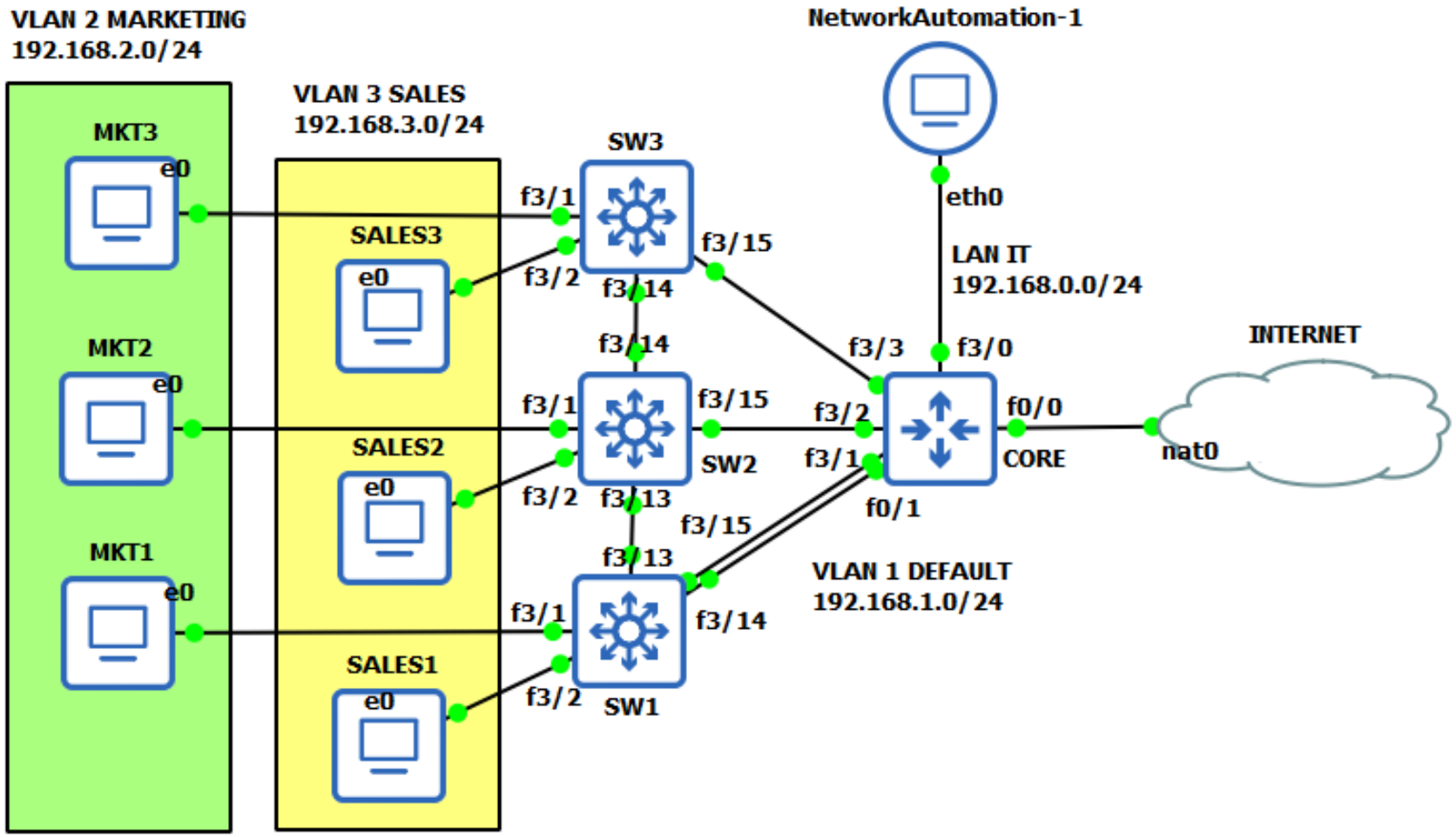
PYTHON NETWORK PROGRAMMING FOR NETWORK ENGINEERS

I PUTU HARIYADI (PUTU.HARIYADI@UNIVERSITASBUMIGORA.AC.ID)

FAKULTAS TEKNIK DAN DESAIN
PROGRAM STUDI TEKNOLOGI INFORMASI
UNIVERSITAS BUMIGORA

RANCANGAN JARINGAN UJICOBA



- Rancangan jaringan ujicoba disimulasikan menggunakan GNS3.
- **Router** dan **Multilayer Switch** menggunakan **Cisco IOS image c3745-adventerprisek9-mz.124-25d**.
- **Tool** pemrograman berbasis jaringan menggunakan **container GNS3 Appliance Network Automation**.
- **PC Client** menggunakan **Virtual PC Simulator (VPCS)**.

RANCANGAN PENGALAMATAN IP JARINGAN UJICOBA

PERANGKAT	HOSTNAME	INTERFACE	IP ADDRESS
Cisco Router CORE	CORE	Vlan 1	192.168.0.1/24
Cisco Multilayer Switch SW1	SW1	Vlan 1	192.168.0.11/24
Cisco Multilayer Switch SW2	SW2	Vlan 1	192.168.0.12/24
Cisco Multilayer Switch SW3	SW3	Vlan 1	192.168.0.13/24
Network Automation		eth0	192.168.0.2/24
VPCS	MKT1, MKT2, MKT3, SALES1, SALES2, SALES3		DHCP Client

RANCANGAN VLAN, PORT MEMBERSHIP DAN NETWORK ADDRESS SERTA INTERFACE TRUNK

- **VLAN Database** dan **Port membership** untuk setiap **VLAN** di masing-masing **Cisco Multilayer Switch** serta alokasi **network address per VLAN**.

VLAN ID	NAME	PORT MEMBERSHIP	NETWORK ADDRESS
2	MARKETING	FastEthernet3/1	192.168.2.0/24
3	SALES	FastEthernet3/2	192.168.3.0/24

- **Interface Trunk** di masing-masing **Cisco Multilayer Switch**.

INTERFACE	MODE
FastEthernet3/13	trunk
FastEthernet3/14	trunk

RANCANGAN INTERVLAN ROUTING DAN DHCP SERVER

- **Interface FastEthernet0/1** pada **Cisco Router CORE** di **subinterface** untuk **InterVLAN routing**.

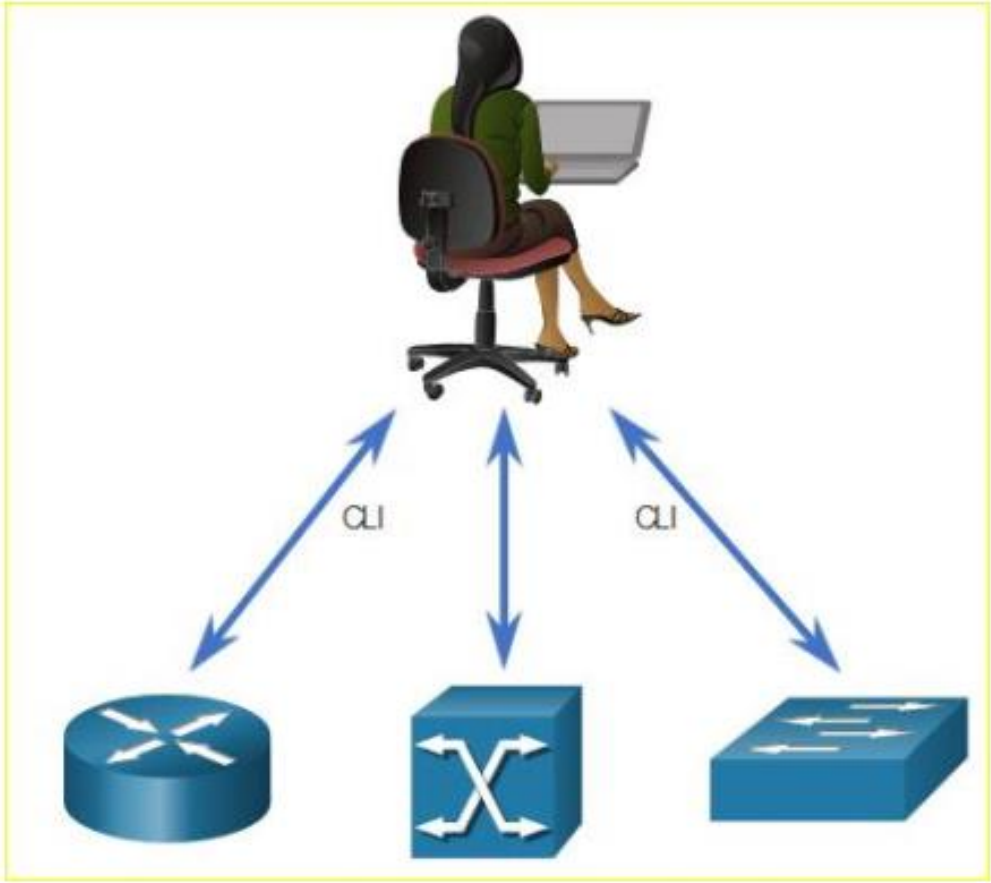
SUBINTERFACE	ENCAPSULATION	IP ADDRESS	DESCRIPTION
FastEthernet0/1.2	dot1q 2	192.168.2.1/24	trunk untuk VLAN 2
FastEthernet0/1.3	dot1q 3	192.168.3.1/24	trunk untuk VLAN 3

- Pengaturan **DHCP Server** pada **Cisco Router CORE**.

POOL NAME	NETWORK ADDRESS	DEFAULT ROUTER	DNS SERVER	EXCLUDED ADDRESS
MARKETING	192.168.2.0/24	192.168.2.1	192.168.2.1	192.168.2.1
SALES	192.168.3.0/24	192.168.3.1	192.168.3.1	192.168.3.1

KONFIGURASI JARINGAN SECARA TRADISIONAL

- Perangkat jaringan seperti **router**, **switch** dan **firewall** secara tradisional telah dikonfigurasi oleh administrator jaringan menggunakan **Command Line Interface (CLI).**
- Ketika terjadi perubahan atau fitur baru, perintah konfigurasi yang diperlukan harus dimasukkan pada semua perangkat yang sesuai.
- Pada banyak kasus, hal ini tidak hanya memakan waktu tetapi dapat juga rentan terhadap kesalahan.
- Ini menjadi masalah besar pada jaringan yang lebih besar atau dengan konfigurasi yang lebih kompleks

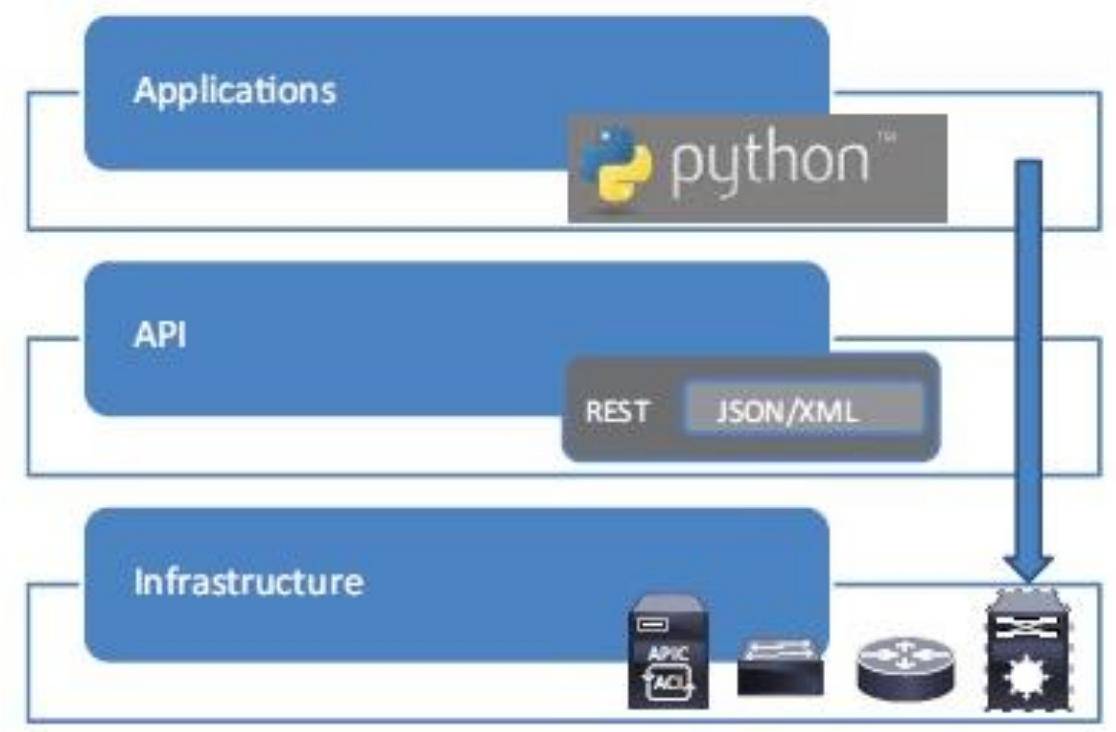


Sumber: Cisco Networking Academy ENSA v.7.0

NETWORK PROGRAMMABILITY

- Menurut **Ryan Tischer** dan **Jason Gooley**, **Network programmability** adalah seperangkat **tool** yang digunakan untuk menyebarkan, mengelola dan memecahkan permasalahan (**troubleshoot**) pada perangkat jaringan.
- Bagi **network engineer**, **programmability** berarti berinteraksi dengan perangkat atau sekelompok perangkat yang menggerakkan konfigurasi, pemecahan masalah (*troubleshooting*) dengan perangkat lunak yang berada secara logikal pada perangkat.
- Keuntungan **Network Programmability**, meliputi:

1. Menghemat waktu dan biaya.
2. Mengurangi **human error**.
3. Inovasi.



Sumber: networkcomputing.com

NETWORK AUTOMATION

- **Otomatisasi** adalah segala proses yang mendorong sesuatu bekerja dengan sendirinya sehingga mengurangi dan berpotensi menghilangkan kebutuhan intervensi manusia.
- **Network Automation** merupakan proses untuk mengotomatisasi konfigurasi, pengelolaan, pengujian, penerapan dan pengoperasian perangkat fisik dan virtual dalam jaringan.
- Ketersediaan layanan jaringan akan meningkat apabila tugas dan fungsi jaringan sehari-hari prosesnya dikelola dan dikontrol ulang secara otomatis.
- Keuntungan otomatisasi, antara lain:

1. Mesin dapat bekerja 24 jam sehari tanpa henti sehingga memberikan hasil yang lebih tinggi.
2. Mesin dapat menghasilkan produk yang seragam.
3. Otomatisasi memberikan kumpulan data yang dapat dengan cepat dianalisa sehingga menyediakan informasi yang dapat membantu suatu kejadian atau proses.

Sumber: Cisco Networking Academy ENSA v.7.0

PYTHON NETWORK AUTOMATION LIBRARY

- **Python** merupakan bahasa pemrograman yang dibuat oleh **Guido van Rossum** dan bersifat **free** serta dapat berjalan pada berbagai platform, seperti **Windows**, **Mac**, **Linux**, dan lain-lain.
- Menurut w3schools.com, **Python** dapat digunakan untuk mengembangkan aplikasi **web (server side)**, pengembangan perangkat lunak yang terhubung ke sistem **database**, dan **system scripting** serta menangani **big data**.
- Python juga dapat digunakan untuk melakukan **Network Automation**.
- Terdapat berbagai *library* **Python** terkait **network automation**, diantaranya adalah Paramiko, NAPALM (*Network Automation and Programmability Abstraction Layer with Multivendor support*), Netmiko.
- Menurut situs Paramiko.org, **Paramiko** adalah implementasi **Python (2.7, 3.4+)** dari protokol **SSHv2** yang menyediakan fungsionalitas *client* dan *server*.
- Menurut situs Napalm.readthedocs.io, **NAPALM** merupakan *library* **Python** yang mengimplementasikan serangkaian fungsi untuk berinteraksi dengan sistem operasi perangkat jaringan yang berbeda menggunakan **Application Programming Interface (API)** terpadu. Mendukung beberapa metode untuk terhubung ke perangkat, manipulasi konfigurasi atau mengambil data.

PYTHON LIBRARY NETMIKO

- Merupakan **Python Library** yang menyederhanakan manajemen koneksi **Secure Shell (SSH)** ke **Command Line Interface (CLI)** dari perangkat jaringan dan mendukung beragam vendor serta platform jaringan.
- Dibuat berdasarkan Paramiko SSH Library.
- **Netmiko** berfokus pada penyederhanaan eksekusi dari **show commands** dan pengambilan data **output**.
- **Netmiko** menyederhanakan eksekusi perubahan konfigurasi (**configuration changes**) termasuk tindakan **commit** dengan mengabstraksi **low-level state control** sehingga menghilangkan pola pencocokan **low-level Regular Expression (Regex)**.
- **Netmiko** dapat diunduh melalui https://github.com/ktbyers/netmiko/releases

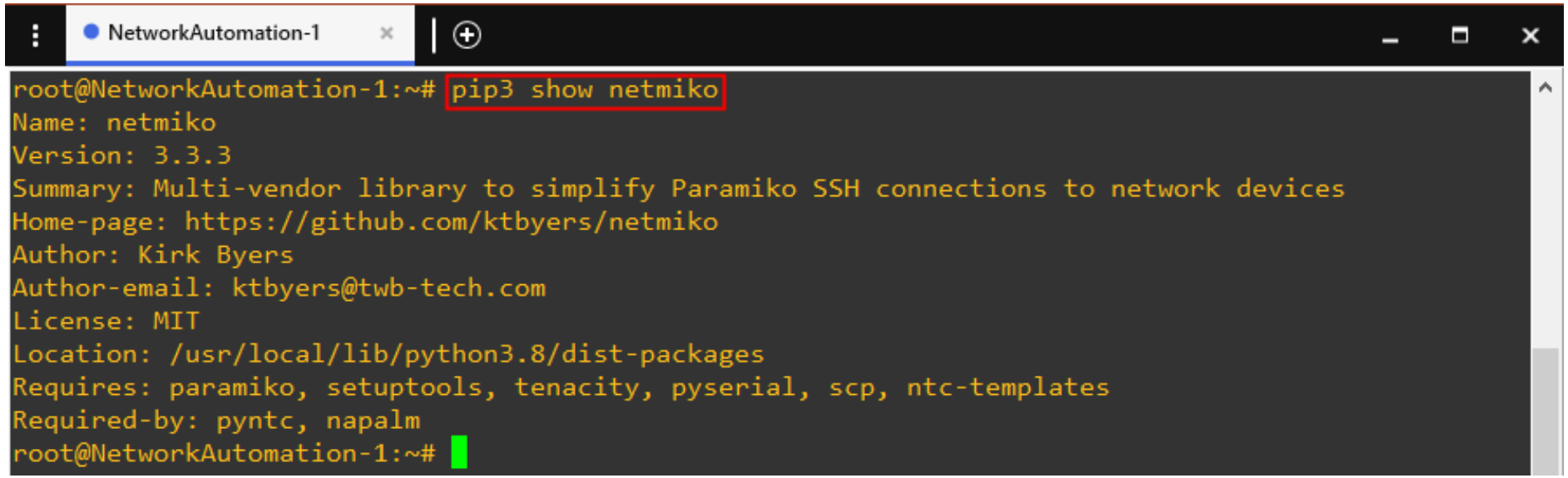
INSTALASI NETMIKO

- Dapat dilakukan melalui manajemen paket *Python* **PIP** dengan mengeksekusi perintah:

```
pip3 install netmiko
```

- Menampilkan informasi detail terkait paket **Netmiko** yang telah terinstalasi pada **Network Automation GNS3 Appliance** dengan mengeksekusi perintah:

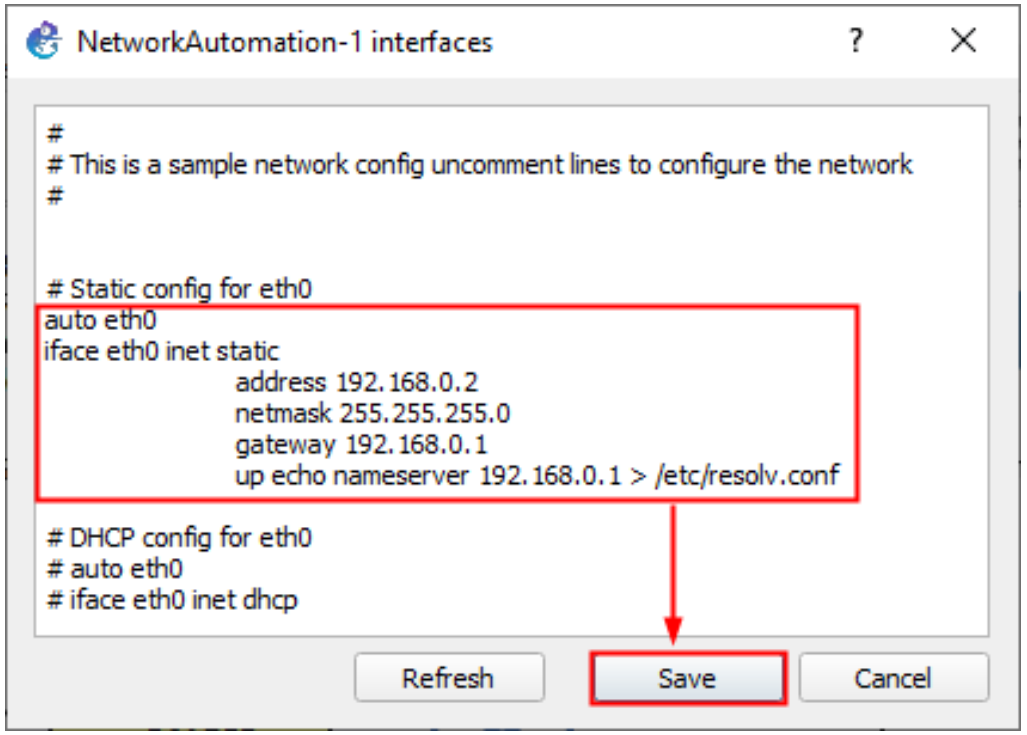
```
pip3 show netmiko
```



DEMO NETWORK PROGRAMMABILITY

KONFIGURASI DASAR DI GNS3 NETWORK AUTOMATION APPLIANCE

- Mengatur pengalamatan IP secara static dengan cara klik kanan pada perangkat **NetworkAutomation-1** di **GNS3** dan pilih **Edit config**.
- Lakukan penyesuaian dengan menghapus tanda **#** disetiap awal baris setelah baris **#Static config for eth0**.
- Mengubah nilai dari beberapa parameter, seperti **IP** dari **interface eth0** agar menggunakan alamat **192.168.0.2** dengan netmask 255.255.255.0 dan **default gateway** serta **nameserver** menggunakan **192.168.0.1**, seperti terlihat pada gambar yang ditandai dengan kotak bergambar merah.
- Klik tombol **Save** untuk menyimpan perubahan.
- Untuk menjalankan perangkat **NetworkAutomation-1**, lakukan klik kanan pada perangkat tersebut dan pilih **Start**.



KONFIGURASI DASAR PADA ROUTER CORE (1)

- Berpindah ke mode *global configuration*

```
Router# conf t
```

- Mengatur *hostname*

```
Router(config)# hostname CORE
```

- Berpindah ke *interface vlan 1*

```
CORE(config)# int vlan 1
```

- Mengatur pengalamatan IP pada *interface vlan 1*

```
CORE(config-if)# ip address 192.168.0.1 255.255.255.0
```

- Mengaktifkan *interface*

```
CORE(config-if)# no shutdown
```

KONFIGURASI DASAR PADA ROUTER CORE (2)

- Berpindah ke satu mode sebelumnya

```
CORE(config-if)# exit
```

- Mengatur nama domain yang digunakan oleh perangkat

```
CORE(config)# ip domain-name ubg.local
```

- Mengatur agar perangkat menggunakan SSH versi 2

```
CORE(config)# ip ssh version 2
```

- Menghasilkan RSA Key dengan modulus 1024

```
CORE(config)# crypto key generate rsa
```

Masukkan **1024** pada pesan konfirmasi "`How many bits in the modulus [512]:`" yang tampil.

