VIETNAM NATIONAL UNIVERSITY HCMC
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER NETWORKS AND COMMUNICATIONS

PROGRESS REPORT
NETWORK AND SYSTEM ADMINISTRATION

PROJECT TITLE
NETWORK AUTOMATION

Lecturer: MS. Trần Thị Dung
Class: NT132.O12.ATCL
Group: 14

Huỳnh Anh Nguyễn	21522388
Bùi Nguyên Phúc	21522469
Trần Tấn Hải	21522036

TABLE OF CONTENTS

I. Introduction

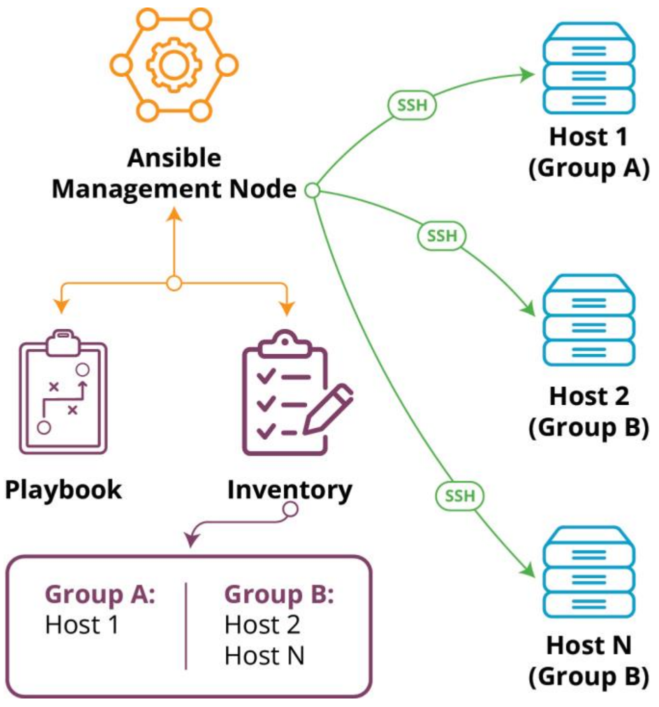
1. General information

- ***Network automation*** is the process of automating the configuring, managing, testing, deploying, and operating of physical and virtual devices within a network. With everyday network tasks and functions automated and repetitive processes controlled and managed automatically, network service availability improves.
- Any type of network can use network automation. Hardware- and software-based solutions enable data centers, service providers, and enterprises to implement network automation to improve efficiency, reduce human error, and lower operating expenses.

2. Components

- ***Ansible:*** A tool that helps us automate the configuration, deployment and management of network devices, server, cloud platform, and application. Ansible is agentless so it does not require any software to be installed on the managed nodes. Ansible uses YAML to describe the desired state of the system and executes modules to achieve that state and supports variables, templates, roles, and collections to make the automation more reusable and modular. It can handle different network protocols, platforms, and vendors and can also gather facts from network devices, validate network configurations, and detect network drift. Ansible network modules are designed to work with various network devices and operating systems, such as Cisco, Juniper, Arista, etc.
- ***GNS3:*** A tool that helps users create topologies and emulate different network devices and platforms, such as routers, switches, firewalls, and servers.
- ***GNS3 VM:*** A Virtual machine that runs the GNS3 Server, which hosts and controls the virtual devices. It can be used to learn, experiment & troubleshoot network scenarios without the need of using physical hardware.
- ***VMWare Workstation Player:*** Software that allows the virtualization of operating systems on a single computer. (Used in running GNS3 VM)
- ***Node Devices & IOS:***
    - (Router) Cisco C3725
    - (Router) Cisco IOSv 15.6(2)T
    - (Switch) Cisco IOSvL2 15.2(4.0.55)E

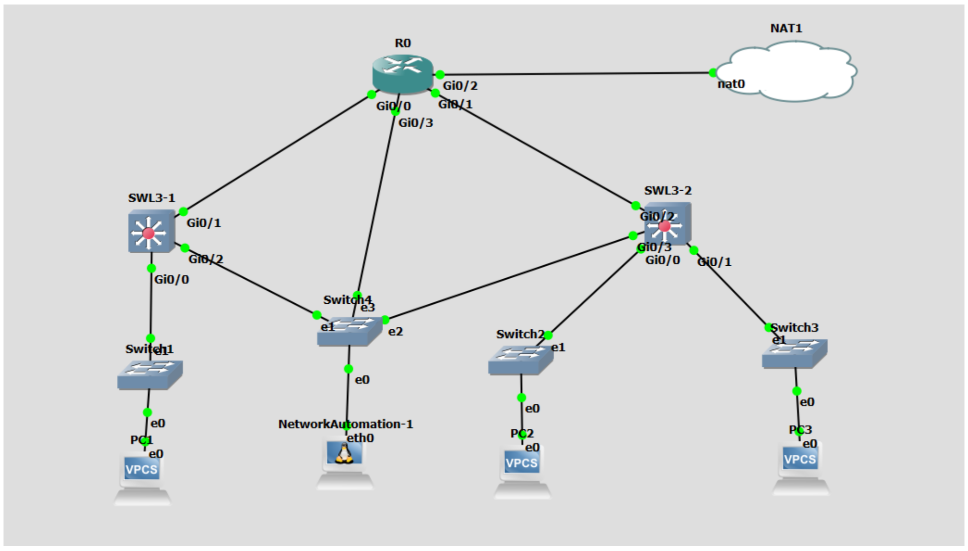
3. Operation



- For automating network devices, Ansible will run on the control node. Since Ansible is agentless, it can still communicate with devices without requiring an application or service to be installed on the managed node. To increase execution capacity for devices without the ability to run modules, Ansible Automation Platform can spread automation jobs out across execution nodes using a technology called automation mesh.
- Automation mesh is an overlay network intended to ease the distribution of automation across a collection of execution nodes using existing connectivity.
- Execution nodes are where Ansible Playbooks are actually executed. A node will run an automation execution environment which will, in turn, run the Ansible Playbook.
- ***Ansible Playbook*** is a blueprint of automation tasks, which are IT actions executed with limited manual effort across an inventory of IT solutions. ***Playbooks*** tell Ansible what to do to which devices.
- ***Ansible Inventory file*** defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. And it uses YAML format.

