

# Laporan Mini Project

Deny Lukman Syarif

## Latar Belakang :

Jaringan komputer adalah jaringan yang memungkinkan antar komputer untuk saling berkomunikasi dengan bertukar data satu sama lain. Untuk memungkinkan komunikasi antar komputer ini bisa terjadi, perlu dilakukan konfigurasi pada hardware dan software. Mulai dari konfigurasi kabel, ip addressing, vlan, subnetting, dll. Konfigurasi ini bisa dilakukan secara manual satu per satu pada setiap *device*. Namun sekarang kita sudah memasuki era *automation* atau sering disebut *Network Programmability*. Sekarang kita dapat mengkonfigurasi *device* hanya dengan beberapa tahap, bahkan bisa hanya dengan menjalankan satu *script* yang kita buat untuk mengkonfigurasi semua *device*.

Cisco developer memberikan fasilitas kepada kita untuk belajar mengenai jaringan komputer. Salah satu layanan yang ada pada cisco developer adalah **sandbox**. Sandbox menyediakan banyak fasilitas di dalamnya dari networking, data center, cloud, security, IoT, dll. Yang dapat kita gunakan secara gratis dengan cara melakukan reservasi. Pada sanbox ini menerapkan konsep *hands on lab*, kita dapat menggunakannya untuk belajar seperti mengkonfigurasi router, switch, membangun simulasi small network, dll.

Cisco Modelling Labs (CML) adalah salah satu layanan yang terdapat pada sandox. CML ini mirip dengan packet tracer namun lebih powerfull. CML pada sandbox terdapat 2 jenis, CML dan CML enterprise. Perbedaannya adalah durasi reservasi yang diberikan, untuk CML hanya 4 jam maksimal, sedangkan CML enterprise maksimal 2 hari. Sandbox menggunakan koneksi vpn untuk akses *client*-nya. Di dalam CML ini kita dapat mendesain, simulasi, testing network, dll. Namun pada CML ini sering terjadi *disconnect*, sehingga tidak dapat menjalankan fitur-fiturnya secara maksimal seperti menjalankan *console*, *info real-time*, dll.

Berdasarkan uraian di atas, saya mencoba membuat program menggunakan python untuk mengatasi masalah tersebut dengan memanfaatkan CML API. Dan mencoba membuat small network pada CML menggunakan script dan mengkonfigurasi perangkatnya dengan RESTCONF untuk router dan NXAPI untuk switch.

## Rancangan Umum Project :

- Membuat program interaktif menggunakan python untuk memantau labs pada CML (informasi labs, menjalankan lab, menghentikan lab, membuat lab, menghapus lab, mendownload topology lab, membuat node/menambahkan device, menghapus node/device, update konfigurasi perangkat/node, menghidupkan interface device, membuat link antar device)
- Membuat script automation untuk membangun topology small network
- Mengkonfigurasi router csr1000v dengan RESTCONF
- Mengkonfigurasi switch nxosv9000 dengan NXAPI
- Mengakses perangkat pada CML lewat terminal local dengan breakout tools pada CML (melalui telnet localhost port)
- Menggunakan ansible untuk backup running-configuration router dan switch pada small network tersebut

## Penjelasan Project :

### A. Membuat Program Interaktif CML

#### 1. Membuat utilities module (utilities.py)

Module ini digunakan untuk utilities clear output dan generate url api.  
from os import system, name

```

from config import *
import pprint
pp = pprint.PrettyPrinter(indent=4, sort_dicts=False)

def clear():
    if name == "posix":
        system("clear")
    else:
        system("cls")

def url(endpoint: str):
    URL = BASE_URL + endpoint
    return URL

```

## 2. Membuat Config Module (config.py)

Module ini digunakan untuk inisialisasi variable seperti host ip CML, token JWT, base url API CML.

```

import requests as req
import json
req.packages.urllib3.disable_warnings()

headers = {
    'Accept': "application/json",
    'Content-Type': "application/json"
}

url = "https://10.10.20.161/api/v0/authenticate"
data = json.dumps({
    'username': 'developer',
    'password': 'C1sco12345'
})
resp_token = req.post(url, headers=headers, data=data, verify=False)

HOST = "https://10.10.20.161"
BASE_URL = "{} /api/v0".format(HOST)
token = resp_token.json()

headers = {
    "Accept": "application/json",
    "Authorization": "Bearer {}".format(token)
}

```

## 3. Membuat Labs Module (Labs.py)

Module ini digunakan untuk meng-*handle* lab-lab pada CML, terdapat:

- ◆ function showDetailLab(labs): untuk menampilkan informasi lab-lab ke layar dari parameter labs
- ◆ function getLabs(): untuk men-get informasi dari CML menggunakan api /labs

- ◆ function getIdLab(process): untuk mengambil id lab, parameter process ini berupa string untuk generate kalimat sesuai proses yang dipilih, misalnya create, hapus, start.
- ◆ function startLab(): untuk menjalankan lab
- ◆ function stopLab(): untuk menghentikan lab
- ◆ function createLab(): untuk membuat lab
- ◆ function deleteLab(): untuk menghapus lab
- ◆ function getTopologyLab(): digunakan untuk men-get topology lab berdasarkan id lab, nantinya akan disimpan ke dalam format yaml dan json pada directory topology

```

from utilities import *
import json
import yaml
import time

def showDetailLab(labs):
    clear()
    print("="*30)
    print("no | \t \tdetails")
    print("="*30)
    for i in range(len(labs)):
        print("{} ".format(str(i+1)), end="")
        pp.pprint(labs[i])

def getLabs():
    # get all id of labs
    URL = url('/labs')
    resp = req.get(URL, headers=headers, verify=False)

    labs = []

    # get detail of each lab from id
    for id_lab in resp.json():
        #URL = "{} /labs/{}".format(BASE_URL, id_lab)
        endpoint = "/labs/" + id_lab
        URL = url(endpoint)
        lab = req.get(URL, headers=headers, verify=False)
        labs.append(lab.json())

    # show the detail lab in screen
    return labs

def getIdLab(process: str):
    labs = getLabs()
    check = False
    lab_no = 0
    while check == False:
        lab_no = input("{} lab no : ".format(process))
        if lab_no.isdigit():

```

```
lab_no = int(lab_no)
if lab_no > 0 and lab_no <= len(labs):
    check = True
else:
    print("only input lab no that show in screen!")
else:
    print("only input lab no that show in screen!")
```

```
id_labs = labs[lab_no-1]['id']
return id_labs
```

```
def startLab():
    id_labs = getIdLab('start')

    endpoint = "/labs/{}/start".format(id_labs)
    URL = url(endpoint)

    print("Starting Lab with id {} ...".format(id_labs))
    resp = req.put(URL, headers=headers, verify=False)
    print("status: "+ resp.json())
```

```
def stopLab():
    id_labs = getIdLab('stop')

    endpoint = "/labs/{}/stop".format(id_labs)
    URL = url(endpoint)

    print("Stoping Lab with id {} ...".format(id_labs))
    resp = req.put(URL, headers=headers, verify=False)
    print("status: "+ resp.json())
```

```
def createLab():
    title = input("lab title : ")

    endpoint = "/labs?title={}".format(title)
    URL = url(endpoint)
    print("Creating Lab with title '{}' ...".format(title))
    resp = req.post(URL, headers=headers, verify=False)
    print(resp.text)
```

```
def deleteLab():
    id_labs = getIdLab('delete')

    endpoint = "/labs/{}/wipe".format(id_labs)
    URLwipe = url(endpoint)
```

```

print("Deleting Lab id {}".format(id_labs))

endpoint = "/labs/{}".format(id_labs)
URL = url(endpoint)
ask = input("Really want to delete lab {} ? (y/n): ".format(id_labs))
if ask == 'y' or ask == '':
    req.put(URL, wipe, headers=headers, verify=False)
    resp = req.delete(URL, headers=headers, verify=False)
    print(resp)
else:
    print("cancelling...")
    time.sleep(1)
    return