- Membuat akun otentikasi dengan nama login "**putu**" yang memiliki **privilege level 15** (hak akses penuh terhadap perangkat) dan sandi login "**cisco**".

```
CORE(config)# username putu privilege 15 secret cisco
```

KONFIGURASI DASAR PADA ROUTER CORE (3)

- Mengaktifkan fitur **Secure Copy (SCP) Server**

```
CORE(config)# ip scp server enable
```

- Mengatur akses SSH untuk 5 (lima) koneksi virtual secara bersamaan

```
CORE(config)# line vty 0 4
```

- Mengatur agar otentikasi menggunakan *local user database*

```
CORE(config-line)# login local
```

- Mengatur agar hanya menerima koneksi virtual melalui SSH

```
CORE(config-line)# transport input ssh
```

- Berpindah ke mode *privilege*

```
CORE(config-line)# end
```

- Menyimpan konfigurasi secara permanen

```
CORE# write mem
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW1 (1)

- Berpindah ke mode *global configuration*

```
Router# conf t
```

- Mengatur *hostname*

```
Router(config)# hostname SW1
```

- Berpindah ke *interface vlan 1*

```
SW1(config)# int vlan 1
```

- Mengatur pengalamatan IP pada *interface vlan 1*

```
SW1(config-if)# ip address 192.168.0.11 255.255.255.0
```

- Mengaktifkan *interface*

```
SW1(config-if)# no shutdown
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW1 (2)

- Berpindah ke satu mode sebelumnya

```
SW1(config-if)# exit
```

- Mengatur nama domain yang digunakan oleh perangkat

```
SW1(config)# ip domain-name ubg.local
```

- Mengatur agar perangkat menggunakan SSH versi 2

```
SW1(config)# ip ssh version 2
```

- Menghasilkan RSA Key dengan modulus 1024

```
SW1(config)# crypto key generate rsa
```

Masukkan **1024** pada pesan konfirmasi "`How many bits in the modulus [512]:`" yang tampil.

- Membuat akun otentikasi dengan nama login "**putu**" yang memiliki **privilege level 15** (hak akses penuh terhadap perangkat) dan sandi login "**cisco**".

```
SW1(config)# username putu privilege 15 secret cisco
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW1 (3)

- Mengaktifkan fitur **Secure Copy (SCP) Server**

```
SW1(config)# ip scp server enable
```

- Mengatur akses SSH untuk 5 (lima) koneksi virtual secara bersamaan

```
SW1(config)# line vty 0 4
```

- Mengatur agar otentikasi menggunakan *local user database*

```
SW1(config-line)# login local
```

- Mengatur agar hanya menerima koneksi virtual melalui SSH

```
SW1(config-line)# transport input ssh
```

- Berpindah ke mode *privilege*

```
SW1(config-line)# end
```

- Menyimpan konfigurasi secara permanen

```
SW1# write mem
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW2 (1)

- Berpindah ke mode *global configuration*

```
Router# conf t
```

- Mengatur *hostname*

```
Router(config)# hostname SW2
```

- Berpindah ke *interface vlan 1*

```
SW2(config)# int vlan 1
```

- Mengatur pengalamatan IP pada *interface vlan 1*

```
SW2(config-if)# ip address 192.168.0.12 255.255.255.0
```

- Mengaktifkan *interface*

```
SW2(config-if)# no shutdown
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW2 (2)

- Berpindah ke satu mode sebelumnya

```
SW2(config-if)# exit
```

- Mengatur nama domain yang digunakan oleh perangkat

```
SW2(config)# ip domain-name ubg.local
```

- Mengatur agar perangkat menggunakan SSH versi 2

```
SW2(config)# ip ssh version 2
```

- Menghasilkan RSA Key dengan modulus 1024

```
SW2(config)# crypto key generate rsa
```

Masukkan **1024** pada pesan konfirmasi "`How many bits in the modulus [512]:`" yang tampil.

- Membuat akun otentikasi dengan nama login "**putu**" yang memiliki **privilege level 15** (hak akses penuh terhadap perangkat) dan sandi login "**cisco**".

```
SW2(config)# username putu privilege 15 secret cisco
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW2 (3)

- Mengaktifkan fitur **Secure Copy (SCP) Server**

```
SW2(config)# ip scp server enable
```

- Mengatur akses SSH untuk 5 (lima) koneksi virtual secara bersamaan

```
SW2(config)# line vty 0 4
```

- Mengatur agar otentikasi menggunakan *local user database*

```
SW2(config-line)# login local
```

- Mengatur agar hanya menerima koneksi virtual melalui SSH

```
SW2(config-line)# transport input ssh
```

- Berpindah ke mode *privilege*

```
SW2(config-line)# end
```

- Menyimpan konfigurasi secara permanen

```
SW2# write mem
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW3 (1)

- Berpindah ke mode *global configuration*

```
Router# conf t
```

- Mengatur *hostname*

```
Router(config)# hostname SW3
```

- Berpindah ke *interface vlan 1*

```
SW3(config)# int vlan 1
```

- Mengatur pengalamatan IP pada *interface vlan 1*

```
SW3(config-if)# ip address 192.168.0.13 255.255.255.0
```

- Mengaktifkan *interface*

```
SW3(config-if)# no shutdown
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW3 (2)

- Berpindah ke satu mode sebelumnya

```
SW3(config-if)# exit
```

- Mengatur nama domain yang digunakan oleh perangkat

```
SW3(config)# ip domain-name ubg.local
```

- Mengatur agar perangkat menggunakan SSH versi 2

```
SW3(config)# ip ssh version 2
```

- Menghasilkan RSA Key dengan modulus 1024

```
SW3(config)# crypto key generate rsa
```

Masukkan **1024** pada pesan konfirmasi "`How many bits in the modulus [512]:`" yang tampil.

- Membuat akun otentikasi dengan nama login "**putu**" yang memiliki **privilege level 15** (hak akses penuh terhadap perangkat) dan sandi login "**cisco**".

```
SW3(config)# username putu privilege 15 secret cisco
```

KONFIGURASI DASAR PADA CISCO MULTILAYER SWITCH SW3 (3)

- Mengaktifkan fitur **Secure Copy (SCP) Server**

```
SW3(config)# ip scp server enable
```

- Mengatur akses SSH untuk 5 (lima) koneksi virtual secara bersamaan

```
SW3(config)# line vty 0 4
```

- Mengatur agar otentikasi menggunakan *local user database*

```
SW3(config-line)# login local
```

- Mengatur agar hanya menerima koneksi virtual melalui SSH

```
SW3(config-line)# transport input ssh
```

- Berpindah ke mode *privilege*

```
SW3(config-line)# end
```

- Menyimpan konfigurasi secara permanen

```
SW3# write mem
```

KONEKSI KE PERANGKAT JARINGAN

- Membuat **file python script** dengan nama "**simple.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano simple.py`
- Membuat kode program sederhana untuk melakukan koneksi SSH ke perangkat jaringan dan menemukan serta menampilkan informasi **prompt CLI** dari perangkat jaringan tersebut.

```
 GNU nano 4.8                                  simple.py
1 from netmiko import ConnectHandler
2 net_connect = ConnectHandler(
3     device_type="cisco_ios",
4     host="core.ubg.local",
5     username="putu",
6     password="cisco"
7 )
8 print(net_connect.find_prompt())
9 net_connect.disconnect()
10
```

KONEKSI KE PERANGKAT JARINGAN

Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2-7: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat parameter-parameter atau **argument** yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**. Parameter **device_type** memuat jenis perangkat yaitu **cisco_ios**. Paramater **host** memuat alamat IP atau *hostname* dari perangkat yaitu "**core.ubg.local**" yang merupakan *hostname* dari **Cisco Router CORE**. Sedangkan parameter **username** yaitu "**putu**" dan **password** yaitu "**cisco**" merupakan akun otentikasi SSH. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 8: Menemukan **prompt CLI** dari perangkat jaringan menggunakan **method find_prompt()** dan menampilkan informasi **prompt CLI** tersebut menggunakan fungsi **print().**
- Baris 9: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

KONEKSI KE PERANGKAT JARINGAN

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**simple.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.
- Eksekusi **file python script** menggunakan perintah:

```
# python3 simple.py
```

- Hasil dari eksekusi program memperlihatkan **prompt** dari perangkat jaringan yaitu **CORE#**.

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

JENIS-JENIS PERANGKAT YANG TERSEDIA

- Membuat **file python script** dengan nama "**device-types.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano device-types.py`
- Membuat kode program untuk menampilkan informasi jenis-jenis perangkat yang didukung oleh **Netmiko** dengan mengatur nilai dari parameter **device_type** menggunakan **invalid**. Koneksi melalui **SSH** dengan **login** menggunakan **username** "**putu**" dan **password** "**cisco**" ke perangkat dengan alamat IP atau **hostname** "**core.ubg.local**".

```
 GNU nano 4.8                                  device-types.py
1 from netmiko import ConnectHandler
2 net_connect = ConnectHandler(
3     device_type="invalid",
4     host="core.ubg.local",
5     username="putu",
6     password="cisco"
7 )
8
```

JENIS-JENIS PERANGKAT YANG TERSEDIA

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**device-types.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.
- Eksekusi **file python script** menggunakan perintah:

```
# python3 device-types.py
```

- Cuplikan hasil dari eksekusi program memperlihatkan beberapa jenis perangkat yang didukung oleh **Netmiko**.

```
root@NetworkAutomation-1:~# python3 device-types.py
Traceback (most recent call last):
  File "device-types.py", line 2, in <module>
    net_connect = ConnectHandler(
  File "/usr/local/lib/python3.8/dist-packages/netmiko/ssh_dispatcher.py", line 310, in ConnectH
andler
    raise ValueError(
ValueError: Unsupported 'device_type' currently supported platforms are:
a10
accedian
adtran_os
alcatel_aos
alcatel_sros
apresia_aeos
```

DUKUNGAN PLATFORM

Diuji berkala:

- Arista vEOS
- Cisco ASA
- **Cisco IOS**
- Cisco IOS-XE
- Cisco IOS-XR
- Cisco NX-OS
- Cisco SG300
- HP ProCurve
- Juniper Junos
- Linux

Pengujian terbatas:

- 6Wind
- Adtran OS
- Alcatel AOS6/AOS8
- Apresia Systems AEOS
- Broadcom ICOS
- Calix B6
- Centec Networks
- Cisco AireOS (Wireless LAN Controllers)
- CloudGenix ION
- Dell OS9 (Force10)

Pengujian terbatas:

- Dell OS10
- Dell PowerConnect
- Ericsson IPOS
- Extreme ERS (Avaya)
- Extreme VSP (Avaya)
- Extreme VDX (Brocade)
- Extreme MLX/NetIron (Brocade/Foundry)
- HPE Comware7
- Huawei
- Huawei OLT

Pengujian terbatas:

- Huawei SmartAX
- IP Infusion OcNOS
- Juniper ScreenOS
- Mellanox
- MikroTik RouterOS
- MikroTik SwitchOS
- NetApp cDOT
- Netgear ProSafe
- Nokia/Alcatel SR OS
- OneAccess

DUKUNGAN PLATFORM

Pengujian terbatas:

- Palo Alto PAN-OS
- Pluribus
- Ruckus ICX/FastIron
- Ruijie Networks
- Supermicro SMIS
- TPLink JetStream
- Ubiquiti EdgeSwitch
- Vyatta VyOS
- Yamaha
- ZTE ZXROS

Eksperimental:

- A10
- Accedian
- Allied Telesis AlliedWare Plus
- Aruba
- Brocade Fabric OS
- C-DOT CROS
- Ciena SAOS
- Citrix Netscaler
- Cisco Telepresence
- Cisco Viptela
- Check Point GAiA

Eksperimental:

- Coriant
- Dell OS6
- Dell EMC Isilon
- Eltex
- Enterasys
- Endace
- Extreme EXOS
- Extreme Wing
- Extreme SLX (Brocade)
- F5 TMSH
- F5 Linux
- Fortinet

Eksperimental:

- MRV Communications OptiSwitch
- MRV LX
- Nokia/Alcatel SR-OS
- QuantaMesh
- Rad ETX
- Raisecom ROAP
- Sophos SFOS
- Ubiquiti Unifi Switch
- Versa Networks FlexVNF
- Watchguard Firebox
- 6WIND TurboRouter

MEMBUAT DICTIONARY SEBAGAI REPRESENTASI PERANGKAT

- Membuat **file python script** dengan nama "**dictionary.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano dictionary.py`
- Membuat kode program untuk koneksi ke perangkat jaringan menggunakan struktur data **dictionary** dari **python**.

```
 GNU nano 4.8                                  dictionary.py
1 from netmiko import ConnectHandler
2 core = {
3     "device_type": "cisco_ios",
4     "host": "core.ubg.local",
5     "username": "putu",
6     "password": "cisco"
7 }
8 net_connect = ConnectHandler(**core)
9 print(net_connect.find_prompt())
10 net_connect.disconnect()
11
```

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.