II. Implementation

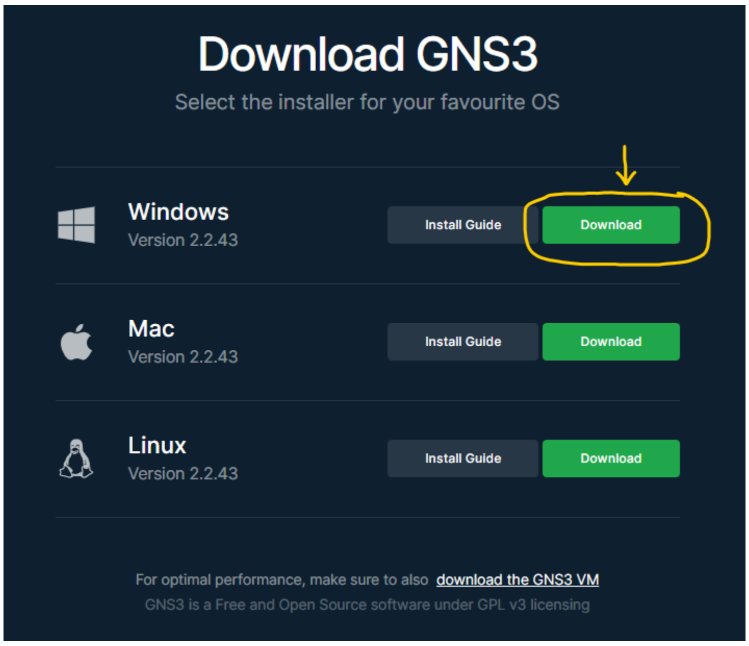
1. Topology



Device	IP address	Subnet mask	Default gateway	Services
R0	G0/0: 192.168.254.2	255.255.255.0	N/A	SSH
	G0/3: 192.168.10.10		N/A	
	G0/1: 192.168.253.2		N/A	
	G0/2: 192.168.122.171		N/A	
SWL3-1	G0/0: 192.168.1.1	255.255.255.0	N/A	SSH
	G0/2: 192.168.10.20		N/A	
	G0/1: 192.168.254.1		N/A	
SWL3-2	G0/0: 192.168.2.1	255.255.255.0	N/A	SSH
	G0/1: 192.168.3.1		N/A	
	G0/2: 192.168.253.1		N/A	
	G0/3: 192.168.10.30		N/A	
PC1	192.168.1.10	255.255.255.0	192.168.1.1	SSH
PC2	192.168.2.10	255.255.255.0	192.168.2.1	SSH
PC3	192.168.3.10	255.255.255.0	192.168.3.1	SSH
NetAuto1	192.168.10.100	255.255.255.0	192.168.10.10	Ansible

2. Installation

- Download & Install GNS3 2.2.43 (Windows) from official website: https://gns3.com/download


Download GNS3

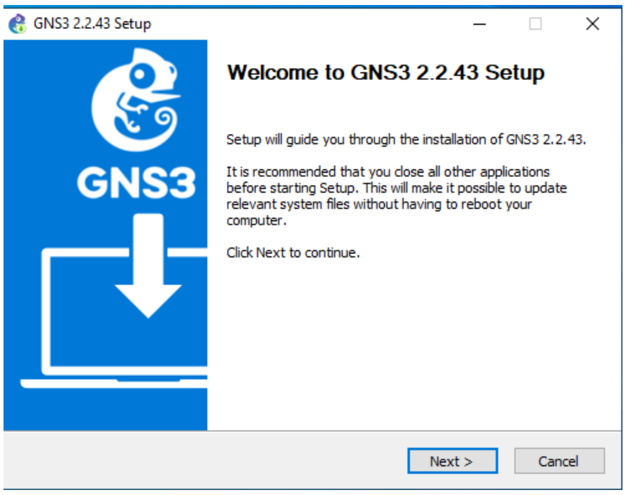
Select the installer for your favourite OS

Windows	Version 2.2.43	Install Guide / Download
Mac	Version 2.2.43	Install Guide / Download
Linux	Version 2.2.43	Install Guide / Download

For optimal performance, make sure to also download the GNS3 VM

GNS3 is a Free and Open Source software under GPL v3 licensing

- Keep the default setup settings and press "Next >" until the installation process is complete.


GNS3 2.2.43 Setup

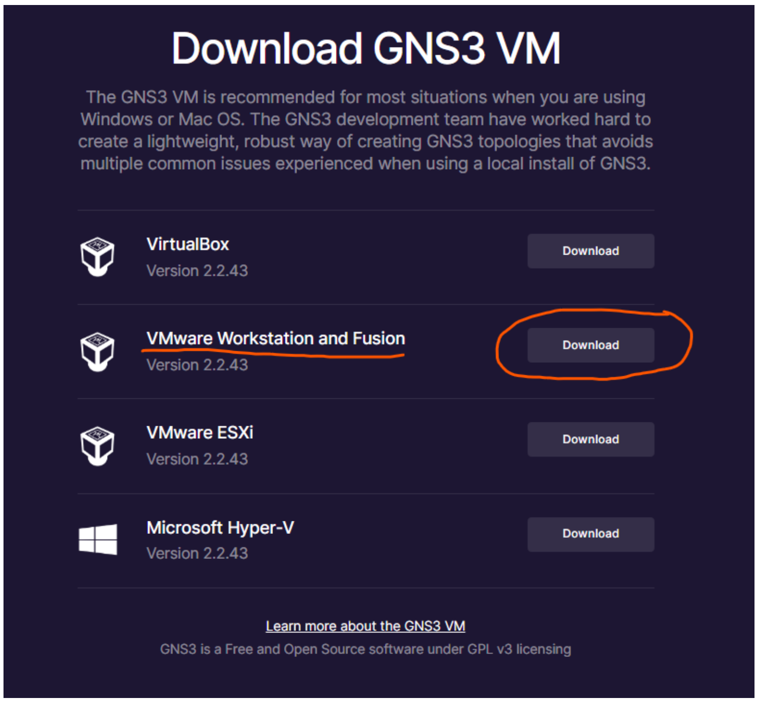
Welcome to GNS3 2.2.43 Setup

Setup will guide you through the installation of GNS3 2.2.43.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