```

```

def getTopologyLab():
    id_labs = getIdLab('get topology')

    endpoint = "/labs/{}/topology".format(id_labs)
    URL = url(endpoint)

    print("Processing get topology {}".format(id_labs))

    headers['Accept'] = 'text/plain'
    resp = req.get(URL, headers=headers, verify=False)
    ask = input('do u want to save the topology to the file? (y/n): ')
    if ask == 'y' or ask == '':
        location = './topology/lab_{}.json'.format(id_labs)
        f = open(location, 'w')
        f.write(resp.text)
        f.close()

        with open(location, 'r') as jsonfile:
            jsonread = json.load(jsonfile)

        location2 = './topology/lab_{}.yaml'.format(id_labs)
        f = open(location2, 'w')
        f.write(yaml.dump(jsonread))
        f.close()

        print("Success save topology in {} and {}".format(location, location2))
        time.sleep(2)
    elif ask == 'n':
        pp.pprint(resp.json())

```

#### 4. Membuat Nodes Module (Nodes.py)

Module ini digunakan untuk meng-*handle* node-node pada lab, terdapat:

- ◆ function createNode(): untuk membuat node/menambahkan perangkat pada lab
- ◆ function updateConfigNodes(): untuk mengupdate konfigurasi perangkat melalui path file config yang diinputkan

- ◆ function deleteNode(): untuk menghapus node/perangkat pada lab

```
from config import *
from utilities import *
from Labs import getIdLab
import json

def createNode():
    id_labs = getIdLab('create node in')
    print("id lab: {}".format(id_labs))
    endpoint = "/labs/{}/nodes".format(id_labs)
    URL = url(endpoint)
    x = input("x: ")
    y = input("y: ")
    label = input('label: ')
    configuration = input('configuration: ')
    node_definition = input("node_definition: ")
    image_definition = input("image_definition: ")

    data = json.dumps({
        "x": int(x),
        "y": int(y),
        "label": label,
        "configuration": configuration,
        "node_definition": node_definition,
        "image_definition": image_definition,
        "ram": None,
        "cpus": None,
        "cpu_limit": None,
        "data_volume": None,
        "boot_disk_size": None,
        "tags": [
            ""
        ]
    })

    print("Creating node in lab with id {}".format(id_labs))
    resp = req.post(URL, headers=headers, data=data, verify=False)
    print(resp.text)

def updateConfigNodes():
    id_labs = getIdLab('update configuration node in')
    print("id lab: {}".format(id_labs))
    node = input("node: ")
    config = input("config path file: ")

    f = open(config, 'r')
    payload = f.read()
    f.close()
```

```

endpoint = "/labs/{}/nodes/{}/wipe_disks".format(id_labs, node)
URL = url(endpoint)
req.put(URL, headers=headers, verify=False)

endpoint = "/labs/{}/nodes/{}/config".format(id_labs, node)
URL = url(endpoint)

headers['Content-Type'] = "text/plain"
resp = req.put(URL, headers=headers, data=payload, verify=False)
print(resp.text)

```

```

def deleteNode():
    id_labs = getIdLab('delete node in')
    print("id lab: {}".format(id_labs))
    node = input("node: ")

    endpoint = "/labs/{}/nodes/{}/wipe_disks".format(id_labs, node)
    URL = url(endpoint)
    req.put(URL, headers=headers, verify=False)

    endpoint = "/labs/{}/nodes/{}".format(id_labs, node)
    URL = url(endpoint)

    resp = req.delete(URL, headers=headers, verify=False)
    print(resp.text)

```

## 5. Membuat Interfaces Module (Interface.py)

Module ini digunakan untuk membuat/mengaktifkan interface/port pada node. User akan diminta untuk menginputkan node yang mana dan pada slot berapa.

```

from config import *
from utilities import *
from Labs import getIdLab
import json

def createInterfaces():
    id_labs = getIdLab('create interface on node in')
    print("id lab: {}".format(id_labs))
    node = input("node: ")
    slot = input("slot: ")

    endpoint = "/labs/{}/nodes/{}/wipe_disks".format(id_labs, node)
    URL = url(endpoint)
    req.put(URL, headers=headers, verify=False)

    endpoint = "/labs/{}/interfaces".format(id_labs)
    URL = url(endpoint)

    data = json.dumps({

```

```

        'node': node,
        'slot': int(slot)
    })

```

```

print("Creating interface on node {} in lab with id {}".format(node,id_labs))
resp = req.post(URL, headers=headers, data=data, verify=False)
print(resp.text)

```

## 6. Membuat Links Module (Links.py)

Module ini digunakan untuk membuat link antar node, dengan menginputkan source interface id dan destination interface id

```

from config import *
from utilities import *
from Labs import getIdLab
import json

def createLinks():
    id_labs = getIdLab('create link in')
    print("id lab: {}".format(id_labs))
    endpoint = "/labs/{}/links".format(id_labs)
    URL = url(endpoint)
    src_int = input("src_int: ")
    dst_int = input("dst_int: ")

    data = json.dumps({
        'src_int': src_int,
        'dst_int': dst_int
    })

    print("Creating links in lab with id {}".format(id_labs))
    resp = req.post(URL, headers=headers, data=data, verify=False)
    print(resp.text)

```

## 7. Membuat Main Program (manage\_cml.py)

Program ini digunakan untuk menyatukan module-module di atas agar dapat dijalankan secara interaktif dan realtime oleh user

```

import time
import json
from utilities import *

import Labs
import Nodes
import Interfaces
import Links

```



```

def main():
    intro = """
=====
Menu:
=====
0. exit
1. start lab
2. stop lab
3. create lab
4. delete lab
5. get topology lab

6. create node
7. delete node
8. update config node

9. create interfaces

10. create links
=====
"""
    print(intro, end="")
    cek = False
    while(cek == False):
        choose = input("choose (1-9) / 0 for exit: ")
        if choose == '0':
            cek = True
            print("Goodbye!!!")
            exit()
        if choose == '1':
            Labs.startLab()
            cek = True
        elif choose == '2':
            Labs.stopLab()
            cek = True
        elif choose == '3':
            Labs.createLab()
            cek = True
        elif choose == '4':
            Labs.deleteLab()
            cek = True
        elif choose == '5':
            Labs.getTopologyLab()
            cek = True
        elif choose == '6':
            Nodes.createNode()
            cek = True
        elif choose == '7':
            Nodes.deleteNode()

```

```
    cek = True
elif choose == '8':
    Nodes.updateConfigNodes()
    cek = True
elif choose == '9':
    Interfaces.createInterfaces()
    cek = True
elif choose == '10':
    Links.createLinks()
    cek = True
elif choose == 'refresh':
    return
```

```
time.sleep(5)
clear()
while True:
    print("Getting Labs...")
    labs = Labs.getLabs()
    Labs.showDetailLab(labs)
    main()
```