MEMBUAT DICTIONARY SEBAGAI REPRESENTASI PERANGKAT

- Baris 2 sampai dengan 7 dari kode program memperlihatkan deklarasi **variable** dengan nama **core** bertipe data **dictionary**. Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**. *Key* **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**core.ubg.local**" yang merupakan *hostname* dari **Cisco Router CORE**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.
- Baris 8: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****core**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 9: Menemukan **prompt CLI** dari perangkat jaringan menggunakan **method find_prompt()** dan menampilkan informasi **prompt CLI** tersebut menggunakan fungsi **print().**
- Baris 10: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**dictionary.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

MEMBUAT DICTIONARY SEBAGAI REPRESENTASI PERANGKAT

- Eksekusi **file python script** menggunakan perintah:

```
# python3 dictionary.py
```

- Hasil dari eksekusi program memperlihatkan **prompt** dari perangkat jaringan yaitu **CORE#**.

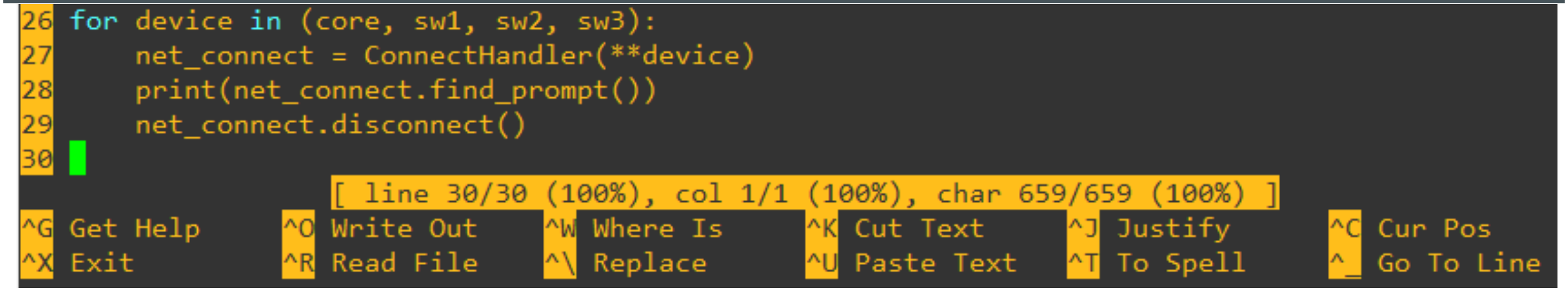
```
root@NetworkAutomation-1:~# python3 dictionary.py
CORE#
root@NetworkAutomation-1:~#
```

KONEKSI KE BEBERAPA PERANGKAT

- Membuat **file python script** dengan nama "**multiple-devices.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano multiple-devices.py`
- Membuat kode program untuk koneksi ke beberapa perangkat jaringan menggunakan struktur data **dictionary** dari **python**. Pada **dictionary** memuat data terkait jenis perangkat yaitu **cisco_ios**, **username** "**putu**" dan **password** "**cisco**" serta alamat IP atau **hostname** dari setiap perangkat untuk koneksi **SSH**.

```
 GNU nano 4.8                                  multiple-devices.py
1  from netmiko import ConnectHandler
2  core = {
3      "device_type": "cisco_ios",
4      "host": "core.ubg.local",
5      "username": "putu",
6      "password": "cisco"
7  }
8  sw1 = {
9      "device_type": "cisco_ios",
10     "host": "sw1.ubg.local",
11     "username": "putu",
12     "password": "cisco"
13 }
14 sw2 = {
15     "device_type": "cisco_ios",
16     "host": "sw2.ubg.local",
17     "username": "putu",
18     "password": "cisco"
19 }
20 sw3 = {
21     "device_type": "cisco_ios",
22     "host": "sw3.ubg.local",
23     "username": "putu",
24     "password": "cisco"
25 }
```

KONEKSI KE BEBERAPA PERANGKAT



Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2-7: Deklarasi **variable** bertipe data **dictionary** dengan nama **core** untuk perangkat **Cisco Router CORE** dengan *hostname* "**core.ubg.local**".
- Baris 8-13:Deklarasi **variable** bertipe data **dictionary** dengan nama **sw1** untuk perangkat **Cisco Multilayer Switch SW1** dengan *hostname* "**sw1.ubg.local**"
- Baris 14-19:Deklarasi **variable** bertipe data **dictionary** dengan nama **sw2** untuk perangkat **Cisco Multilayer Switch SW2** dengan *hostname* "**sw2.ubg.local**"

KONEKSI KE BEBERAPA PERANGKAT

- Baris 20-25: Deklarasi **variable** bertipe data **dictionary** dengan nama **sw3** untuk perangkat **Cisco Multilayer Switch SW3** dengan *hostname* "**sw3.ubg.local**"
- Baris 26: Melakukan perulangan sejumlah *dictionary variable* yaitu **core**, **sw1**, **sw2**, **sw3** dan menyimpannya pada variable **device**.
- Baris 27: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****device**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 28: Menemukan **prompt CLI** dari perangkat jaringan menggunakan **method find_prompt()** dan menampilkan informasi **prompt CLI** tersebut menggunakan fungsi **print().**
- Baris 29: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**multiple-devices.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

KONEKSI KE BEBERAPA PERANGKAT

- Eksekusi **file python script** menggunakan perintah:

```
# python3 multiple-devices.py
```

- Hasil dari eksekusi program memperlihatkan **prompt** dari setiap perangkat jaringan yaitu `CORE#`, `SW1#`, `SW2#`, `SW3#`.

```
root@NetworkAutomation-1:~# python3 multiple-devices.py
CORE#
SW1#
SW2#
SW3#
root@NetworkAutomation-1:~#
```

MENGEKSEKUSI PERINTAH SHOW

- Membuat **file python script** dengan nama "**show-ip.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano show-ip.py`
- Membuat kode program untuk menampilkan informasi pengalamatan IP dan status dari setiap *interface* yang terdapat pada perangkat **Cisco Router CORE** dengan mengeksekusi perintah "**show ip int brief**".

```
 GNU nano 4.8                                    show-ip.py
 1 from netmiko import ConnectHandler
 2 core = {
 3     "device_type": "cisco_ios",
 4     "host": "core.ubg.local",
 5     "username": "putu",
 6     "password": "cisco"
 7 }
 8 net_connect = ConnectHandler(**core)
 9 output = net_connect.send_command("show ip int brief")
10 print(output)
11 net_connect.disconnect()
12
```

MENGEKSEKUSI PERINTAH SHOW

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2 sampai dengan 7 dari kode program memperlihatkan deklarasi **variable** dengan nama **core** bertipe data **dictionary**. Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**.
  Key **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**core.ubg.local**" yang merupakan *hostname* dari **Cisco Router CORE**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.
- Baris 8: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****core**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 9: Mengeksekusi perintah "**show ip int brief**" menggunakan **method send_command()** dan menyimpan hasilnya ke *variable* **output**.
- Baris 10: Menampilkan isi dari *variable* **output** menggunakan fungsi **print().**
- Baris 11: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

MENGEKSEKUSI PERINTAH SHOW

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**show-ip.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.
- Eksekusi **file python script** menggunakan perintah:

```
# python3 show-ip.py
```

- Hasil eksekusi dari program tersebut salah satunya memperlihatkan bahwa **interface Vlan1** menggunakan alamat IP **192.168.0.1** dengan **status up** dan **protocol up**.

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

Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        unassigned      YES NVRAM  administratively down down
FastEthernet0/1        unassigned      YES NVRAM  up                    up
FastEthernet1/0        unassigned      YES NVRAM  administratively down down
FastEthernet2/0        unassigned      YES NVRAM  administratively down down
FastEthernet3/0        unassigned      YES unset  up                    up
FastEthernet3/1        unassigned      YES unset  up                    up
FastEthernet3/2        unassigned      YES unset  up                    up
FastEthernet3/3        unassigned      YES unset  up                    up
FastEthernet3/4        unassigned      YES unset  up                    down
FastEthernet3/5        unassigned      YES unset  up                    down
FastEthernet3/6        unassigned      YES unset  up                    down
FastEthernet3/7        unassigned      YES unset  up                    down
FastEthernet3/8        unassigned      YES unset  up                    down
FastEthernet3/9        unassigned      YES unset  up                    down
FastEthernet3/10       unassigned      YES unset  up                    down
FastEthernet3/11       unassigned      YES unset  up                    down
FastEthernet3/12       unassigned      YES unset  up                    down
FastEthernet3/13       unassigned      YES unset  up                    down
FastEthernet3/14       unassigned      YES unset  up                    down
FastEthernet3/15       unassigned      YES unset  up                    down
Vlan1                  192.168.0.1     YES NVRAM  up                    up
root@NetworkAutomation-1:~#
```

EKSEKUSI PERINTAH UNTUK MENGUBAH DAN MENYIMPAN KONFIGURASI PADA CISCO IOS

NETMIKO

- Membuat **file python script** dengan nama "**dhcp-client.py**" menggunakan editor **nano** dengan mengeksekusi perintah:

```
# nano dhcp-client.py
```

- Membuat kode program untuk mengatur pengalamatan IP secara dinamis atau sebagai **DHCP Client** pada **interface FastEthernet0/0** yang terhubung ke *Internet* pada perangkat **Cisco Router CORE** dan mengaktifkan *interface* tersebut dengan mengeksekusi perintah:

```
interface fastethernet0/0
    ip address dhcp
    no shutdown
```

- Selain itu juga menyimpan konfigurasi secara permanen.

```
GNU nano 4.8                                    dhcp-client.py
from netmiko import ConnectHandler
core = {
    "device_type": "cisco_ios",
    "host": "core.ubg.local",
    "username": "putu",
    "password": "cisco"
}
commands = ["interface fastetherne0/0","ip address dhcp","no shutdown"]
net_connect = ConnectHandler(**core)
output = net_connect.send_config_set(commands)
output += net_connect.save_config()
print(output)
net_connect.disconnect()
```

EKSEKUSI PERINTAH UNTUK MENGUBAH DAN MENYIMPAN KONFIGURASI PADA CISCO IOS

Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2 sampai dengan 7 dari kode program memperlihatkan deklarasi **variable** dengan nama **core** bertipe data **dictionary**. Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**.
  Key **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**core.ubg.local**" yang merupakan *hostname* dari **Cisco Router CORE**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.

  Baris 8: Deklarasi *variable* bertipe data **list** dengan nama **commands** yang memuat perintah-perintah konfigurasi yang akan dieksekusi di **global configuration mode** meliputi `interface fastethernet0/0`, `ip address dhcp` dan `no shutdown`.
- Baris 9: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****core**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 10: Mengeksekusi perintah konfigurasi yang tersimpan pada *variable* **commands** menggunakan **method send_config_set()** dan menyimpan hasilnya ke *variable* **output**.

EKSEKUSI PERINTAH UNTUK MENGUBAH DAN MENYIMPAN KONFIGURASI PADA CISCO IOS

- Baris 11: Menyimpan konfigurasi secara permanen dari **Random Access Memory (RAM)** ke **Non Volatile Random Access Memory (NVRAM)** menggunakan **method save_config ()** dan menyimpan hasilnya ke *variable* **output**. Metode ini serupa dengan perintah CLI dari **Cisco IOS** yaitu `write mem`.
- Baris 12: Menampilkan isi dari *variable* **output** menggunakan fungsi **print().**
- Baris 13: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**dhcp-client.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

- Eksekusi **file python script** menggunakan perintah:

```
# python3 dhcp-client.py
```

EKSEKUSI PERINTAH UNTUK MENGUBAH DAN MENYIMPAN KONFIGURASI PADA CISCO IOS

- Hasil eksekusi dari program tersebut memperlihatkan perintah-perintah CLI dari **Cisco IOS**. Mulai dari berpindah dari mode *privilege* ke mode *global configuration* (`configure terminal`). Berpindah ke mode *interface configuration* untuk *fastethernet0/0* (`interface fastethernet0/0`). Mengatur pengalamatan IP secara dinamis atau DHCP (`ip address dhcp`). Mengaktifkan *interface* (`no shutdown`). Berpindah dari mode *interface configuration* ke *privilege mode* (`end`). Terakhir menyimpan konfigurasi secara permanen (`write mem`).