- Download & Install VMWare Workstation Player: https://www.vmware.com/content/vmware/vmware-published-sites/us/products/workstation-player/workstation-player-evaluation.html
- Download GNS3 VM: https://gns3.com/software/download-vm



Download GNS3 VM

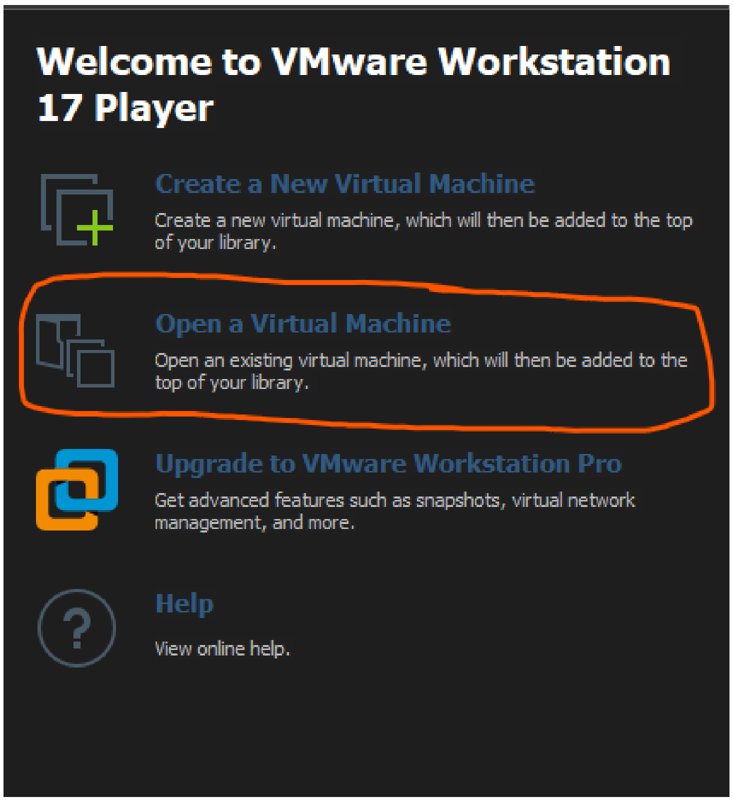
The GNS3 VM is recommended for most situations when you are using Windows or Mac OS. The GNS3 development team have worked hard to create a lightweight, robust way of creating GNS3 topologies that avoids multiple common issues experienced when using a local install of GNS3.

VirtualBox Version 2.2.43	Download
VMware Workstation and Fusion Version 2.2.43	Download
VMware ESXi Version 2.2.43	Download
Microsoft Hyper-V Version 2.2.43	Download

Learn more about the GNS3 VM

GNS3 is a Free and Open Source software under GPL v3 licensing

- Extract the downloaded archive.
- In VMWare Workstation Player, Choose Open a Virtual Machine & Import the "GNS3 VM.ova" file then choose the desired location for the Virtual Machine.



Welcome to VMware Workstation 17 Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation Pro
Get advanced features such as snapshots, virtual network management, and more.

Help
View online help.

-

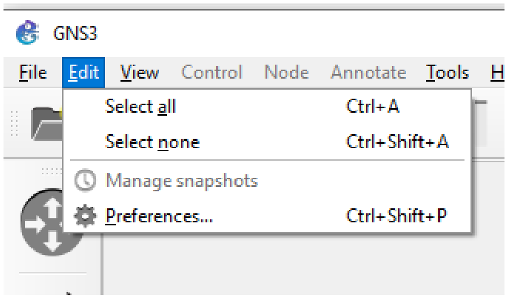
- Download C3725 & IOS Images:

  https://drive.google.com/drive/folders/1AUD4zwBhoVQW0SOOQr_mM-HNnfDVbdPl

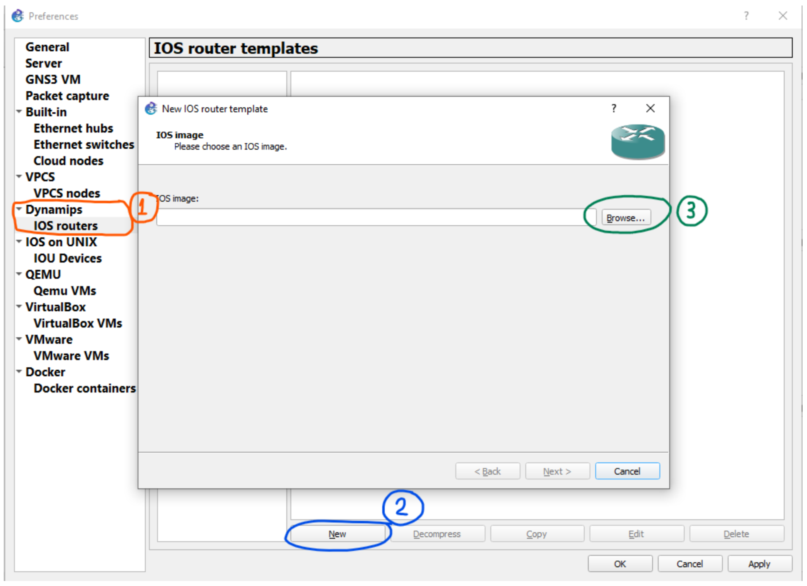
  https://drive.google.com/file/d/1b8wD1WtBHLqQfoHrztxaZ2yMeTsaprbe/view

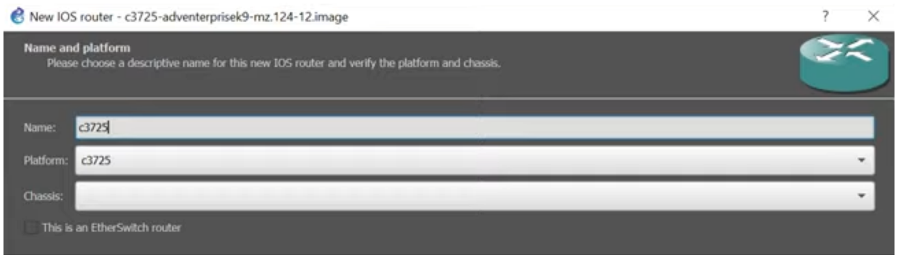
  https://upw.io/75g/vios_l2-adventerprisek9-m.vmdk.SSA.152-4.0.55.E