## B. Membuat Automation Build Small Network pada CML

### 1. Membuat Design Small Network

bridge = n0

unmanaged switch = n1

R1 = n2

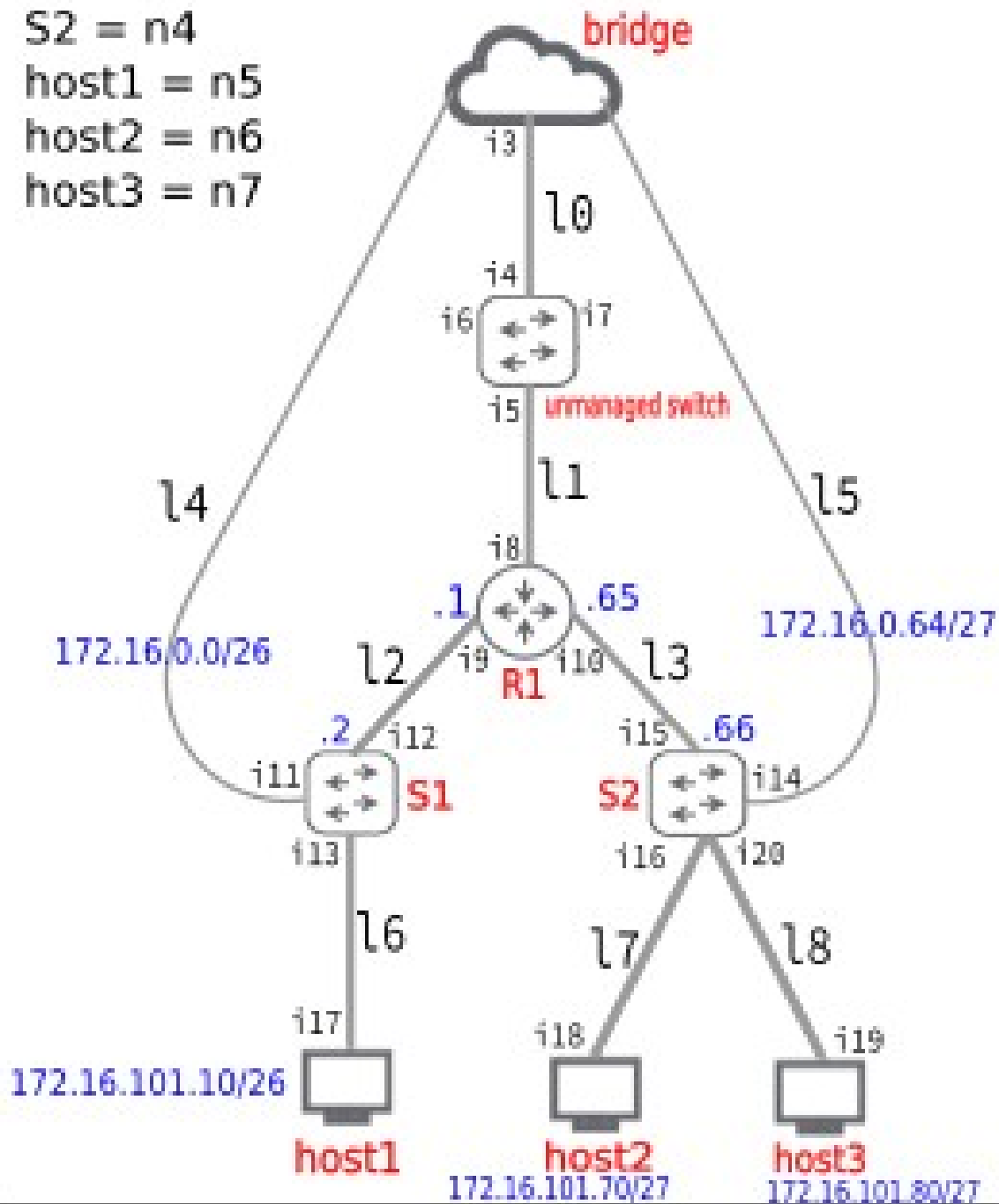
S1 = n3

S2 = n4

host1 = n5

host2 = n6

host3 = n7



## 2. Membuat Automation Module (automation.py)

Module ini digunakan untuk automation build network pada CML. Function-function nya menggunakan function dari modul sebelumnya ( function createLab(), createNode(), updateConfigNodes(), createInterface(), dan createLinks() ) dengan memodifikasi beberapa bagian dan parameter, karena sebelumnya untuk program interaktif.

```
from config import *
from utilities import *
import json
import yaml
import time

# LABS
def createLab(getTitle: str):
    title = getTitle

    endpoint = "/labs?title={}".format(title)
    URL = url(endpoint)
    print("Creating Lab with title '{}' ...".format(title))
    resp = req.post(URL, headers=headers, verify=False)
    idlab = resp.json()
    return idlab['id']

# NODES
def createNode(getId, data):
    id_labs = getId
    print("id lab: {}".format(id_labs))
    endpoint = "/labs/{}/nodes".format(id_labs)
    URL = url(endpoint)

    payload = json.dumps({
        "x": data['x'],
        "y": data['y'],
        "label": data['label'],
        "configuration": data['configuration'],
        "node_definition": data['node_definition'],
        "image_definition": data['image_definition'],
        "ram": None,
        "cpus": None,
        "cpu_limit": None,
        "data_volume": None,
        "boot_disk_size": None,
        "tags": [
            ""
        ]
    })

    print("Creating node in lab with id {}".format(id_labs))
    resp = req.post(URL, headers=headers, data=payload, verify=False)
```

```

return resp.text

def updateConfigNodes(getId, getNode, fileConfig):
    id_labs = getId
    print("id lab: {}".format(id_labs))
    node = getNode
    config = fileConfig

    f = open(config, 'r')
    payload = f.read()
    f.close()

    endpoint = "/labs/{}/nodes/{}/wipe_disks".format(id_labs, node)
    URL = url(endpoint)
    req.put(URL, headers=headers, verify=False)

    endpoint = "/labs/{}/nodes/{}/config".format(id_labs, node)
    URL = url(endpoint)

    headers['Content-Type'] = "text/plain"
    resp = req.put(URL, headers=headers, data=payload, verify=False)
    return resp.text

# INTERFACES
def createInterfaces(getId, data):
    id_labs = getId
    print("id lab: {}".format(id_labs))

    endpoint = "/labs/{}/nodes/{}/wipe_disks".format(id_labs, data['node'])
    URL = url(endpoint)
    req.put(URL, headers=headers, verify=False)

    endpoint = "/labs/{}/interfaces".format(id_labs)
    URL = url(endpoint)

    payload = json.dumps({
        'node': data['node'],
        'slot': data['slot']
    })

    print("Creating interface on node {} in lab with id {}...".format(data['node'], id_labs))
    resp = req.post(URL, headers=headers, data=payload, verify=False)
    return resp.text

# LINKS
def createLinks(getId, data):

```

```

id_labs = getId
print("id lab: {}".format(id_labs))
endpoint = "/labs/{}/links".format(id_labs)
URL = url(endpoint)

payload = json.dumps({
    'src_int': data['src_int'],
    'dst_int': data['dst_int']
})

print("Creating links in lab with id {}".format(id_labs))
resp = req.post(URL, headers=headers, data=payload, verify=False)
return resp.text

```

### 3. Membuat Script Automation Build Lab (script-automation-lab.py)

membuat script automation dari design network pada step 1. Pada script ini meng-import module automation yang dibuat pada step 2.

from automation import \*

id\_lab = createLab('project\_lab')

```

nodes = [
    {
        'x': 0,
        'y': 0,
        'label': 'bridge',
        'configuration': "./script-config/config-external-connector.txt",
        'node_definition': 'external_connector',
        'image_definition': 'null'
    },
    {
        'x': 0,
        'y': 200,
        'label': 'unmanaged switch',
        'configuration': "",
        'node_definition': 'unmanaged_switch',
        'image_definition': 'null'
    },
    {
        'x': 0,
        'y': 400,
        'label': 'R1',
        'configuration': "",
        'node_definition': 'csr1000v',
        'image_definition': 'csr1000vb'
    },
    {
        'x': -400,
        'y': 700,

```

```

    'label': 'S1',
    'configuration': '',
    'node_definition': 'nxosv9000',
    'image_definition': 'nxosv9000'
},
{
    'x': 400,
    'y': 700,
    'label': 'S2',
    'configuration': '',
    'node_definition': 'nxosv9000',
    'image_definition': 'nxosv9000'
},
{
    'x': -400,
    'y': 900,
    'label': 'host1',
    'configuration': './script-config/config-host1.txt',
    'node_definition': 'desktop',
    'image_definition': 'desktop-3-13-2-xfce'
},
{
    'x': 200,
    'y': 900,
    'label': 'host2',
    'configuration': './script-config/config-host2.txt',
    'node_definition': 'desktop',
    'image_definition': 'desktop-3-13-2-xfce'
},
{
    'x': 600,
    'y': 900,
    'label': 'host3',
    'configuration': './script-config/config-host3.txt',
    'node_definition': 'desktop',
    'image_definition': 'desktop-3-13-2-xfce'
}
]

```

```

for node in nodes:
    print(createNode(id_lab, node))

```

```

mynode = {
    'bridge': 'n0',
    'uswitch': 'n1',
    'router': 'n2',

```

```

's1': 'n3',
's2': 'n4',
'host1': 'n5',
'host2': 'n6',
'host3': 'n7'
}

# update config host setting ip
for node in nodes:
    if node['label'] == 'bridge' or node['label'] == 'host1' or node['label'] == 'host2' or node['label'] ==
'host3':
        n = node['label']
        print(updateConfigNodes(id_lab, mynode[n], node['configuration']))

interfaces = [
    {
        'node': mynode['bridge'],
        'slot': 0
    },
    {
        'node': mynode['uswitch'],
        'slot': 0
    },
    {
        'node': mynode['uswitch'],
        'slot': 1
    },
    {
        'node': mynode['uswitch'],
        'slot': 2
    },
    {
        'node': mynode['uswitch'],
        'slot': 3
    },
    {
        'node': mynode['router'],
        'slot': 0
    },
    {
        'node': mynode['router'],
        'slot': 1
    },
    {
        'node': mynode['router'],
        'slot': 2
    },
    {
        'node': mynode['s1'],