```
root@NetworkAutomation-1:~# python3 dhcp-client.py
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
CORE(config)#interface fastetherne0/0
CORE(config-if)#ip address dhcp
CORE(config-if)#no shutdown
CORE(config-if)#end
CORE#write mem

Building configuration...
[OK]
CORE#
root@NetworkAutomation-1:~#
```

VERIFIKASI KONFIGURASI DHCP CLIENT PADA INTERFACE F0/0 DARI ROUTER CORE

- Eksekusi **file python script** menggunakan perintah:

```
# python3 show-ip.py
```

- Hasil eksekusi dari program tersebut salah satunya memperlihatkan bahwa **interface FastEthernet0/0** telah memperoleh pengalamatan IP secara dinamis melalui metode **DHCP** yaitu **192.168.126.134** dan *interface* dalam keadaan aktif dimana ditandai dengan **status up** serta **protocol up**.

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

Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            192.168.126.134 YES DHCP   up                    up
FastEthernet0/1            unassigned      YES NVRAM  up                    up
FastEthernet1/0            unassigned      YES NVRAM  administratively down down
FastEthernet2/0            unassigned      YES NVRAM  administratively down down
FastEthernet3/0            unassigned      YES unset  up                    up
FastEthernet3/1            unassigned      YES unset  up                    up
FastEthernet3/2            unassigned      YES unset  up                    up
FastEthernet3/3            unassigned      YES unset  up                    up
FastEthernet3/4            unassigned      YES unset  up                    down
FastEthernet3/5            unassigned      YES unset  up                    down
FastEthernet3/6            unassigned      YES unset  up                    down
FastEthernet3/7            unassigned      YES unset  up                    down
FastEthernet3/8            unassigned      YES unset  up                    down
FastEthernet3/9            unassigned      YES unset  up                    down
FastEthernet3/10           unassigned      YES unset  up                    down
FastEthernet3/11           unassigned      YES unset  up                    down
FastEthernet3/12           unassigned      YES unset  up                    down
FastEthernet3/13           unassigned      YES unset  up                    down
FastEthernet3/14           unassigned      YES unset  up                    down
FastEthernet3/15           unassigned      YES unset  up                    down
Vlan1                      192.168.0.1     YES NVRAM  up                    up
root@NetworkAutomation-1:~#
```

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

- Membuat **file text** dengan nama "**config-management-router.txt**" menggunakan editor **nano** dengan mengeksekusi perintah: `# nano config-management-router.txt`
- Pada **file text** memuat perintah-perintah **Cisco IOS** untuk manajemen konfigurasi **InterVLAN routing** dan **DHCP Server**.

Penjelasan kode program:

- Baris 1-3: Perintah *Cisco IOS* untuk berpindah ke **interface FastEthernet0/1** dan mengatur agar *interface* tidak menggunakan pengalamatan IP serta mengaktifkan *interface* tersebut.

```
 GNU nano 4.8                          config-management-router.txt
 1 interface FastEthernet0/1
 2  no ip address
 3  no shutdown
 4 interface FastEthernet0/1.2
 5  description trunk untuk VLAN 2
 6  encapsulation dot1Q 2
 7  ip address 192.168.2.1 255.255.255.0
 8 interface FastEthernet0/1.3
 9  description trunk untuk VLAN 3
10  encapsulation dot1Q 3
11  ip address 192.168.3.1 255.255.255.0
12 ip dhcp pool MARKETING
13    network 192.168.2.0 255.255.255.0
14    default-router 192.168.2.1
15    dns-server 192.168.2.1
16 ip dhcp pool SALES
17    network 192.168.3.0 255.255.255.0
18    default-router 192.168.3.1
19    dns-server 192.168.3.1
20    exit
21 ip dhcp excluded-address 192.168.2.1
22 ip dhcp excluded-address 192.168.3.1
```

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

- Baris 4-7: Perintah Cisco IOS untuk membuat **subinterface FastEthernet0/1.2** yang difungsikan sebagai trunk untuk VLAN 2 Marketing dan mengatur agar menggunakan alamat **IP 192.168.2.1/24**.
- Baris 8-11: Perintah *Cisco IOS* untuk membuat **subinterface FastEthernet0/1.3** yang difungsikan sebagai **trunk** untuk **VLAN 3 Sales** dan mengatur agar menggunakan alamat **IP 192.168.3.1/24**.
- Baris 12-15: Perintah *Cisco IOS* untuk membuat **IP DHCP Pool** atau rentang alamat IP yang didistribusikan secara dinamis dengan nama "**MARKETING**" ke *client* di **VLAN 2** menggunakan alamat **network 192.168.2.0/24** dan **default gateway 192.168.2.1**.
- Baris 16-19:Perintah *Cisco IOS* untuk membuat IP DHCP Pool atau rentang alamat IP yang didistribusikan secara dinamis dengan nama "SALES" ke client di VLAN 3 menggunakan alamat network 192.168.3.0/24 dan default gateway 192.168.3.1.
- Baris 20: Perintah *Cisco IOS* untuk berpindah ke satu mode sebelumnya.

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

- Baris 21: Perintah *Cisco IOS* untuk mengatur alamat IP yang tidak disewakan oleh **DHCP Server** ke **Client** yaitu **192.168.2.1**.
- Baris 22: Perintah *Cisco IOS* untuk mengatur alamat IP yang tidak disewakan oleh **DHCP Server** ke **Client** yaitu **192.168.3.1**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**config-management-router.txt**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

- Membuat **file python script** dengan nama "**config-core.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano config-core.py`
- Membuat kode program untuk mengubah konfigurasi pada perangkat **Cisco Router CORE** yang bersumber dari **file text** dengan nama "**config-management-router.txt**". Pada **file text** tersebut telah memuat perintah-perintah **Cisco IOS** untuk manajemen konfigurasi **InterVLAN routing** dan **DHCP Server**.

```
 GNU nano 4.8                                config-core.py
1 from netmiko import ConnectHandler
2 core = {
3     "device_type": "cisco_ios",
4     "host": "core.ubg.local",
5     "username": "putu",
6     "password": "cisco"
7 }
8 cfg_file="config-management-router.txt"
9 net_connect = ConnectHandler(**core)
10 output = net_connect.send_config_from_file(cfg_file)
11 output += net_connect.save_config()
12 print(output)
13 net_connect.disconnect()
14
```

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2 sampai dengan 7 dari kode program memperlihatkan deklarasi **variable** dengan nama **core** bertipe data **dictionary**. Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**.
  Key **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**core.ubg.local**" yang merupakan *hostname* dari **Cisco Router CORE**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.
- Baris 8: Deklarasi *variable* dengan nama **cfg_file** yang memuat nama **file text** yaitu "**config-management-router.txt**" . Pada **file text** tersebut telah memuat perintah-perintah **Cisco IOS** untuk manajemen konfigurasi **InterVLAN routing** dan **DHCP Server** yang akan dieksekusi pada **router CORE**.
- Baris 9: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****core**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

- Baris 10: Mengeksekusi perintah konfigurasi yang tersimpan pada *variable* **cfg_file** menggunakan **method send_config_from_file()** dan menyimpan hasilnya ke *variable* **output**.
- Baris 11: Menyimpan konfigurasi secara permanen dari **Random Access Memory (RAM)** ke **Non Volatile Random Access Memory (NVRAM)** menggunakan **method save_config ()** dan menyimpan hasilnya ke *variable* **output**. Metode ini serupa dengan perintah CLI dari **Cisco IOS** yaitu `write mem`.
- Baris 12: Menampilkan isi dari *variable* **output** menggunakan fungsi **print().**
- Baris 13: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**config-core.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

EKSEKUSI PERINTAH UNTUK MENGUBAH KONFIGURASI DARI FILE PADA CISCO IOS

- Eksekusi **file python script** menggunakan perintah:

```
# python3 config-core.py
```

- Hasil dari eksekusi kode program tersebut memperlihatkan bahwa pengaturan **InterVLAN Routing** dan **DHCP Server** pada **router CORE** telah berhasil dilakukan.

```
root@NetworkAutomation-1:~# python3 config-core.py
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
CORE(config)#interface FastEthernet0/1
CORE(config-if)# no ip address
CORE(config-if)# no shutdown
CORE(config-if)#interface FastEthernet0/1.2
CORE(config-subif)# description trunk untuk VLAN 2
CORE(config-subif)# encapsulation dot1Q 2
CORE(config-subif)# ip address 192.168.2.1 255.255.255.0
CORE(config-subif)#interface FastEthernet0/1.3
CORE(config-subif)# description trunk untuk VLAN 3
CORE(config-subif)# encapsulation dot1Q 3
CORE(config-subif)# ip address 192.168.3.1 255.255.255.0
CORE(config-subif)#ip dhcp pool MARKETING
CORE(dhcp-config)#   network 192.168.2.0 255.255.255.0
CORE(dhcp-config)#   default-router 192.168.2.1
CORE(dhcp-config)#   dns-server 192.168.2.1
CORE(dhcp-config)#ip dhcp pool SALES
CORE(dhcp-config)#   network 192.168.3.0 255.255.255.0
CORE(dhcp-config)#   default-router 192.168.3.1
CORE(dhcp-config)#   dns-server 192.168.3.1
CORE(dhcp-config)#   exit
CORE(config)#ip dhcp excluded-address 192.168.2.1
CORE(config)#ip dhcp excluded-address 192.168.3.1
CORE(config)#end
CORE#write mem

Building configuration...
[OK]
CORE#
root@NetworkAutomation-1:~#
```

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Terdapat 2 (dua) **VLAN** yang dibuat pada setiap **MSW** yaitu **VLAN ID 2** dengan **name "MARKETING"** dan **VLAN ID 3** dengan **name "SALES"**.
- **Port membership** untuk **VLAN ID 2** di setiap **MSW** adalah **interface FastEthernet3/1**.
- **Port membership** untuk **VLAN ID 3** di setiap **MSW** adalah **interface FastEthernet3/2**.