- Add device images: Edit > Preferences

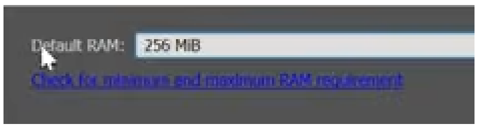


- ***Adding C3725 Router:*** Create a new Dynamips IOS Router & Browse the downloaded C3725 image file with the following configurations:



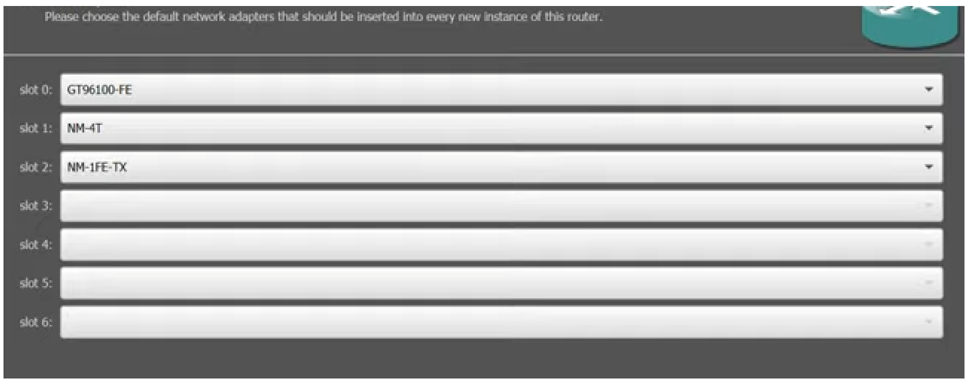

New IOS router - c3725-adventerprisek9-mz.124-12.image
Name and platform
Please choose a descriptive name for this new IOS router and verify the platform and chassis.
This is an EtherSwitch router

Name: c3725
Platform: c3725
Chassis:



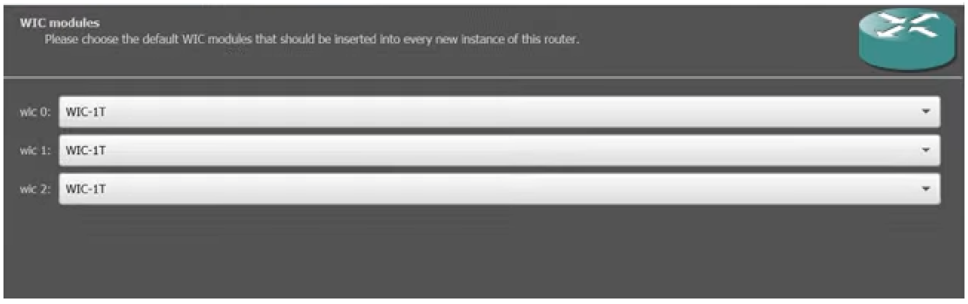
Default RAM: 256 MiB

Check for minimum and maximum RAM requirement



Please choose the default network adapters that should be inserted into every new instance of this router.

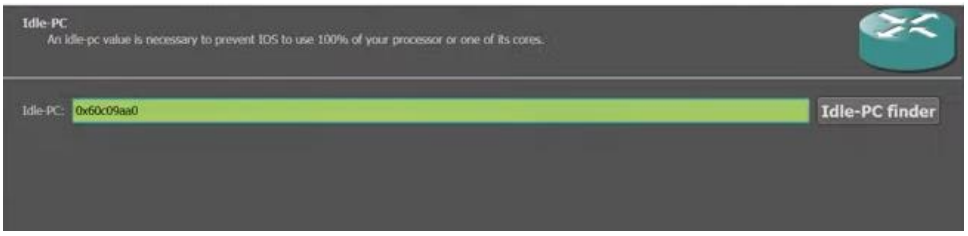
slot 0:	GT96100-FE
slot 1:	NM-4T
slot 2:	NM-1FE-TX
slot 3:	
slot 4:	
slot 5:	
slot 6:	



WIC modules

Please choose the default WIC modules that should be inserted into every new instance of this router.

wic 0:	WIC-1T
wic 1:	WIC-1T
wic 2:	WIC-1T



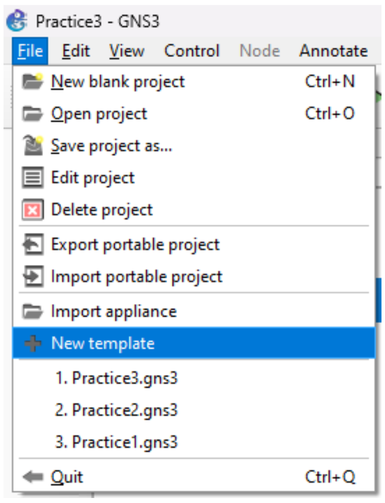
Idle-PC

An idle-pc value is necessary to prevent IOS to use 100% of your processor or one of its cores.

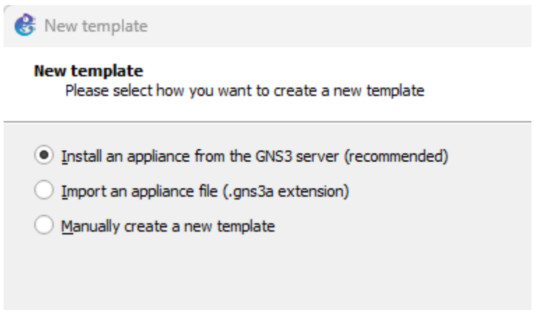
Idle-PC: 0x60c09aa0

- When finished, press “Apply” to save changes.