```



```

        'slot': 0
    },
    {
        'node': mynode['s1'],
        'slot': 1
    },
    {
        'node': mynode['s1'],
        'slot': 2
    },
    {
        'node': mynode['s2'],
        'slot': 0
    },
    {
        'node': mynode['s2'],
        'slot': 1
    },
    {
        'node': mynode['s2'],
        'slot': 2
    },
    {
        'node': mynode['host1'],
        'slot': 0
    },
    {
        'node': mynode['host2'],
        'slot': 0
    },
    {
        'node': mynode['host3'],
        'slot': 0
    },
    {
        'node': mynode['s2'],
        'slot': 3
    },
]

```

```

for interface in interfaces:
    print(createInterfaces(id_lab, interface))

```

```

links = [
    {
        'src_int': 'i3',
        'dst_int': 'i4'
    },
    {

```

```

        'src_int': 'i5',
        'dst_int': 'i8'
    },
    {
        'src_int': 'i9',
        'dst_int': 'i12'
    },
    {
        'src_int': 'i10',
        'dst_int': 'i15'
    },
    {
        'src_int': 'i11',
        'dst_int': 'i6'
    },
    {
        'src_int': 'i14',
        'dst_int': 'i7'
    },
    {
        'src_int': 'i13',
        'dst_int': 'i17'
    },
    {
        'src_int': 'i16',
        'dst_int': 'i18'
    },
    {
        'src_int': 'i19',
        'dst_int': 'i20'
    }
]

```

for link in links:

    print(createLinks(id\_lab, link))

print("Done!!!")

untuk konfigurasi node external-connector, host1, host2, dan host3 terdapat scriptnya pada directory script-config:

<b>config-external-connector.txt</b>	<b>host1.txt</b>	<b>host2.txt</b>	<b>host3.txt</b>
bridge0	sudo ifconfig eth0 172.16.101.10 netmask 255.255.255.192 hostname host2 USERNAME=host1 PASSWORD=host1	sudo ifconfig eth0 172.16.101.70 netmask 255.255.255.224 hostname host2 USERNAME=host2 PASSWORD=host2	sudo ifconfig eth0 172.16.101.80 netmask 255.255.255.224 hostname host3 USERNAME=host3 PASSWORD=host3

### C. Membuat Script RESTCONF (./RESTCONF/config.py)

script ini digunakan untuk mengkonfigurasi router csr1000v (R1) menggunakan fitur restconf. Pada script ini berisi konfigurasi interface GigabitEthernet2 172.16.0.1/26 untuk subnet 1 dan GigabitEthernet3 172.16.0.65/27 untuk subnet 2. Lalu juga save running-config ke startup-config.

```
import requests
import json
requests.packages.urllib3.disable_warnings()

data_ip = ['172.16.0.1','172.16.0.65']
data_subnet = ['255.255.255.192','255.255.255.224']

host = "10.10.20.200"
username = "admin"
password = "Admin12345"

headers = {
'Accept': 'application/yang-data+json',
'Content-Type': 'application/yang-data+json'
}

auth = (username, password)

for i in range(2,4):
    url = "https://{}/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet{}".format(host,
str(i))

    payload = json.dumps({
    "ietf-interfaces:interface": {
        "name": "GigabitEthernet{}".format(str(i)),
        "description": "Configured by RESTCONF subnet {}".format(str(i-1)),
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": data_ip[i-2],
                    "netmask": data_subnet[i-2]
                }
            ]
        }
    }
    })

    response = requests.put(url, auth=auth, headers=headers, data=payload, verify=False)
    print(response)

url = "https://{}/restconf/operations/cisco-ia:save-config/".format(host)
resp = requests.post(url, auth=auth, headers=headers, verify=False)
print("save " + resp.text)
```

#### D. Membuat Script NXAPI (./NXAPI/config.py)

Script ini digunakan untuk mengkonfigurasi 2 switch nxos9000 (S1) dan (S2) menggunakan nxapi. Pada script ini berisi konfigurasi interface dan konfigurasi vlan. Disini vlan yang digunakan vlan 101 untuk interface yang terhubung dengan PC. Lalu juga save running-config ke startup-config.

```
import requests
import json
requests.packages.urllib3.disable_warnings()

host = ['10.10.20.201', '10.10.20.202']
ip_interfaces = ['172.16.0.2/26', '172.16.0.66/27']
ip_vlan = ['172.16.101.2/26', '172.16.101.66/27']

headers = {
    'Content-Type': 'application/json',
}
auth = ('admin','Admin12345')

for i in range(1, len(ip_interfaces) + 1):
    url = "https://{}/ins".format(host[i-1])
    commands = [
        "vlan 101 ;name prod",
        "interface vlan 101 ;description config by NXAPI prod ;no shutdown ;no ip redirects ;ip address",
        "{} ;".format(ip_vlan[i-1]),
        "interface ethernet 1/1 ;description config by NXAPI connect to R{} ;ip address {} ;no",
        "shut".format(i, ip_interfaces[i-1]),
        "interface ethernet 1/2 ;description config by NXAPI connect to host ;switchport ;switchport access",
        "vlan 101 ;spanning-tree port type edge ;no shutdown ;"
    ]

    for x in range(4):
        payload = json.dumps({
            "ins_api": {
                "version": "1.0",
                "type": "cli_conf",
                "chunk": "0",
                "sid": "1",
                "input": commands[x],
                "output_format": "json"
            }
        })
    response = requests.request("POST", url, auth=auth, headers=headers, data=payload, verify=False)
    print(response.json()['ins_api']['outputs']['output'])

# config interface S2 e1/3
url = "https://{}/ins".format(host[1])
commands = [
    "interface ethernet 1/3 ;description config by NXAPI connect to host ;switchport ;switchport access",
    "vlan 101 ;spanning-tree port type edge ;no shutdown ;",
]
```

```

"copy run start"
]
for i in range(2):
    payload = json.dumps({
        "ins_api": {
            "version": "1.0",
            "type": "cli_conf",
            "chunk": "0",
            "sid": "1",
            "input": commands[i],
            "output_format": "json"
        }
    })

    response = requests.post(url, auth=auth, headers=headers, data=payload, verify=False)
    print(response.json()['ins_api']['outputs']['output'])

```

#### **D. Ansible for Backup Running Configuration Router and Switch**

##### **1. Membuat Ansible Config (ansible.cfg)**

script untuk konfigurasi ansible yang akan digunakan

```

[defaults]
inventory=./my_hosts
host_key_checking=False
retry_files_enabled=False
deprecation_warnings=False

```

##### **2. Membuat host file (my\_hosts)**

file ini untuk inialisasi ip host, username, password, hostname, ansible network, ansible python interpreter, port

```

R1 ansible_user=admin ansible_password=Admin12345 ansible_host=10.10.20.200 ansible_port=22
ansible_network_os=cisco.ios.ios ansible_python_interpreter=/usr/bin/python3

```

```

S1 ansible_user=admin ansible_password=Admin12345 ansible_host=10.10.20.201 ansible_port=22
ansible_network_os=cisco.ios.ios ansible_python_interpreter=/usr/bin/python3

```

```

S2 ansible_user=admin ansible_password=Admin12345 ansible_host=10.10.20.202 ansible_port=22
ansible_network_os=cisco.ios.ios ansible_python_interpreter=/usr/bin/python3

```

##### **3. Membuat playbook (playbook.yml)**

playbook untuk backup running-configuration pada router dan switch pada lab

```

---
- name: AUTOMATIC BACKUP OF RUNNING CONFIG Project Lab R1
  hosts: R1
  gather_facts: false
  connection: ansible.netcommon.network_cli

  tasks:
  - name: DISPLAYING THE RUNNING-CONFIG