VLAN ID	Name	Port Membership
2	MARKETING	**FastEthernet3/1**
3	SALES	**FastEthernet3/2**

- Terdapat 2 (dua) **interface** dengan **mode trunk** yang diatur pada setiap **MSW** yaitu **FastEthernet3/13** dan **FastEthernet3/14**.
- Terdapat 2 (dua) **file text** yang digunakan untuk menyimpan perintah-perintah konfigurasi *Cisco IOS* pada MSW yaitu "**vlan-database.txt**" membuat perintah konfigurasi **pembuatan VLAN** dan "**config-management-switch.txt**" memuat perintah konfigurasi **port membership per VLAN** dan **interface trunk**.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Membuat **file text** dengan nama "**vlan-database.txt**" menggunakan editor **nano** dengan mengeksekusi perintah:
  ```
  # nano vlan-database.txt
  ```
- Pada **file text** memuat perintah-perintah **Cisco IOS** untuk membuat **VLAN** yang akan dieksekusi di ketiga **Multilayer Switch (MSW)** yaitu **SW1**, **SW2** dan **SW3**.

```
 GNU nano 4.8                                  vlan-database.txt
1 vlan 2 name MARKETING
2 vlan 3 name SALES
3 exit
4
```

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

Penjelasan konten dari **file text**:

- Baris 1: Membuat VLAN dengan ID **2** dan mengatur namanya dengan **MARKETING**.
- Baris 2: Membuat VLAN dengan ID **3** dan mengatur namanya dengan **SALES**.
- Baris 3: Menyimpan perubahan dan keluar dari konfigurasi **vlan database**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**vlan-database.txt**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Membuat **file text** dengan nama "**config-management-switch.txt**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano config-management-switch.txt`
- Pada **file text** memuat perintah-perintah **Cisco IOS** untuk pengaturan **port membership** dan **interface trunk** pada ketiga **Multilayer Switch (MSW)** yaitu **SW1**, **SW2** dan **SW3**.

```
 GNU nano 4.8                          config-management-switch.txt
1 interface FastEthernet3/1
2  switchport mode access
3  switchport access vlan 2
4 interface FastEthernet3/2
5  switchport mode access
6  switchport access vlan 3
7 interface range FastEthernet3/13-14
8  switchport mode trunk
9
```

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

Penjelasan konten dari **file text**:

- Baris 1-3: Berpindah ke *interface configuration* untuk **FastEthernet3/1** dan mengatur agar *interface* tersebut sebagai **access port** serta menjadi anggota dari **VLAN 2**.
- Baris 4-6: Berpindah ke *interface configuration* untuk **FastEthernet3/2** dan mengatur agar *interface* tersebut sebagai **access port** serta menjadi anggota dari **VLAN 3**.
- Baris 7-8: Berpindah ke *interface configuration* untuk **FastEthernet3/13** sampai dengan **FastEthernet3/14** dan mengatur agar *interface* tersebut sebagai **trunk port**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**config-management-switch.txt**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Membuat **file python script** dengan nama "**config-switch.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano config-switch.py`
- Membuat kode program untuk mengubah konfigurasi pada ketiga perangkat **Cisco Multilayer Switch** yang bersumber dari **file text** dengan nama "**vlan-database.txt**" dan "**config-management-switch.txt**". Pada **file text** tersebut telah memuat perintah-perintah **Cisco IOS** untuk manajemen konfigurasi **VLAN** dan **port membership** serta **interface trunk**.

```
GNU nano 4.8                                    config-switch.py
from netmiko import ConnectHandler
sw1 = {
    "device_type": "cisco_ios",
    "host": "sw1.ubg.local",
    "username": "putu",
    "password": "cisco"
}
sw2 = {
    "device_type": "cisco_ios",
    "host": "sw2.ubg.local",
    "username": "putu",
    "password": "cisco"
}
sw3 = {
    "device_type": "cisco_ios",
    "host": "sw3.ubg.local",
    "username": "putu",
    "password": "cisco"
}
```

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

NETMIKO

```
20 cfg_vlan = "vlan-database.txt"
21 cfg_file = "config-management-switch.txt"
22 for device in (sw1, sw2, sw3):
23     net_connect = ConnectHandler(**device)
24     output = net_connect.send_config_from_file(cfg_vlan, config_mode_command='vlan database')
25     output += net_connect.send_config_from_file(cfg_file)
26     output += net_connect.save_config()
27     print(output)
28     print("*"*30)
29     net_connect.disconnect()
30
```

Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2-7: Deklarasi **variable** bertipe data **dictionary** dengan nama **sw1** untuk perangkat **Cisco Multilayer Switch** dengan *hostname* "**sw1.ubg.local**". Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

Key **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**sw1.ubg.local**" yang merupakan *hostname* dari **Cisco Multilayer Switch SW1**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.

- Baris 8-13: Deklarasi **variable** bertipe data **dictionary** dengan nama **sw2** untuk perangkat **Cisco Multilayer Switch** dengan *hostname* "**sw2.ubg.local**". Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**.

  Key **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**sw2.ubg.local**" yang merupakan *hostname* dari **Cisco Multilayer Switch SW2**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

NETMIKO

- Baris 14-19: Deklarasi **variable** bertipe data **dictionary** dengan nama **sw3** untuk perangkat **Cisco Multilayer Switch** dengan *hostname* "**sw3.ubg.local.** Pada **dictionary** tersebut memuat data yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**.

  Key **device_type** memuat *value* jenis perangkat yaitu **cisco_ios**. *Key* **host** memuat *value* berupa alamat IP atau *hostname* dari perangkat yaitu "**sw3.ubg.local**" yang merupakan *hostname* dari **Cisco Multilayer Switch SW3**. Sedangkan *key* **username** dengan *value* yaitu "**putu**" dan *key* **password** dengan *value* yaitu "**cisco**" yang merupakan akun otentikasi SSH.

- Baris 20: Deklarasi *variable* dengan nama **cfg_vlan** yang memuat nama **file text** yaitu "**vlan-database.txt**" . Pada **file text** tersebut telah memuat perintah-perintah **Cisco IOS** untuk manajemen konfigurasi **VLAN** yang akan dieksekusi pada setiap perangkat **Cisco Multilayer Switch**.

- Baris 21: Deklarasi *variable* dengan nama **cfg_file** yang memuat nama **file text** yaitu "**config-management-switch.txt**" . Pada **file text** tersebut telah memuat perintah-perintah **Cisco IOS** untuk manajemen konfigurasi **port membership per VLAN** dan pengaturan **interface trunk** yang akan dieksekusi pada setiap perangkat **Cisco Multilayer Switch**.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Baris 22: Melakukan perulangan sejumlah *dictionary variable* yaitu **sw1**, **sw2**, **sw3** dan menyimpannya pada variable **device**.
- Baris 23: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****device**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 24: Mengeksekusi perintah konfigurasi yang tersimpan pada *variable* **cfg_vlan** pada mode konfigurasi **vlan database** menggunakan **method send_config_from_file()** dan menyimpan hasilnya ke *variable* **output**.
- Baris 25: Mengeksekusi perintah konfigurasi yang tersimpan pada *variable* **cfg_file** menggunakan **method send_config_from_file()** dan menyimpan hasilnya ke *variable* **output**.
- Baris 26: Menyimpan konfigurasi secara permanen dari **Random Access Memory (RAM)** ke **Non Volatile Random Access Memory (NVRAM)** menggunakan **method save_config ()** dan menyimpan hasilnya ke *variable* **output**. Metode ini serupa dengan perintah CLI dari **Cisco IOS** yaitu **`write mem`**.
- Baris 27: Menampilkan isi dari *variable* **output** menggunakan fungsi **print().**

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Baris 28: Membuat tanda * sebanyak **30** menggunakan perkalian **"*"*30** sebagai pemisah *output* dari hasil eksekusi perintah *Cisco IOS* pada setiap *switch* dan menampilkannya ke layar menggunakan fungsi **print().**
- Baris 29: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**config-switch.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

- Eksekusi **file python script** menggunakan perintah:

```
# python3 config-switch.py
```

- Hasil dari eksekusi kode program tersebut memperlihatkan bahwa pengaturan **VLAN** dan **port membership** serta **interface trunk** pada **ketiga Multilayer Switch** telah berhasil dilakukan.

```
root@NetworkAutomation-1:~# python3 config-switch.py
vlan database
SW1(vlan)#vlan 2 name MARKETING
VLAN 2 modified:
    Name: MARKETING
SW1(vlan)#vlan 3 name SALES
VLAN 3 modified:
    Name: SALES
SW1(vlan)#exit
APPLY completed.
Exiting....
SW1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)#interface FastEthernet3/1
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 2
SW1(config-if)#interface FastEthernet3/2
SW1(config-if)# switchport mode access
SW1(config-if)# switchport access vlan 3
SW1(config-if)#interface range FastEthernet3/13 - 14
SW1(config-if-range)# switchport mode trunk
SW1(config-if-range)#end
SW1#write mem

Building configuration...
[OK]
SW1#
******************************
```

STUDI KASUS MANAJEMEN VLAN, PORT MEMBERSHIP DAN INTERFACE TRUNK

```
vlan database
SW2(vlan)#vlan 2 name MARKETING
VLAN 2 modified:
    Name: MARKETING
SW2(vlan)#vlan 3 name SALES
VLAN 3 modified:
    Name: SALES
SW2(vlan)#exit
APPLY completed.
Exiting....
SW2#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW2(config)#interface FastEthernet3/1
SW2(config-if)# switchport mode access
SW2(config-if)# switchport access vlan 2
SW2(config-if)#interface FastEthernet3/2
SW2(config-if)# switchport mode access
SW2(config-if)# switchport access vlan 3
SW2(config-if)#interface range FastEthernet3/13 - 14
SW2(config-if-range)# switchport mode trunk
SW2(config-if-range)#end
SW2#write mem

Building configuration...
[OK]
SW2#
*****************************
```

```
vlan database
SW3(vlan)#vlan 2 name MARKETING
VLAN 2 modified:
    Name: MARKETING
SW3(vlan)#vlan 3 name SALES
VLAN 3 modified:
    Name: SALES
SW3(vlan)#exit
APPLY completed.
Exiting....
SW3#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SW3(config)#interface FastEthernet3/1
SW3(config-if)# switchport mode access
SW3(config-if)# switchport access vlan 2
SW3(config-if)#interface FastEthernet3/2
SW3(config-if)# switchport mode access
SW3(config-if)# switchport access vlan 3
SW3(config-if)#interface range FastEthernet3/13 - 14
SW3(config-if-range)# switchport mode trunk
SW3(config-if-range)#end
SW3#write mem

Building configuration...
[OK]
SW3#
*****************************
root@NetworkAutomation-1:~#
```

KONFIGURASI VPCS SEBAGAI DHCP CLIENT

- Mengatur pengalamatan IP secara dinamis atau **DHCP Client** pada setiap **VPCS** dengan mengeksekusi perintah "**ip dhcp**".
- Hasil konfigurasi **DHCP Client** pada **VPCS MKT1, MKT2, MKT3, SALES1, SALES2** dan **SALES3** adalah sebagai berikut:

```
MKT1> ip dhcp
DDORA IP 192.168.2.2/24 GW 192.168.2.1
```

```
SALES1> ip dhcp
DDORA IP 192.168.3.2/24 GW 192.168.3.1
```

```
MKT2> ip dhcp
DORA IP 192.168.2.3/24 GW 192.168.2.1
```

```
SALES2> ip dhcp
DORA IP 192.168.3.3/24 GW 192.168.3.1
```

```
MKT3> ip dhcp
DDORA IP 192.168.2.4/24 GW 192.168.2.1
```

```
SALES3> ip dhcp
DORA IP 192.168.3.4/24 GW 192.168.3.1
```

VERIFIKASI KONEKSI ANTAR VPCS

- Hasil verifikasi koneksi dari **VPCS MKT1** ke **MKT2** dengan mengeksekusi perintah "**ping 192.168.2.3**".

```
MKT1> ping 192.168.2.3
84 bytes from 192.168.2.3 icmp_seq=1 ttl=64 time=2.351 ms
84 bytes from 192.168.2.3 icmp_seq=2 ttl=64 time=4.724 ms
84 bytes from 192.168.2.3 icmp_seq=3 ttl=64 time=2.368 ms
84 bytes from 192.168.2.3 icmp_seq=4 ttl=64 time=4.501 ms
84 bytes from 192.168.2.3 icmp_seq=5 ttl=64 time=1.926 ms
```

Terlihat koneksi berhasil dilakukan.