- *Adding Cisco IOSv 15.6(2)T Router:* Go to File > + New Template



- Choose Install an Appliance from the GNS3 Server:

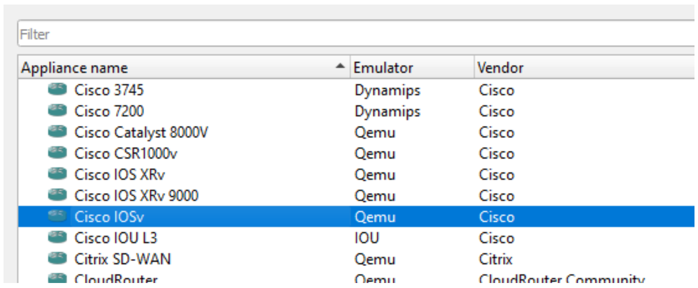


New template

New template
Please select how you want to create a new template

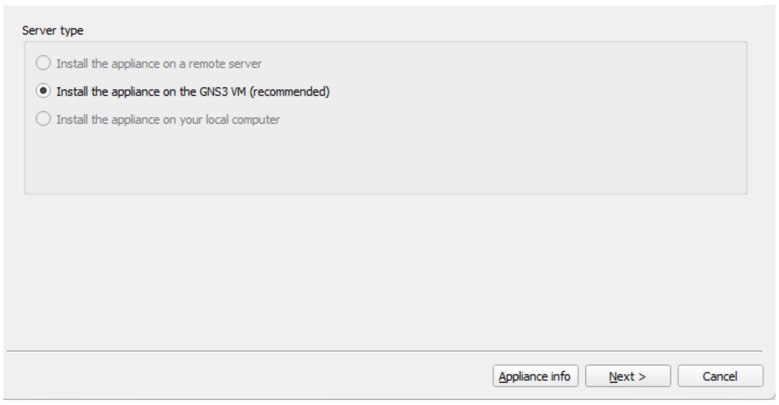
[✓ Install an appliance from the GNS3 server (recommended)]
Import an appliance file (.gns3a extension)
Manually create a new template

- Choose Routers > Cisco IOSv (Quemu Emulator) and Install



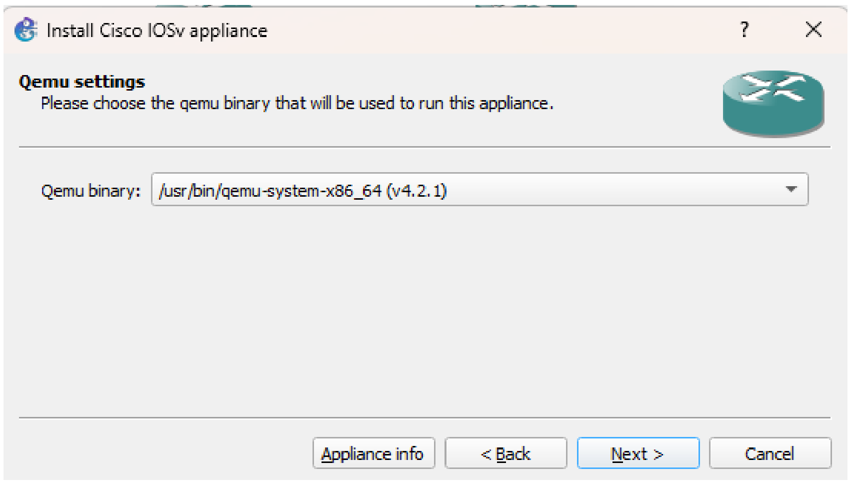
Appliance name	Emulator	Vendor
Cisco 3745	Dynamips	Cisco
Cisco 7200	Dynamips	Cisco
Cisco Catalyst 8000V	Qemu	Cisco
Cisco CSR1000v	Qemu	Cisco
Cisco IOS XRv	Qemu	Cisco
Cisco IOS XRv 9000	Qemu	Cisco
Cisco IOSv	Qemu	Cisco
Cisco IOU L3	IOU	Cisco
Citrix SD-WAN	Qemu	Citrix
CloudRouter	Qemu	CloudRouter Community

- Choose Next >



Server type

Install the appliance on a remote server
[✓ Install the appliance on the GNS3 VM (recommended)]
Install the appliance on your local computer

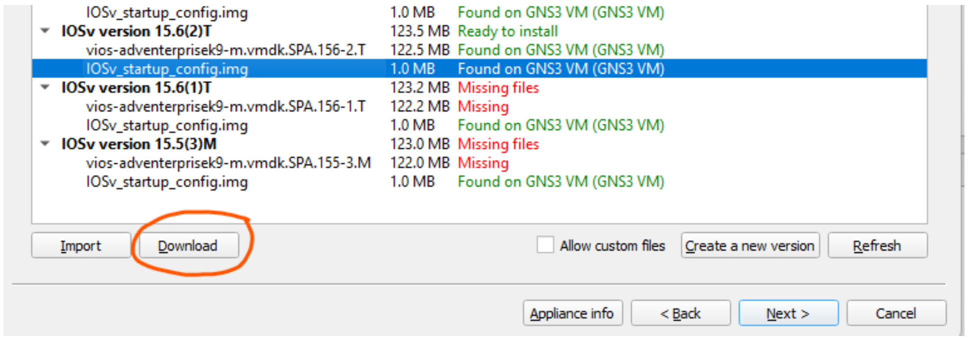


Install Cisco IOSv appliance

Qemu settings
Please choose the qemu binary that will be used to run this appliance.

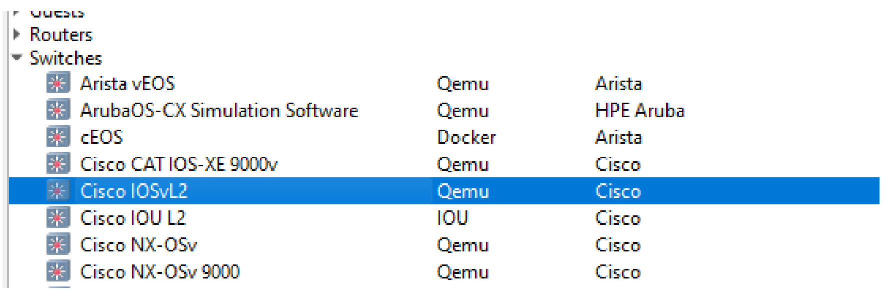
Qemu binary: /usr/bin/qemu-system-x86_64 (v4.2.1)

- Download the missing IOSv_startup_config.img file from SourceForge & Import it.