```

```

ios_command:
  commands:
    - show running-config
register: config

- name: SAVE OUTPUT TO ./backups/
  copy:
    content: "{{ config.stdout[0] }}"
    dest: "backups/backup_{{ inventory_hostname }}_project.txt"

- name: AUTOMATIC BACKUP OF RUNNING CONFIG Project Lab S1
  hosts: S1
  gather_facts: false
  connection: ansible.netcommon.network_cli

  tasks:
    - name: DISPLAYING THE RUNNING-CONFIG
      ios_command:
        commands:
          - show running-config
      register: config

    - name: SAVE OUTPUT TO ./backups/
      copy:
        content: "{{ config.stdout[0] }}"
        dest: "backups/backup_{{ inventory_hostname }}_project.txt"

- name: AUTOMATIC BACKUP OF RUNNING CONFIG Project Lab S2
  hosts: S2
  gather_facts: false
  connection: ansible.netcommon.network_cli

  tasks:
    - name: DISPLAYING THE RUNNING-CONFIG
      ios_command:
        commands:
          - show running-config
      register: config

    - name: SAVE OUTPUT TO ./backups/
      copy:
        content: "{{ config.stdout[0] }}"
        dest: "backups/backup_{{ inventory_hostname }}_project.txt"

```

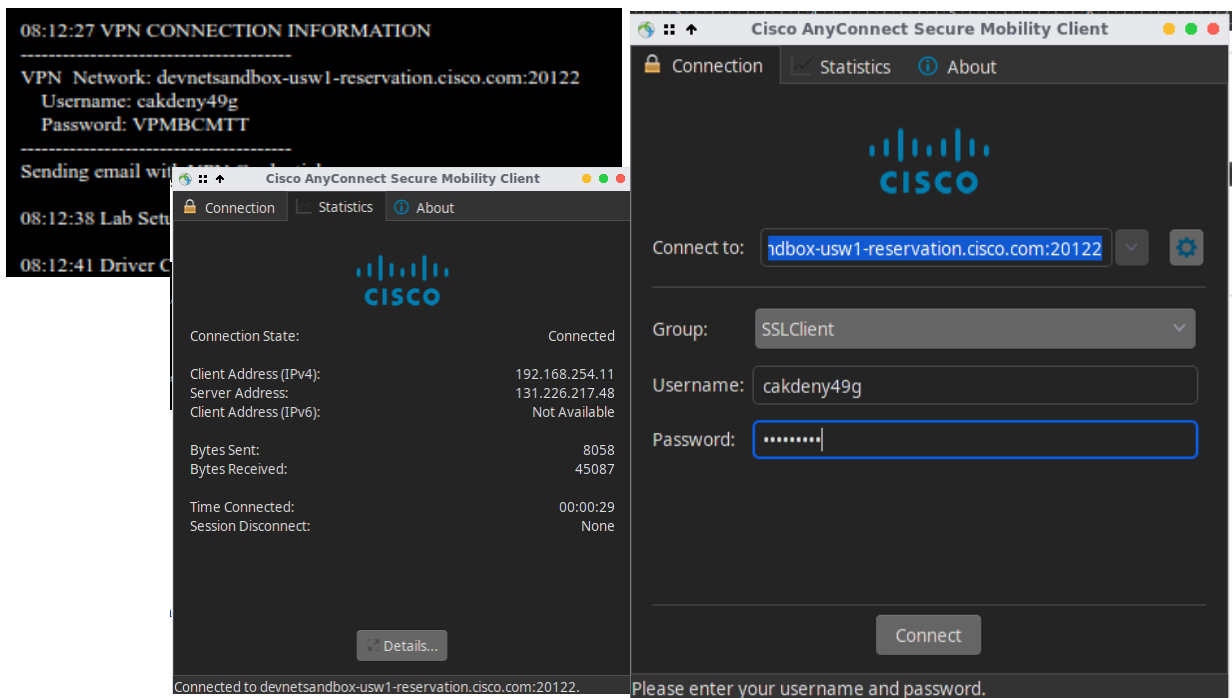
## E. Menyiapkan Breakout Tools

1. Mendownload *breakout-linux-x86\_amd64*, karena saya menggunakan os linux, bagi yang windows silahkan download yang windows version
2. menambahkan execute file permission pada user dengan ***chmod u+x breakout-linux-x86\_amd64***

3. menjalankan `./breakout-linux-x86_amd64 config` untuk membuat file config.yaml
4. menjalankan `./breakout-linux-x86_amd64 ui` untuk menjalankan breakout tool gui pada localhost:port sesuai pada config.yaml

#### F. Menjalankan Simulasi Build Automation dan Backup Automation

1. Membuat virtual environment agar package tidak mengganggu root package pada laptop dengan `python3 -m venv finalProject`
2. masuk ke directory dengan `cd finalProject`
3. aktivasi virtual environment dengan `source bin/activate`
4. install **requests** untuk kegunaan RESTFUL API dan **ansible** untuk kegunaan backup running configuration dengan ansible dengan `pip3 install requests && pip3 install ansible`
5. Connect ke VPN yang telah disiapkan oleh Cisco melau Cisco AnyConnect Secure Mobility dengan memasukkan url dan password



6. Menjalankan interaktif program `manage_cml.py` pada directory `src/CML/`

```
=====
no | details
=====
1. { 'state': 'STARTED',
    'created': '2021-07-27 12:49:04',
    'lab_title': 'Small NXOS/IOSXE Network',
    'lab_description': 'A sample network built with IOS XE, NX-OS, IOS XR, and
    'owner': 'developer',
    'node_count': 8,
    'link_count': 16,
    'id': 'b4bb2d'}

=====
Menu:
=====
0. exit
1. start lab
2. stop lab
3. create lab
4. delete lab
5. get topology lab

6. create node
7. delete node
8. update config node
9. create interfaces
10. create links

=====
choose (1-9) / 0 for exit:
```

7. Menjalankan **script-automation-lab.py** untuk membuat network sesuai skema sebelumnya

```
=====
no | details
=====
1. { 'state': 'STARTED',
    'created': '2021-07-27 12:49:04',
    'lab_title': 'Small NXOS/IOSXE Network',
    'lab_description': 'A sample network built with IOS XE, NX-OS, IOS XR, and
    'owner': 'developer',
    'node_count': 8,
    'link_count': 16,
    'id': 'b4bb2d'}

=====
Menu:
=====
0. exit
1. start lab
2. stop lab
3. create lab
4. delete lab
5. get topology lab

6. create node
7. delete node
8. update config node
9. create interfaces
10. create links

=====
choose (1-9) / 0 for exit:
```

8. cek pada interaktif **manage\_cml.py** dengan mengetikan **refresh** pada input untuk merefresh

```
=====
no | details
=====
1. { 'state': 'STARTED',
    'created': '2021-07-27 12:49:04',
    'lab_title': 'Small NXOS/IOSXE Network',
    'lab_description': 'A sample network built with IOS XE, NX-OS, IOS XR, and
    'owner': 'developer',
    'node_count': 8,
    'link_count': 16,
    'id': 'b4bb2d'}

2. { 'state': 'DEFINED_ON_CORE',
    'created': '2022-07-02 10:55:20',
    'lab_title': 'project_lab',
    'lab_description': '',
    'owner': 'developer',
    'node_count': 8,
    'link_count': 9,
    'id': '8cf7ab'}

=====
Menu:
=====
0. exit
1. start lab
```

9. Menjalankan lab dengan menginputkan **1** dan pada lab no inputkan **2** untuk menjalankan lab tersebut



```

8. update config node
9. create interfaces
10. create links
=====
choose (1-9) / 0 for exit: 1
start lab no : 2
Starting Lab with id 8cf7ab ...
status: Success