- Hasil verifikasi koneksi dari **VPCS MKT1** ke **MKT3** dengan mengeksekusi perintah "**ping 192.168.2.4**".

```
MKT1> ping 192.168.2.4
84 bytes from 192.168.2.4 icmp_seq=1 ttl=64 time=3.173 ms
84 bytes from 192.168.2.4 icmp_seq=2 ttl=64 time=2.424 ms
84 bytes from 192.168.2.4 icmp_seq=3 ttl=64 time=4.034 ms
84 bytes from 192.168.2.4 icmp_seq=4 ttl=64 time=2.842 ms
84 bytes from 192.168.2.4 icmp_seq=5 ttl=64 time=2.797 ms
```

VERIFIKASI KONEKSI ANTAR VPCS

- Hasil verifikasi koneksi dari **VPCS MKT1** ke **SALES1** dengan mengeksekusi perintah "**ping 192.168.3.2**".

```
MKT1> ping 192.168.3.2
84 bytes from 192.168.3.2 icmp_seq=1 ttl=63 time=31.847 ms
84 bytes from 192.168.3.2 icmp_seq=2 ttl=63 time=32.973 ms
84 bytes from 192.168.3.2 icmp_seq=3 ttl=63 time=32.117 ms
84 bytes from 192.168.3.2 icmp_seq=4 ttl=63 time=32.599 ms
84 bytes from 192.168.3.2 icmp_seq=5 ttl=63 time=32.901 ms
```

- Hasil verifikasi koneksi dari **VPCS MKT1** ke **SALES2** dengan mengeksekusi perintah "**ping 192.168.3.3**".

```
MKT1> ping 192.168.3.3
84 bytes from 192.168.3.3 icmp_seq=1 ttl=63 time=31.834 ms
84 bytes from 192.168.3.3 icmp_seq=2 ttl=63 time=31.095 ms
84 bytes from 192.168.3.3 icmp_seq=3 ttl=63 time=31.858 ms
84 bytes from 192.168.3.3 icmp_seq=4 ttl=63 time=31.251 ms
84 bytes from 192.168.3.3 icmp_seq=5 ttl=63 time=32.298 ms
```

VERIFIKASI KONEKSI ANTAR VPCS

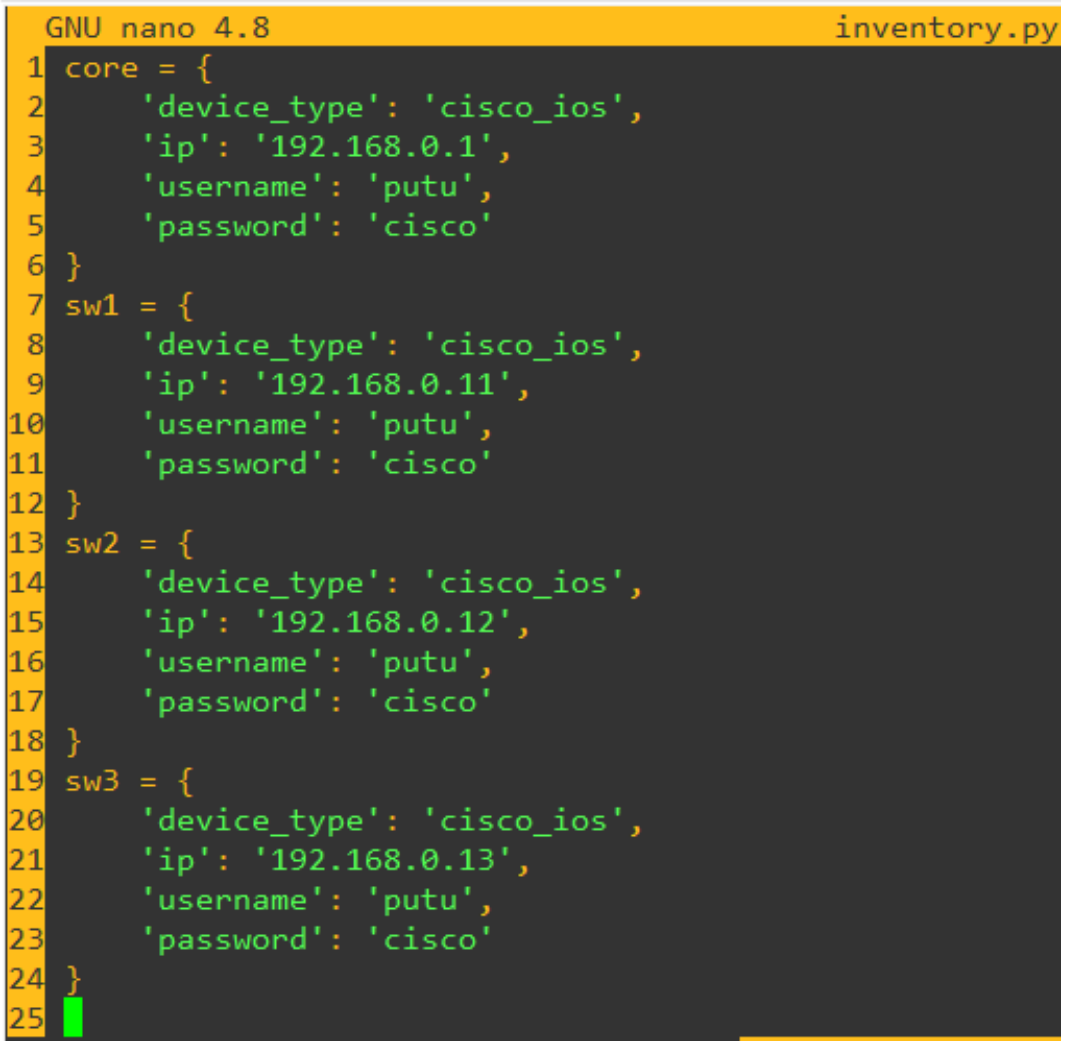
- Hasil verifikasi koneksi dari **VPCS MKT1** ke **SALES3** dengan mengeksekusi perintah "**ping 192.168.3.4**".

```
MKT1> ping 192.168.3.4
84 bytes from 192.168.3.4 icmp_seq=1 ttl=63 time=31.385 ms
84 bytes from 192.168.3.4 icmp_seq=2 ttl=63 time=32.760 ms
84 bytes from 192.168.3.4 icmp_seq=3 ttl=63 time=31.421 ms
84 bytes from 192.168.3.4 icmp_seq=4 ttl=63 time=31.893 ms
84 bytes from 192.168.3.4 icmp_seq=5 ttl=63 time=32.410 ms
```

Terlihat koneksi berhasil dilakukan.

STUDI KASUS BACKUP KONFIGURASI PERANGKAT

- Membuat file dengan nama "**inventory.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano inventory.py`
- Membuat kode program yang memuat deklarasi **variable** bertipe data **dictionary** yang mendefinisikan informasi perangkat meliputi **device_type**, alamat **IP** atau **hostname** dan **username** serta **password**. Terdapat 4 (empat) *variable* yang dibuat yaitu **core**, **sw1**, **sw2**, dan **sw3** sebagai variable yang menampung data dari seluruh perangkat jaringan.
- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**inventory.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.



STUDI KASUS BACKUP KONFIGURASI PERANGKAT

- Membuat file dengan nama "**backup-multiple-devices.py**" menggunakan editor **nano** dengan mengeksekusi perintah: `# nano backup-multiple-devices.py`
- Membuat kode program untuk mem**backup** konfigurasi yang sedang aktif di seluruh perangkat.

```
 GNU nano 4.8                          backup-multiple-devices.py
1  from netmiko import ConnectHandler
2  import inventory
3  import time
4
5  for device in (inventory.core, inventory.sw1, inventory.sw2, inventory.sw3):
6      net_connect = ConnectHandler(**device)
7      hostname = net_connect.find_prompt()[:-1]
8      backuptime = time.strftime("%d-%m-%Y_%H-%M-%S")
9      backupfilename = f"{hostname}-{device['ip']}_{backuptime}"
10     output = net_connect.send_command("show running-config")
11     file = open(f"backup/{backupfilename}.cfg", "w")
12     file.write(output)
13     file.close()
14     net_connect.disconnect()
15     print(f"Backup config dari device {hostname} dengan IP {device['ip']}")
16     print(f"Sukses dibackup pada {backuptime}.")
17     print("*"*50)
18
```

STUDI KASUS BACKUP KONFIGURASI PERANGKAT

Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler factory function** dari **Netmiko**.
- Baris 2: Melakukan *import* **inventory** sehingga dapat mengakses data perangkat yang terdapat di *file* **inventory.py**.
- Baris 3: Melakukan *import* **time** sehingga dapat mengambil kapan waktu proses **backup** dilakukan meliputi tanggal dan jam, menit serta detik.
- Baris 5: Melakukan perulangan sejumlah *dictionary variable* yaitu **inventory.core**, **inventory.sw1**, **inventory.sw2**, **inventory.sw3** dan menyimpannya pada variable **device**.
- Baris 6: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****device**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 7: Menemukan **prompt CLI** dari perangkat jaringan menggunakan **method find_prompt()** dan mengambil informasi nilai dari **prompt CLI** tersebut hingga sebelum tanda **#** yang menjadi penanda dari **mode privilege mode**. Hasilnya disimpan pada *variable* **hostname**.

STUDI KASUS BACKUP KONFIGURASI PERANGKAT

- Baris 8: Mendeklarasikan *variable* dengan nama "**backuptime**" yang didalamnya memuat string dengan format penulisan **tanggal-bulan-tahun_jam-menit-detik**.
- Baris 9: Mendeklarasikan *variable* dengan nama "**backupfilename**" yang didalamnya memuat *string* dengan format penulisan **hostname-ip-backuptime**. Nilai **hostname** diambil dari *variable* **hostname**. Nilai IP diambil dari **dictionary device** dengan **key ip**. Sedangkan nilai **backuptime** diambil dari *variable* **backuptime**.
- Baris 10: Mengeksekusi perintah "**show running-config**" menggunakan **method send_command()** dan menyimpan hasilnya ke *variable* **output**.
- Baris 11: Membuat *file* baru yang disimpan di dalam direktori backup dengan nama yang diambil dari *variable* **backupfilename**. *File* dibuka dengan mode **write**.
- Baris 12: Menyimpan isi dari *variable* **output** ke *file*.
- Baris 13: Menutup *file* yang terbuka.
- Baris 14: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.

STUDI KASUS BACKUP KONFIGURASI PERANGKAT

- Baris 15: Menampilkan pesan yang memuat informasi terkait **backup** dilakukan pada perangkat dengan **hostname** dan **IP** berapa.

- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**backup-multiple-devices.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

- Eksekusi **file python script** menggunakan perintah:

```
# python3 backup-multiple-devices.py
```

STUDI KASUS BACKUP KONFIGURASI PERANGKAT

- Hasil dari eksekusi kode program tersebut memperlihatkan bahwa **backup** konfigurasi dari seluruh perangkat telah berhasil dilakukan.

```
root@NetworkAutomation-1:~# python3 backup-multiple-devices.py
Backup config dari device CORE dengan IP 192.168.0.1
Sukses dibackup pada 22-06-2021_23-11-40.
**********************************************

Backup config dari device SW1 dengan IP 192.168.0.11
Sukses dibackup pada 22-06-2021_23-11-47.
**********************************************

Backup config dari device SW2 dengan IP 192.168.0.12
Sukses dibackup pada 22-06-2021_23-11-53.
**********************************************

Backup config dari device SW3 dengan IP 192.168.0.13
Sukses dibackup pada 22-06-2021_23-12-00.
**********************************************
```

STUDI KASUS BACKUP KONFIGURASI PERANGKAT

- Eksekusi perintah "`ls backup`" untuk menampilkan isi dari directory **backup** yang menampung hasil dari keseluruhan file konfigurasi setiap perangkat yang dibackup.

```
root@NetworkAutomation-1:~# ls backup
CORE-192.168.0.1_22-06-2021_23-11-40.cfg  SW2-192.168.0.12_22-06-2021_23-11-53.cfg
SW1-192.168.0.11_22-06-2021_23-11-47.cfg  SW3-192.168.0.13_22-06-2021_23-12-00.cfg
root@NetworkAutomation-1:~#
```

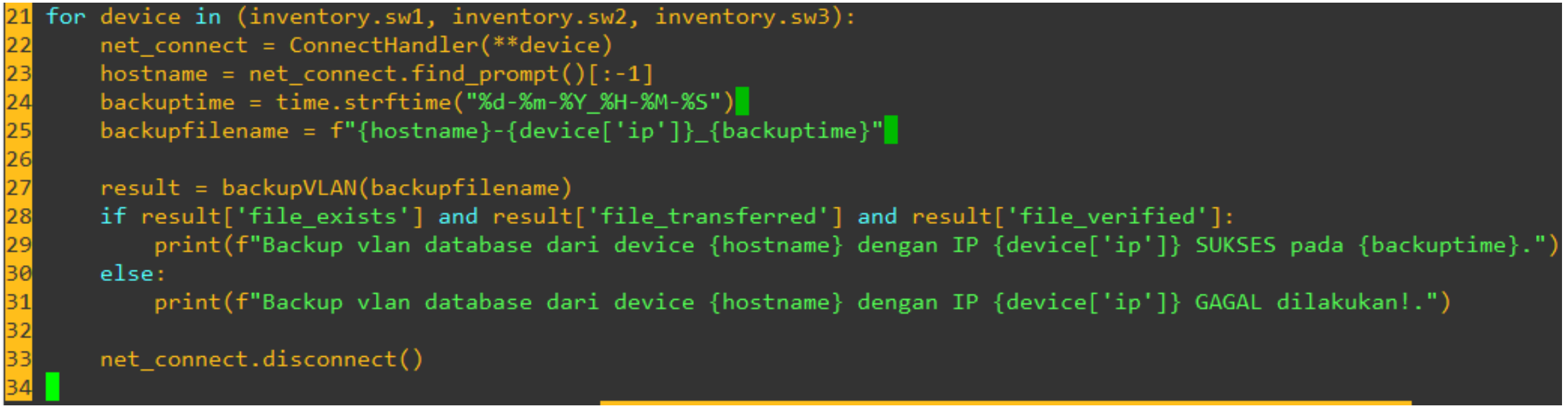
- Terlihat terdapat 4 (empat) **file backup** dengan ekstensi ***.cfg**.