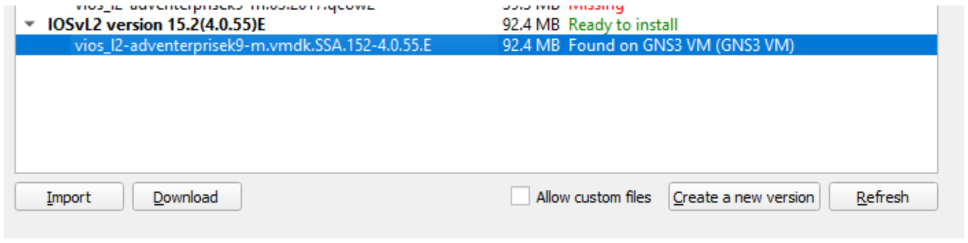
```
    IOSv_startup_config.img                       1.0 MB    Found on GNS3 VM (GNS3 VM)
 IOSv version 15.6(2)T                            123.5 MB  Ready to install
    vios-adventerprisek9-m.vmdk.SPA.156-2.T       122.5 MB  Found on GNS3 VM (GNS3 VM)
    IOSv_startup_config.img                       1.0 MB    Found on GNS3 VM (GNS3 VM)
 IOSv version 15.6(1)T                            123.2 MB  Missing files
    vios-adventerprisek9-m.vmdk.SPA.156-1.T       122.2 MB  Missing
    IOSv_startup_config.img                       1.0 MB    Found on GNS3 VM (GNS3 VM)
 IOSv version 15.5(3)M                            123.0 MB  Missing files
    vios-adventerprisek9-m.vmdk.SPA.155-3.M       122.0 MB  Missing
    IOSv_startup_config.img                       1.0 MB    Found on GNS3 VM (GNS3 VM)
```

- After that Import the vios-adventerprisek9-m.vmdk.SPA.156-2.T image file and select the device then finish the installation.
- ***Adding Cisco IOSvL2 15.2(4.0.55)E Switch:*** Go to File > + New Template
- Choose Install an Appliance from the GNS3 Server.
- Choose Switches > Cisco IOSvL2 and Install.



```
 Guests
 Routers
 Switches
    Arista vEOS                          Qemu     Arista
    ArubaOS-CX Simulation Software       Qemu     HPE Aruba
    cEOS                                 Docker   Arista
    Cisco CAT IOS-XE 9000v               Qemu     Cisco
    Cisco IOSvL2                         Qemu     Cisco
    Cisco IOU L2                         IOU      Cisco
    Cisco NX-OSv                         Qemu     Cisco
    Cisco NX-OSv 9000                    Qemu     Cisco
```

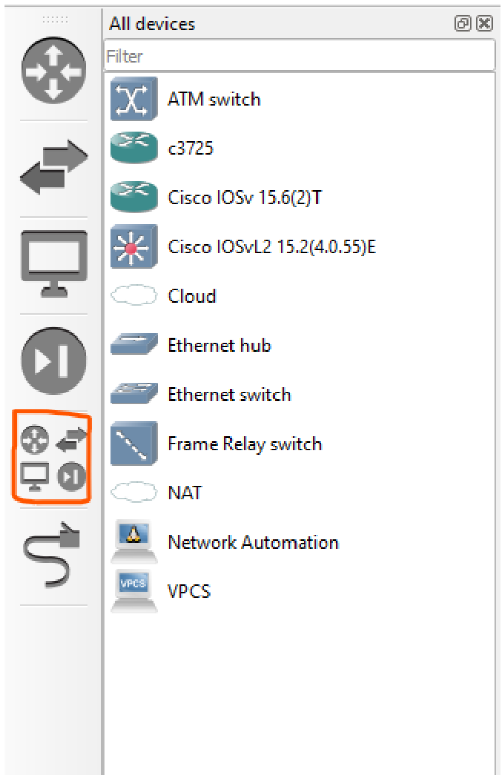
- Import viosl2-adventerprisek9-m.vmdk.SSA.152-4.0.55.E and and select the device then finish the installation.



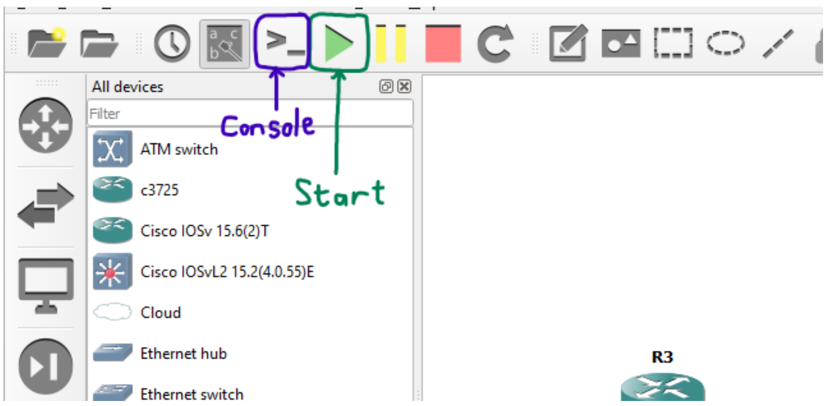
```
 IOSvL2 version 15.2(4.0.55)E                       92.4 MB  Ready to install
    vios_l2-adventerprisek9-m.vmdk.SSA.152-4.0.55.E  92.4 MB  Found on GNS3 VM (GNS3 VM)
```

3. Configuration

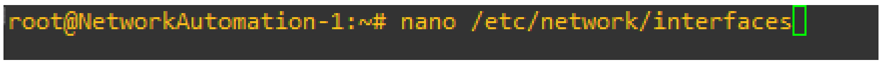
- Create a new project & from the list of devices, build the topology shown in "1. Topology":