```

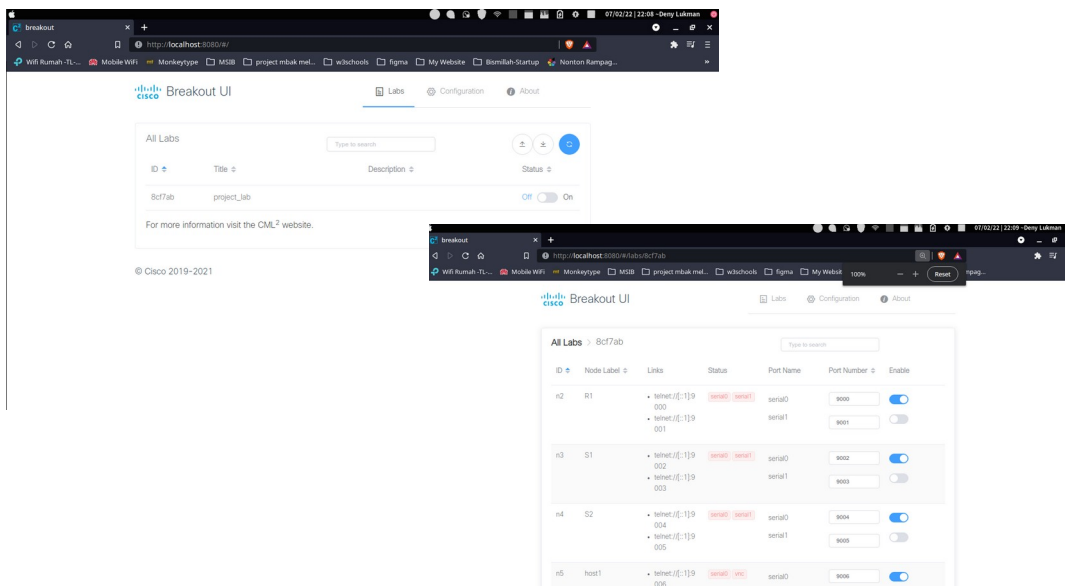
10. Menjalankan `./breakout-linux-x86_amd64 ui`

```

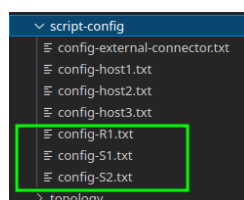
cakdeny49@cakdeny-thplaptop14ck0xxx ~/Documents/msib-network-programmability/Fin
md64 ui
Starting up...
W0702 22:03:02.640621 *19684 run.go:238] open labs.yaml: no such file or direc
Running... Serving UI/API on http://[::1]:8080, Ctrl-C to stop.

```

11. Buka web dengan url yang tertera pada terminal dan ubah ke mode **on**, supaya bisa diakses lewat local

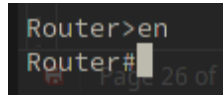


12. Setting awal router dan switch melalui telnet localhost port untuk konfigurasi ip management agar bisa diakses lewat ssh melalui vpn, karena ketika disetting melalui interaktif program methodnya tftp sehingga tidak bisa diakses. Untuk konfigurasi management router dan switchnya sudah ada pada directory `./script-config/` pada **config-R1.txt**, **config-S1.txt**, dan **config-S2.txt**.



13. Buka console R1, S1, dan S2 melalui terminal, copy dari script config dan paste ke console melalui R1: localhost 9000, S1: localhost 9002, S2: localhost 9004. Untuk username dan password pada Switch menggunakan cisco untuk pertama kali.

```
cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML $ telnet localhost 9000
Trying ::1...
Connected to localhost.
Escape character is '^['.
% Please answer 'yes' or 'no'.
Would you like to enter the initial configuration dialog? [yes/no]: no
Would you like to terminate autostart? [yes]: yes
```



```
Router>conf t
Router(config)#hostname R1
R1(config)#vrf definition Mgmt-Intf
R1(config-vrf)#address-family ipv4
R1(config-vrf-af)#exit-address-family
R1(config-vrf)#exit
R1(config)#enable secret Admin12345
R1(config)#ip domain lookup
R1(config)#ip domain name admin.info
R1(config)#username admin privilege 15 secret Admin12345
R1(config)#crypto key generate rsa general-keys modulus 2048
The name for the keys will be: R1.admin.info
get active VNC keys from controller...
% The key modulus size is 2048 bits
% Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)
R1(config)#ip ssh version 2
R1(config)#int GigabitEthernet1
R1(config-if)#no ip dhcp client
R1(config-if)#vrf forwarding Mgmt-Intf
R1(config-if)#description interface for management
R1(config-if)#ip add 10.10.20.200 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#line vty 0 4
R1(config-line)#login local
R1(config-line)#transport input ssh
R1(config-line)#exit
R1(config)#ip route vrf Mgmt-Intf 0.0.0.0 0.0.0.0 10.10.20.254
R1(config)#end
```

```
switch# conf t
switch(config)# hostname S1
S1(config)# feature interface-vlan
S1(config)# feature nxapi
S1(config)# feature ssh
S1(config)# vrf context management
S1(config-vrf)# ip route 0.0.0.0/0 10.10.20.254
S1(config-vrf)# exit
S1(config)# int mgmt 0
S1(config-if)# vrf member management
S1(config-if)# description interface for management
S1(config-if)# ip add 10.10.20.201/24
S1(config-if)# exit
S1(config)# username admin password Admin12345
S1(config)# end
S1# copy run start
[#####] 100%
Copy complete, now saving to disk (please wait)...
Copy complete.
```

14. cek ssh akses ke R1, S1, dan S2 melalui username **admin** dan password **Admin12345**. R1: 10.10.20.200, S1: 10.10.20.201, S2: 10.10.20.202

```
cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML $ ssh admin@10.10.20.200
The authenticity of host '10.10.20.200 (10.10.20.200)' can't be established.
RSA key fingerprint is SHA256:Yc4KHjxxCFvwmYEI6+PLLoFKQt4gIqNw4YmTrqWFMc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.20.200' (RSA) to the list of known hosts.
(admin@10.10.20.200) Password:
get active VNC keys from controller...
get all the labs from controller...
R1# get all the nodes for the labs from controller...
get nodes for lab 8cf7ab from controller...
get nodes for lab b4bb2d from controller...
2022/07/02 22:05:23 [cakdeny-hplaptop14ck0xxx/8YDfLWTnV-00000061] "GET http://localhost:8080/api/refresh HTTP/1.1"
```

```
cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML $ ssh admin@10.10.20.201
The authenticity of host '10.10.20.201 (10.10.20.201)' can't be established.
RSA key fingerprint is SHA256:ZiPbDJsxE04f80hX2KV5w59FnGX2PeucI7Ha+EJHHVY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.20.201' (RSA) to the list of known hosts.
User Access Verification
(admin@10.10.20.201) Password:

Cisco NX-OS Software
Copyright (c) 2002-2020, Cisco Systems, Inc. All rights reserved.
Nexus 9000v software ("Nexus 9000v Software") and related documentation,
files or other reference materials ("Documentation") are
the proprietary property and confidential information of Cisco
Systems, Inc. ("Cisco") and are protected, without limitation,
pursuant to United States and International copyright and trademark
laws in the applicable jurisdiction which provide civil and criminal
```

```

cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML $ ssh admin@10.10.20.202
The authenticity of host '10.10.20.202 (10.10.20.202)' can't be established.
RSA key fingerprint is SHA256:i9pJ3EPFmPRgx02zma0Tyf2Gk0yvVx7orFu4wY4KwPM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.20.202' (RSA) to the list of known hosts.
User Access Verification
(admin@10.10.20.202) Password:
14 cek ssh akses ke R1, S1, dan S2 melalui username admin dan password Admin12345. R1:
10.10.20.200 S1: 10.10.20.201 S2: 10.10.20.202
Cisco NX-OS Software
Copyright (c) 2002-2020, Cisco Systems, Inc. All rights reserved.
Nexus 9000v software ("Nexus 9000v Software") and related documentation,
files or other reference materials ("Documentation") are

```

### 15. Setting router dengan RESTCONF dengan menjalankan config.py pada ./RESTCONF

```

cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML/RESTCONF $ python3
config.py
<Response [204]>
<Response [204]>
save {
  "cisco-ia:output": {
    "result": "Save running-config successful"
  }
}
cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML/RESTCONF $

```

### 16. cek running-configuration pada router melalui ssh

```

!
interface GigabitEthernet1
description interface for management
vrf forwarding Mgmt-intf
ip address 10.10.20.200 255.255.255.0
negotiation auto
no mop enabled
no mop sysid
!
interface GigabitEthernet2
description Configured by RESTCONF subnet 1
ip address 172.16.0.1 255.255.255.192
negotiation auto
no mop enabled
no mop sysid
!
interface GigabitEthernet3
description Configured by RESTCONF subnet 2
ip address 172.16.0.65 255.255.255.224
negotiation auto
no mop enabled
no mop sysid
!
ip forward-protocol nd

```

### 17. setting 2 switch S1 dan S2 dengan NXAPI dengan menjalankan config.py pada ./NXAPI

```

#####] 98%\n[#####] 100%\nCopy complete, now saving to disk (please wait)... \nCopy complete.\n', 'code': '200', 'msg': 'Success'}
cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML/NXAPI $

