 'NC-8-380f38c01ed343e5b0f4-nbi',
'Content-Type': 'application/json'
}
response = requests.request("DELETE", api_url, headers=headers, data=payload, verify=False)
print(response.text)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Python
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/Delete Ticket.py"
{
"response": true,
"version": "1.0"
}
PS C:\Users\noerf\Desktop\FINAL PROJECT NP\FinalProjectNP\PTAutomation>
main*
Ln 6, Col 49
Spaces: 2
UTF-8
CRLF
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 meggunakan 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*

Nama Program	**api_url**
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 *Network Device*	http://(netcontrollerIP)/api/v1/network-device
Add Nework 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)