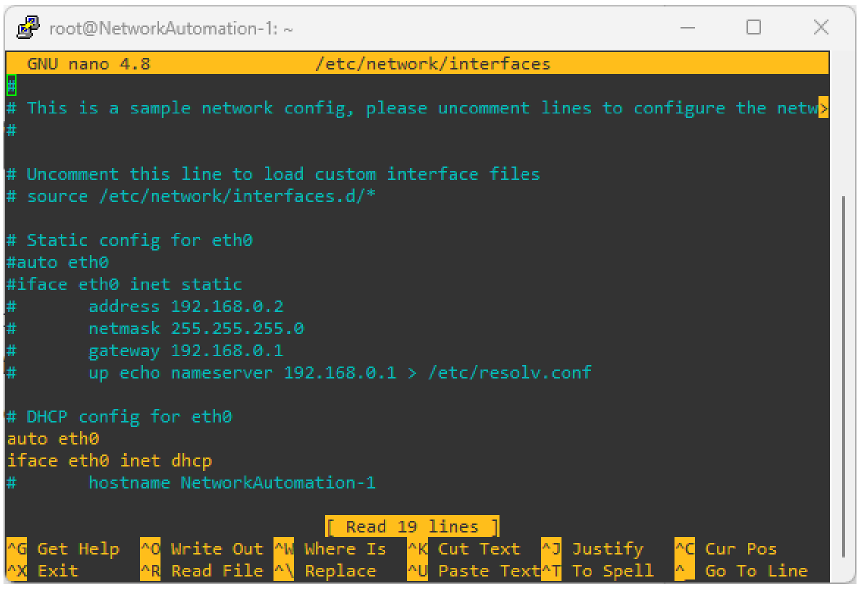
- Start the devices in the topology and open the console of each devices using the buttons below:



- Inside NetworkAutomation-1 Node, use the command #nano /etc/network/interfaces to edit and remove comment symbol "#" from the following lines:
  - auto eth0
  - iface eth0 inet dhcp
- Use ***Ctrl+X*** to exit and press ***Y*** and ***Enter*** to save changes to the config file.

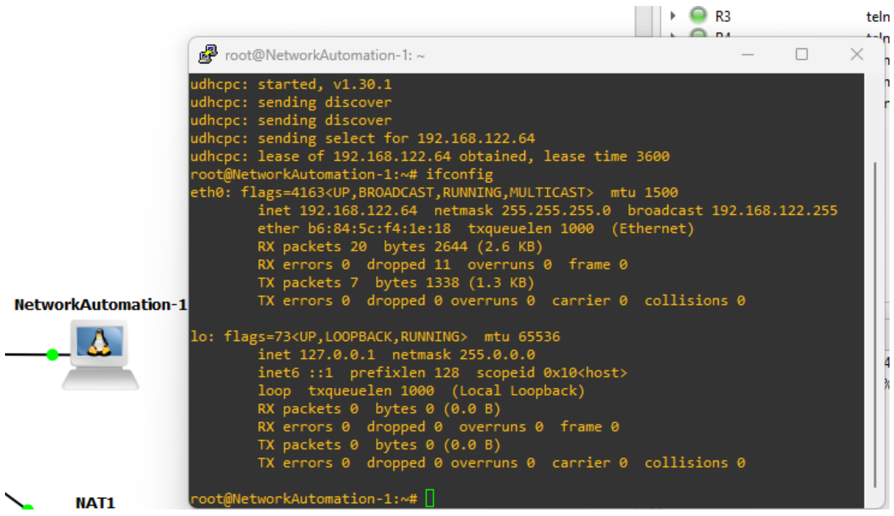


```
root@NetworkAutomation-1:~# nano /etc/network/interfaces
```



```
  GNU nano 4.8                  /etc/network/interfaces
#
# This is a sample network config, please uncomment lines to configure the netw>
#

# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*

# Static config for eth0
#auto eth0
#iface eth0 inet static
#       address 192.168.0.2
#       netmask 255.255.255.0
#       gateway 192.168.0.1
#       up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
auto eth0
iface eth0 inet dhcp
#       hostname NetworkAutomation-1

                               [ Read 19 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

- After that, restart the device and use #ifconfig to check, the device now has an assigned IP Address.