```

### 18. cek running-configuration pada switch melalui ssh dan cek konektivitas dengan router

```

rmon event 3 log trap public description ERROR(3) owner PHON@ERROR
rmon event 4 log trap public description WARNING(4) owner PHON@WARNING
rmon event 5 log trap public description INFORMATION(5) owner PHON@INFO

vlan 1,101
vlan 101
name prod

vrf context management
ip route 0.0.0.0/0 10.10.20.254

Interface Vlan1
S1

Interface Vlan101
description config by NXAPI prod
no shutdown
no ip redirects
ip address 172.16.101.2/26

Interface Ethernet1/1
description config by NXAPI connect to R1
ip address 172.16.0.2/26
no shutdown

Interface Ethernet1/2
description config by NXAPI connect to host
switchport
switchport access vlan 101
spanning-tree port type edge
no shutdown

Interface Ethernet1/3

name prod
vrf context management
ip route 0.0.0.0/0 10.10.20.254
S2

Interface Vlan1
Interface Vlan101
description config by NXAPI prod
no shutdown
no ip redirects
ip address 172.16.101.66/27

Interface Ethernet1/1
description config by NXAPI connect to R2
ip address 172.16.0.66/27
no shutdown

Interface Ethernet1/2
description config by NXAPI connect to host
switchport
switchport access vlan 101
spanning-tree port type edge
no shutdown

Interface Ethernet1/3
description config by NXAPI connect to host
switchport
switchport access vlan 101
spanning-tree port type edge
no shutdown

Interface Ethernet1/4

```



```

S1# ping 172.16.0.1
PING 172.16.0.1 (172.16.0.1): 56 data bytes
36 bytes from 172.16.0.2: Destination Host Unreachable
Request 0 timed out
64 bytes from 172.16.0.1: icmp_seq=1 ttl=254 time=2.133 ms
64 bytes from 172.16.0.1: icmp_seq=2 ttl=254 time=2.42 ms
64 bytes from 172.16.0.1: icmp_seq=3 ttl=254 time=2.432 ms
64 bytes from 172.16.0.1: icmp_seq=4 ttl=254 time=1.498 ms

--- 172.16.0.1 ping statistics ---
5 packets transmitted, 4 packets received, 20.00% packet loss
round-trip min/avg/max = 1.498/2.12/2.432 ms
S1#

S2# ping 172.16.0.65
PING 172.16.0.65 (172.16.0.65): 56 data bytes
36 bytes from 172.16.0.66: Destination Host Unreachable
Request 0 timed out
64 bytes from 172.16.0.65: icmp_seq=1 ttl=254 time=54.68 ms
64 bytes from 172.16.0.65: icmp_seq=2 ttl=254 time=1.839 ms
64 bytes from 172.16.0.65: icmp_seq=3 ttl=254 time=1.512 ms
64 bytes from 172.16.0.65: icmp_seq=4 ttl=254 time=2.112 ms

--- 172.16.0.65 ping statistics ---
5 packets transmitted, 4 packets received, 20.00% packet loss
round-trip min/avg/max = 1.512/15.035/54.68 ms
S2#

```

19. cek ip host1 yang telah diconfig saat pembuatan node dengan ifconfig melalui telnet localhost 9006, username host1 dan password host1 dan cek konektivitas dengan S1

```

cakdeny49@cakdeny-hpiaptop14ck0xxx:~/Documents/project/finalProject/src/CML $ telnet localhost 9006
Trying ::1...
Connected to localhost.
Escape character is '^['.

Password:
Login incorrect
host1 login: host1
Password:
Login incorrect
host1 login: host1
Password:
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

host1:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:1A:2E:B4
          inet addr:172.16.101.10  Bcast:172.16.101.63  Mask:255.255.255.192
          inet6 addr: fe80::5054:ff:fe1a:2eb4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:426 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:632 (632.0 B)  TX bytes:140132 (136.8 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

host1:~$

host1:~$ ping 172.16.101.2
PING 172.16.101.2 (172.16.101.2): 56 data bytes
64 bytes from 172.16.101.2: seq=0 ttl=42 time=1.965 ms
64 bytes from 172.16.101.2: seq=1 ttl=42 time=1.820 ms
64 bytes from 172.16.101.2: seq=2 ttl=42 time=1.753 ms
64 bytes from 172.16.101.2: seq=3 ttl=42 time=1.755 ms
64 bytes from 172.16.101.2: seq=4 ttl=42 time=1.929 ms
64 bytes from 172.16.101.2: seq=5 ttl=42 time=1.737 ms
64 bytes from 172.16.101.2: seq=6 ttl=42 time=1.669 ms
64 bytes from 172.16.101.2: seq=7 ttl=42 time=1.760 ms
64 bytes from 172.16.101.2: seq=8 ttl=42 time=1.695 ms
64 bytes from 172.16.101.2: seq=9 ttl=42 time=1.736 ms
64 bytes from 172.16.101.2: seq=10 ttl=42 time=2.021 ms
64 bytes from 172.16.101.2: seq=11 ttl=42 time=1.864 ms
64 bytes from 172.16.101.2: seq=12 ttl=42 time=1.936 ms
64 bytes from 172.16.101.2: seq=13 ttl=42 time=2.375 ms
64 bytes from 172.16.101.2: seq=14 ttl=42 time=1.812 ms
64 bytes from 172.16.101.2: seq=15 ttl=42 time=1.947 ms
64 bytes from 172.16.101.2: seq=16 ttl=42 time=1.885 ms
64 bytes from 172.16.101.2: seq=17 ttl=42 time=1.824 ms
64 bytes from 172.16.101.2: seq=18 ttl=42 time=1.857 ms
64 bytes from 172.16.101.2: seq=19 ttl=42 time=2.134 ms
64 bytes from 172.16.101.2: seq=20 ttl=42 time=2.399 ms
64 bytes from 172.16.101.2: seq=21 ttl=42 time=1.868 ms
64 bytes from 172.16.101.2: seq=22 ttl=42 time=1.890 ms
64 bytes from 172.16.101.2: seq=23 ttl=42 time=1.884 ms
64 bytes from 172.16.101.2: seq=24 ttl=42 time=1.630 ms
64 bytes from 172.16.101.2: seq=25 ttl=42 time=1.783 ms
^C
--- 172.16.101.2 ping statistics ---
26 packets transmitted, 26 packets received, 0% packet loss
round-trip min/avg/max = 1.630/1.881/2.399 ms
host1:~$

```

20. cek konektivitas host2 dan host3, melalui host2 dengan telnet localhost 9007 username host2 dan password host2, ping ip host3 yaitu 172.16.101.80.

```

cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-netwo
project/finalProject/src/CML $ telnet localhost 9007
Trying ::1...
Connected to localhost.
Escape character is '^]'.

Welcome to Alpine Linux 3.13
Kernel 5.10.16-0-virt on an x86_64 (/dev/ttyS0)

host2 login:
Welcome to Alpine Linux 3.13
Kernel 5.10.16-0-virt on an x86_64 (/dev/ttyS0)

host2 login: host2
Password:
Welcome to Alpine!

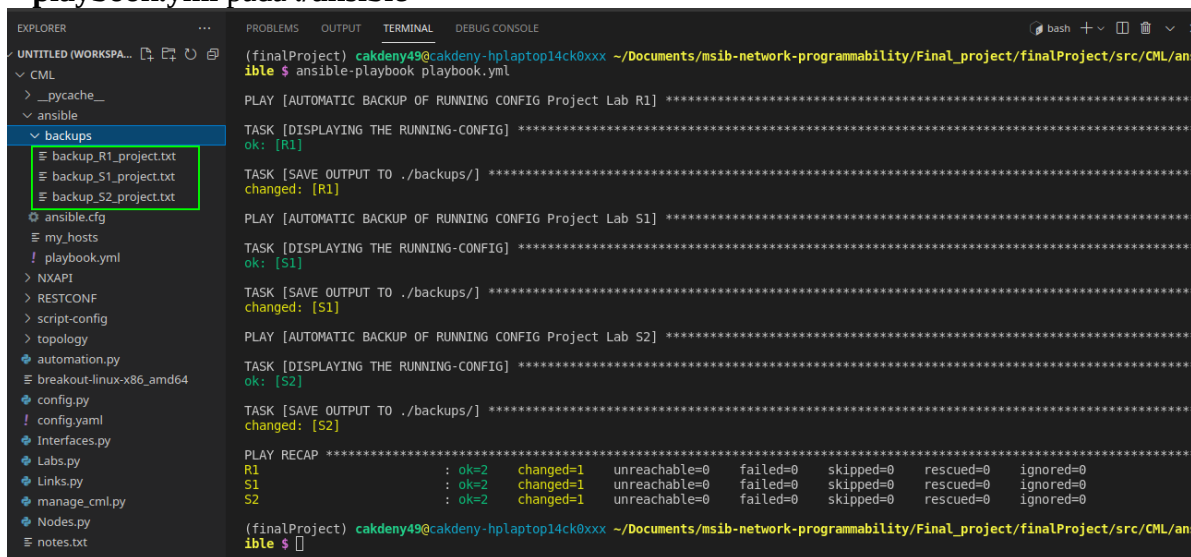
```

```

host2:~$ ping 172.16.101.80
PING 172.16.101.80 (172.16.101.80): 56 data bytes
64 bytes from 172.16.101.80: seq=0 ttl=42 time=3.007 ms
64 bytes from 172.16.101.80: seq=1 ttl=42 time=1.755 ms
64 bytes from 172.16.101.80: seq=2 ttl=42 time=1.715 ms
64 bytes from 172.16.101.80: seq=3 ttl=42 time=1.658 ms
64 bytes from 172.16.101.80: seq=4 ttl=42 time=1.515 ms
64 bytes from 172.16.101.80: seq=5 ttl=42 time=1.490 ms
64 bytes from 172.16.101.80: seq=6 ttl=42 time=2.705 ms
^C
--- 172.16.101.80 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 1.490/1.977/3.007 ms
host2:~$