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

- Membuat *file* dengan nama "**backup-vlan-db-multiple-devices.py**" menggunakan editor **nano** dengan mengeksekusi perintah:
  `# nano backup-vlan-db-multiple-devices.py`
- Membuat kode program untuk mem**backup VLAN Database** dari seluruh perangkat **Cisco Multilayer Switch** meliputi **SW1, SW2** dan **SW3**.

```
 GNU nano 4.8                                  backup-vlan-db-multiple-devices.py
1  from netmiko import ConnectHandler, file_transfer
2  import inventory
3  import time
4
5  def backupVLAN(backupfilename):
6      source_file = "vlan.dat"
7      dest_file = f"backup/{backupfilename}.vlan.dat"
8      direction = "get"
9      file_system = "flash:"
10
11     transfer_dict = file_transfer(
12        net_connect,
13        source_file=source_file,
14        dest_file=dest_file,
15        file_system=file_system,
16        direction=direction,
17        overwrite_file=True,
18     )
19     return transfer_dict
```

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH



```
21 for device in (inventory.sw1, inventory.sw2, inventory.sw3):
22     net_connect = ConnectHandler(**device)
23     hostname = net_connect.find_prompt()[:-1]
24     backuptime = time.strftime("%d-%m-%Y_%H-%M-%S")
25     backupfilename = f"{hostname}-{device['ip']}_{backuptime}"
26
27     result = backupVLAN(backupfilename)
28     if result['file_exists'] and result['file_transferred'] and result['file_verified']:
29         print(f"Backup vlan database dari device {hostname} dengan IP {device['ip']} SUKSES pada {backuptime}.")
30     else:
31         print(f"Backup vlan database dari device {hostname} dengan IP {device['ip']} GAGAL dilakukan!.")
32
33     net_connect.disconnect()
34
```

Penjelasan kode program:

- Baris 1: Melakukan *import* **ConnectHandler** dan **file_transfer** **factory function** dari **Netmiko**.
- Baris 2: Melakukan *import* **inventory** sehingga dapat mengakses data perangkat yang terdapat di *file* **inventory.py**.

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

- Baris 3: Melakukan *import* **time** sehingga dapat mengambil kapan waktu proses **backup** dilakukan meliputi tanggal dan jam, menit serta detik.
- Baris 5: Mendeklarasikan fungsi dengan nama "**backupVLAN**" dengan satu argumen yaitu "**backupfilename**".
- Baris 6: Mendeklarasikan *variable* dengan nama "**source_file**" yang memuat data berupa nama *file* sumber yang akan di*backup* yaitu "**vlan.dat**".
- Baris 7: Mendeklarasikan *variable* dengan nama "**dest_file**" yang memuat data berupa lokasi direktori yaitu "**backup**" dan nama *file* sebagai tujuan penyimpanan *file* yang di*backup*. Nama *file* tujuan penyimpanan menggunakan **prefix** berupa nilai dari argumen fungsi **backupVLAN** yaitu **backupfilename** yang disambung dengan "**.vlan.dat**". Sehingga **path** lengkapnya adalah "**backup/{backupfilename}.vlan.dat**".
- Baris 8: Mendeklarasikan *variable* dengan nama "**direction**" yang memuat data berupa operasi yang dilakukan yaitu mengunduh dengan nilai "**get**".
- Baris 9: Mendeklarasikan *variable* dengan nama "**file_system**" yang memuat data berupa *file system* yaitu "**flash:**".

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

- Baris 11-18: Mentransfer *file* **vlan.dat** dari perangkat **Cisco Multilayer Switch** ke **Network Automation node** menggunakan **Secure Copy (SCP)** melalui pemanggilan **file_transfer**.

  Terdapat beberapa parameter atau **argument** yang dimiliki oleh **file_transfer** meliputi **net_connect**, **source_file**, **dest_file**, **file_system**, dan **direction**, serta **overwrite_file**. Parameter **net_connect** merupakan nama *variable* yang memuat hasil pembentukan koneksi SSH ke perangkat. Paramater **source_file** memuat nama *variable* yang menyimpan nama *file* sumber yang akan ditransfer atau di**backup** yaitu **source_file**. Parameter **dest_file** memuat nama *variable* yang menyimpan lokasi direktori dan nama file tujuan penyimpanan dari *file* yang di**backup** yaitu **dest_file**. Parameter **file_system** memuat nama *variable* yang menyimpan *file system* yaitu **file_system**. Sedangkan parameter **overwrite_file** bernilai **True** berfungsi untuk menimpa (mengganti) *file* tujuan jika sudah ada. Hasil dari pemanggilan **file_transfer** disimpan pada *variable* **transfer_dict**.

- Baris 19: *keyword* **return** berfungsi untuk mengembalikan nilai dari fungsi ketika dipanggil.

- Baris 21: Melakukan perulangan sejumlah *dictionary variable* yaitu **inventory.sw1**, **inventory.sw2**, **inventory.sw3** dan menyimpannya pada variable **device**.

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

NETMIKO

- Baris 22: Membentuk koneksi SSH dengan memanggil **ConnectHandler** yang memuat *keyworded argument* ****device**. Hasil dari pembentukan koneksi SSH disimpan pada *variable* **net_connect**.
- Baris 23: Menemukan **prompt CLI** dari perangkat jaringan menggunakan **method find_prompt()** dan mengambil informasi nilai dari **prompt CLI** tersebut hingga sebelum tanda **#** yang menjadi penanda dari **mode privilege mode**. Hasilnya disimpan pada *variable* **hostname**.
- Baris 24: Mendeklarasikan *variable* dengan nama "**backuptime**" yang didalamnya memuat string dengan format penulisan **tanggal-bulan-tahun_jam-menit-detik**.
- Baris 25: Mendeklarasikan *variable* dengan nama "**backupfilename**" yang didalamnya memuat *string* dengan format penulisan **hostname-ip-backuptime**. Nilai **hostname** diambil dari *variable* **hostname**. Nilai IP diambil dari **dictionary device** dengan **key ip**. Sedangkan nilai **backuptime** diambil dari *variable* **backuptime**.
- Baris 27: Memanggil fungsi **backupVLAN** dengan argumen **backupfilename** dan menyimpan hasilnya ke *variable* **result**.

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

- Baris 28-31: Menggunakan kendali program **if…else** untuk mengecek nilai balik dari pemanggilan fungsi **backupVLAN** yang berupa **dictionary** dengan struktur sebagai berikut:

```
{
    'file_exists': True,
    'file_transferred': True,
    'file_verified': True
}
```

File_exists mengindikasikan apakah *file* yang ditransfer telah berada di sistem lokal untuk operasi **get**. **File_transferred** mengindikasikan apakah file perlu ditransfer atau sudah benar. **File_verified** mengindikasikan apakah *file* telah melalui pemeriksaan MD5. Apabila ketiga **key** pada *dictionary* tersebut bernilai **True** maka **backup file VLAN database** berhasil (**sukses**) dilakukan. Sebaliknya akan menampilkan pesan terkait **backup file VLAN database** yang tidak berhasil (**gagal**) dilakukan.

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

- Baris 33: Memutuskan koneksi SSH dari perangkat jaringan menggunakan **method disconnect()**.
- Tekan **CTRL+O** dan tekan **Enter** untuk menyimpan perubahan pada **file** "**backup-vlan-db-multiple-devices.py**".
- Tekan **CTRL+X** untuk keluar dari **editor nano**.

- Eksekusi **file python script** menggunakan perintah:

```
# python3 backup-vlan-db-multiple-devices.py
```

- Hasil dari eksekusi kode program tersebut memperlihatkan bahwa **backup VLAN database** dari seluruh perangkat **Cisco Multilayer Switch** telah berhasil dilakukan.

```
root@NetworkAutomation-1:~# python3 backup-vlan-db-multiple-devices.py
Backup vlan database dari device SW1 dengan IP 192.168.0.11 SUKSES pada 28-06-2021_21-39-42.
Backup vlan database dari device SW2 dengan IP 192.168.0.12 SUKSES pada 28-06-2021_21-39-47.
Backup vlan database dari device SW3 dengan IP 192.168.0.13 SUKSES pada 28-06-2021_21-39-52.
root@NetworkAutomation-1:~#
```

STUDI KASUS BACKUP VLAN DATABASE DARI PERANGKAT MULTILAYER SWITCH

- Hasil verifikasi isi dari direktori **backup** dengan mengeksekusi perintah "**ls backup**" telah memperlihatkan *file* **vlan.dat** dari **SW1**, **SW2** dan **SW3**.

```
root@NetworkAutomation-1:~# ls backup
CORE-192.168.0.1_28-06-2021_21-38-38.cfg        SW2-192.168.0.12_28-06-2021_21-39-47.vlan.dat
SW1-192.168.0.11_28-06-2021_21-38-44.cfg        SW3-192.168.0.13_28-06-2021_21-38-54.cfg
SW1-192.168.0.11_28-06-2021_21-39-42.vlan.dat   SW3-192.168.0.13_28-06-2021_21-39-52.vlan.dat
SW2-192.168.0.12_28-06-2021_21-38-49.cfg
root@NetworkAutomation-1:~#
```

COMMONLY-USED NETMIKO METHODS

Netmiko Methods	Deskripsi
net_connect.find_prompt()	Mengambil informasi *prompt* perangkat saat ini.
net_connect.disconnect()	Menutup koneksi SSH.
net_connect.send_command(arguments)	Mengirimkan perintah ke SSH dan mengembalikan nilai berupa *output* eksekusi perintah.
net_connect.send_config_set(arguments)	Mengirimkan sekumpulan perintah konfigurasi ke perangkat.
net_connect.send_config_from_file(arguments)	Mengirimkan sekumpulan perintah konfigurasi yang bersumber dari *file* ke perangkat.
net_connect.save_config()	Menyimpan konfigurasi secara permanen pada perangkat.

ADA PERTANYAAN?

REFERENSI

- Cisco Networking Academy, Enterprise Networking, Security, and Automation v7.0 (ENSA)
- Network Programmability Basics, https://www.networkcomputing.com/networking/network-programmability-basics
- Beginner's Guide to Python, https://wiki.python.org/moin/BeginnersGuide
- W3Schools, Python Introduction, https://www.w3schools.com/python/python_intro.asp
- Paramiko Website, http://www.paramiko.org/
- NAPALM Website, https://napalm.readthedocs.io/en/latest/
- Netmiko Website, https://ktbyers.github.io/netmiko/
- Kirk Byers, Netmiko Library, https://pynet.twb-tech.com/blog/automation/netmiko.html

TERIMAKASIH