```

## 21. Backup running-configuration dengan ansible dengan menjalankan **ansible-playbook** **playbook.yml** pada **./ansible**



```

(finalProject) cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML/ans
ible $ ansible-playbook playbook.yml

PLAY [AUTOMATIC BACKUP OF RUNNING CONFIG Project Lab R1] *****

TASK [DISPLAYING THE RUNNING-CONFIG] *****
ok: [R1]

TASK [SAVE OUTPUT TO ./backups/] *****
changed: [R1]

PLAY [AUTOMATIC BACKUP OF RUNNING CONFIG Project Lab S1] *****

TASK [DISPLAYING THE RUNNING-CONFIG] *****
ok: [S1]

TASK [SAVE OUTPUT TO ./backups/] *****
changed: [S1]

PLAY [AUTOMATIC BACKUP OF RUNNING CONFIG Project Lab S2] *****

TASK [DISPLAYING THE RUNNING-CONFIG] *****
ok: [S2]

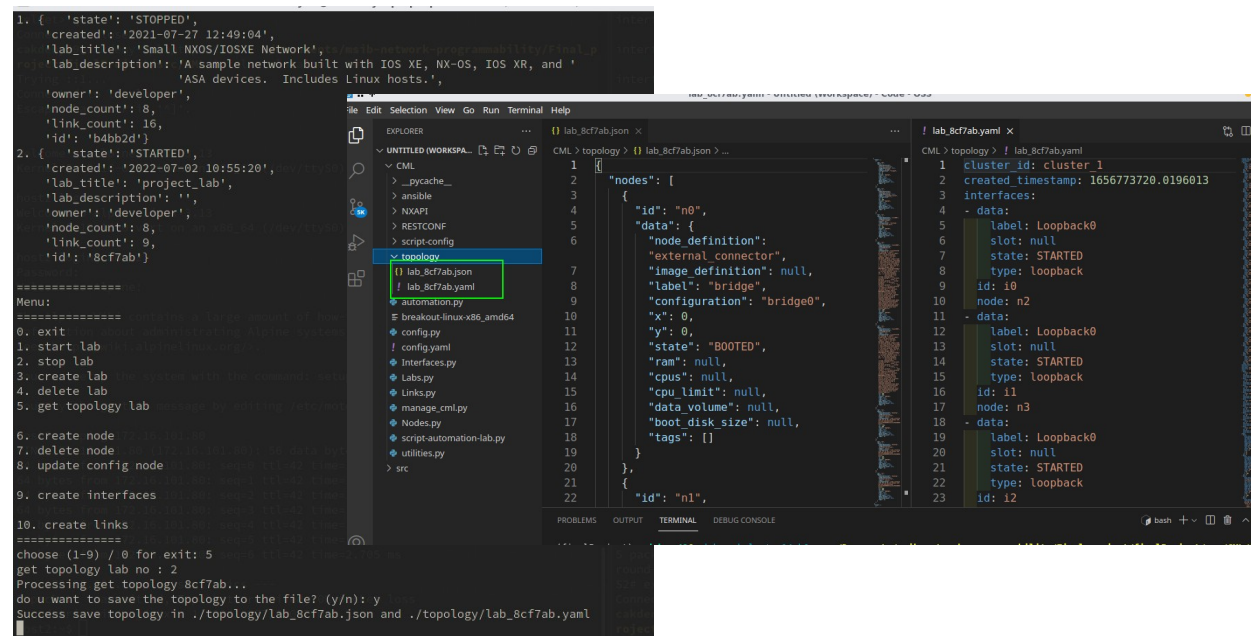
TASK [SAVE OUTPUT TO ./backups/] *****
changed: [S2]

PLAY RECAP *****
R1      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S1      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
S2      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

(finalProject) cakdeny49@cakdeny-hplaptop14ck0xxx ~/Documents/msib-network-programmability/Final_project/finalProject/src/CML/ans
ible $

```

## 22. get topology lab dengan interaktif program



```

1. { 'state': 'STOPPED',
    'created': '2021-07-27 12:49:04',
    'lab_title': 'Small NXOS/IOSXE Network',
    'lab_description': 'A sample network built with IOS XE, NX-OS, IOS XR, and
    'owner': 'developer',
    'node_count': 8,
    'link_count': 16,
    'id': 'b4bb2d' }

2. { 'state': 'STARTED',
    'created': '2022-07-02 10:55:20',
    'lab_title': 'project_lab',
    'lab_description': '',
    'owner': 'developer',
    'node_count': 8,
    'link_count': 9,
    'id': '8cf7ab' }

Menu:
=====
0. exit
1. start lab
2. stop lab
3. create lab
4. delete lab
5. get topology lab
6. create node
7. delete node
8. update config node
9. create interfaces
10. create links

choose (1-9) / 0 for exit: 5
get topology lab no: 2
Processing get topology 8cf7ab...
do u want to save the topology to the file? (y/n): y
Success save topology in ./topology/lab_8cf7ab.json and ./topology/lab_8cf7ab.yml

```

```

{
  "nodes": [
    {
      "id": "n0",
      "data": {
        "node_definition": "external connector",
        "image_definition": null,
        "label": "bridge",
        "configuration": "bridge0",
        "x": 0,
        "y": 0,
        "state": "BOOTED",
        "ram": null,
        "cpus": null,
        "cpu_limit": null,
        "data_volume": null,
        "boot_disk_size": null,
        "tags": []
      }
    },
    {
      "id": "n1",
      "data": {
        "node_definition": "loopback",
        "image_definition": null,
        "label": "loopback",
        "configuration": "loopback",
        "x": 0,
        "y": 0,
        "state": "STARTED",
        "ram": null,
        "cpus": null,
        "cpu_limit": null,
        "data_volume": null,
        "boot_disk_size": null,
        "tags": []
      }
    },
    {
      "id": "n2",
      "data": {
        "node_definition": "loopback",
        "image_definition": null,
        "label": "loopback",
        "configuration": "loopback",
        "x": 0,
        "y": 0,
        "state": "STARTED",
        "ram": null,
        "cpus": null,
        "cpu_limit": null,
        "data_volume": null,
        "boot_disk_size": null,
        "tags": []
      }
    },
    {
      "id": "n3",
      "data": {
        "node_definition": "loopback",
        "image_definition": null,
        "label": "loopback",
        "configuration": "loopback",
        "x": 0,
        "y": 0,
        "state": "STARTED",
        "ram": null,
        "cpus": null,
        "cpu_limit": null,
        "data_volume": null,
        "boot_disk_size": null,
        "tags": []
      }
    }
  ]
}

```