```
udhcpc: started, v1.30.1
udhcpc: sending discover
udhcpc: sending discover
udhcpc: sending select for 192.168.122.64
udhcpc: lease of 192.168.122.64 obtained, lease time 3600
root@NetworkAutomation-1:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.64  netmask 255.255.255.0  broadcast 192.168.122.255
        ether b6:84:5c:f4:1e:18  txqueuelen 1000  (Ethernet)
        RX packets 20  bytes 2644 (2.6 KB)
        RX errors 0  dropped 11  overruns 0  frame 0
        TX packets 7  bytes 1338 (1.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@NetworkAutomation-1:~#
```

Configure the network devices in the network using the following commands:

R0:

- int g0/0
- ip add 192.168.254.2 255.255.255.0
- int g0/1
- ip add 192.168.253.2 255.255.255.0
- int g0/3

- ip add 192.168.10.10 255.255.255.0
- int g0/2
- ip add dhcp

SWL3-1:

- int g0/0
- no sw
- no shut
- ip add 192.168.1.1 255.255.255.0
- int g0/1
- no sw
- no shut
- ip add 192.168.254.1 255.255.255.0
- int g0/2
- no sw
- no shut
- ip add 192.168.10.20 255.255.255.0

SWL3-2:

- int g0/0
- no sw
- no shut
- ip add 192.168.2.1 255.255.255.0
- int g0/1
- no sw
- no shut
- ip add 192.168.3.1 255.255.255.0
- int g0/2
- no sw
- no shut

- ip add 192.168.253.1 255.255.255.0
- int g0/3
- no sw
- no shut
- ip add 192.168.10.30

PC1:

- ip 192.168.1.10 255.255.255.0 192.168.1.1

PC2:

- ip 192.168.2.10 255.255.255.0 192.168.2.1

PC3:

- ip 192.168.3.10 255.255.255.0 192.168.3.1

***RIP ROUTING & NAT:**

SWL3-1:

- conf t
- ip routing
- router rip
- version 2
- network 192.168.1.0
- network 192.168.254.0
- passive-interface g0/0

SWL3-2:

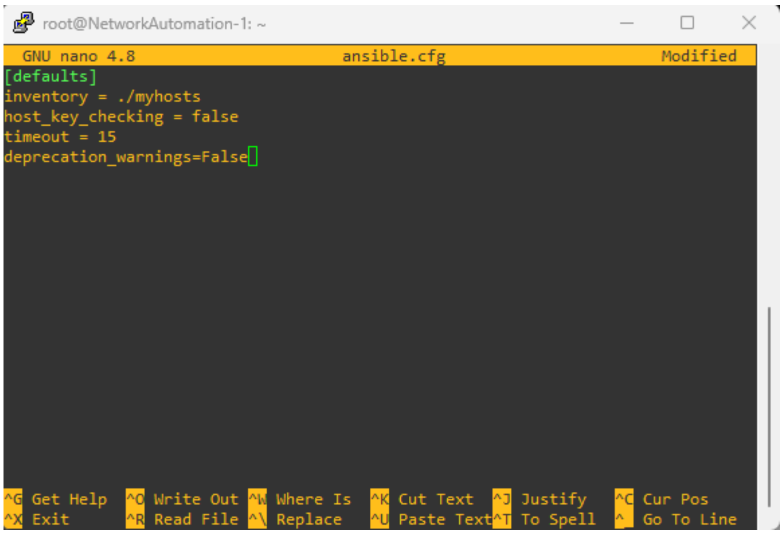
- conf t
- ip routing
- router rip
- version 2
- network 192.168.2.0
- network 192.168.3.0
- network 192.168.253.0
- passive-interface g0/0
- passive-interface g0/1

R0:

- conf t
- ip route 0.0.0.0 0.0.0.0 192.168.122.1
-
- access-list 1 permit 192.168.0.0 0.0.255.255
- ip nat inside source list 1 interface g0/2 overload

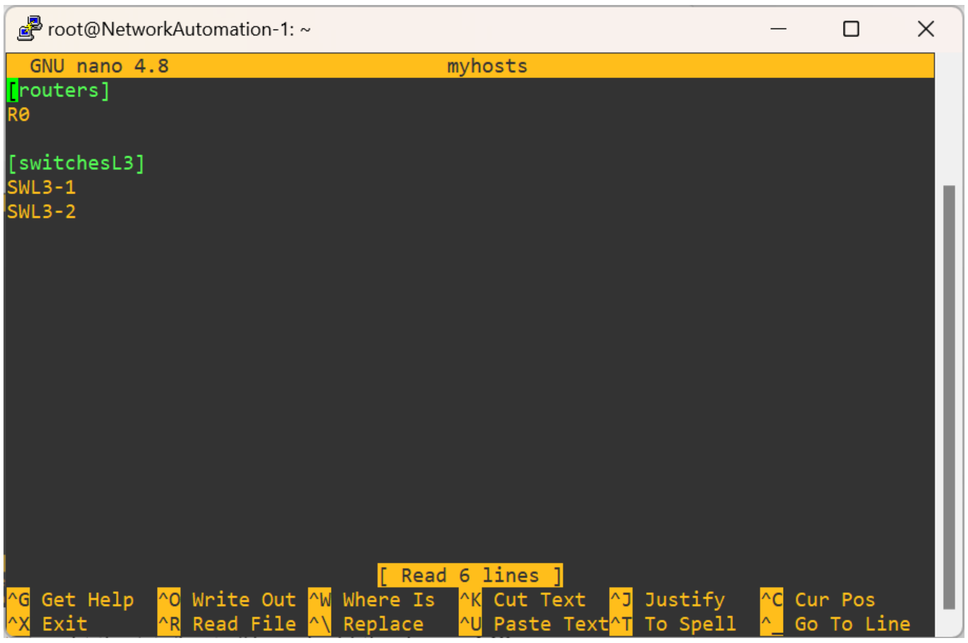
- int g0/0
- ip nat inside
- int g0/1
- ip nat inside
- int g0/2
- ip nat outside
-
- router rip
- version 2
- network 192.168.254.0
- network 192.168.253.0
- passive-interface g0/3
-
- router rip
- version 2
- default-information originate

- Use the command #nano ansible.cfg to create an Ansible config file with the following text:



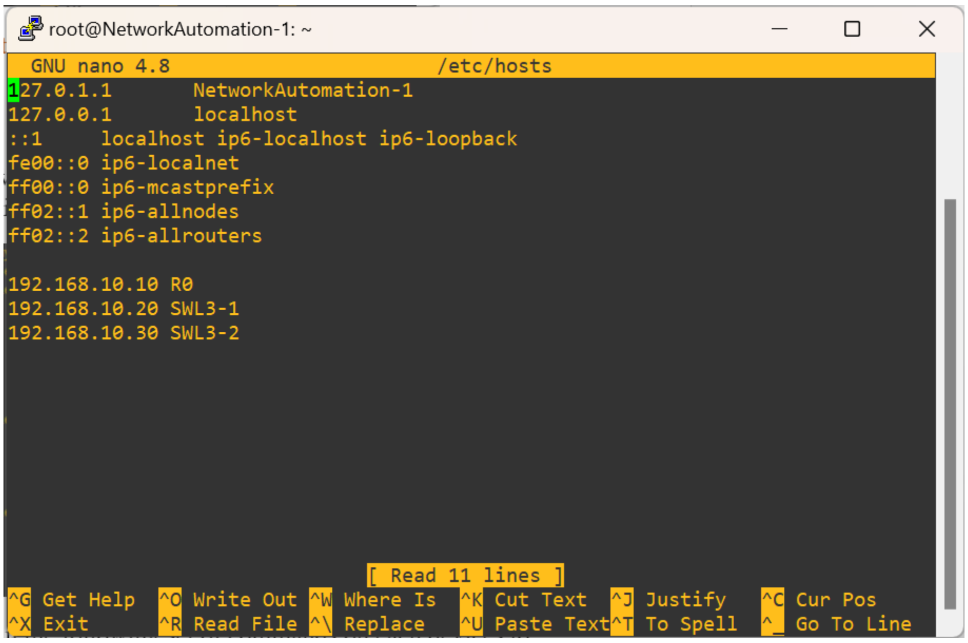
```
root@NetworkAutomation-1: ~
  GNU nano 4.8                    ansible.cfg                     Modified
[defaults]
inventory = ./myhosts
host_key_checking = false
timeout = 15
deprecation_warnings=False

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

- Use the command #nano myhosts to create a hosts list and enter the following groups containing the devices:


root@NetworkAutomation-1: ~

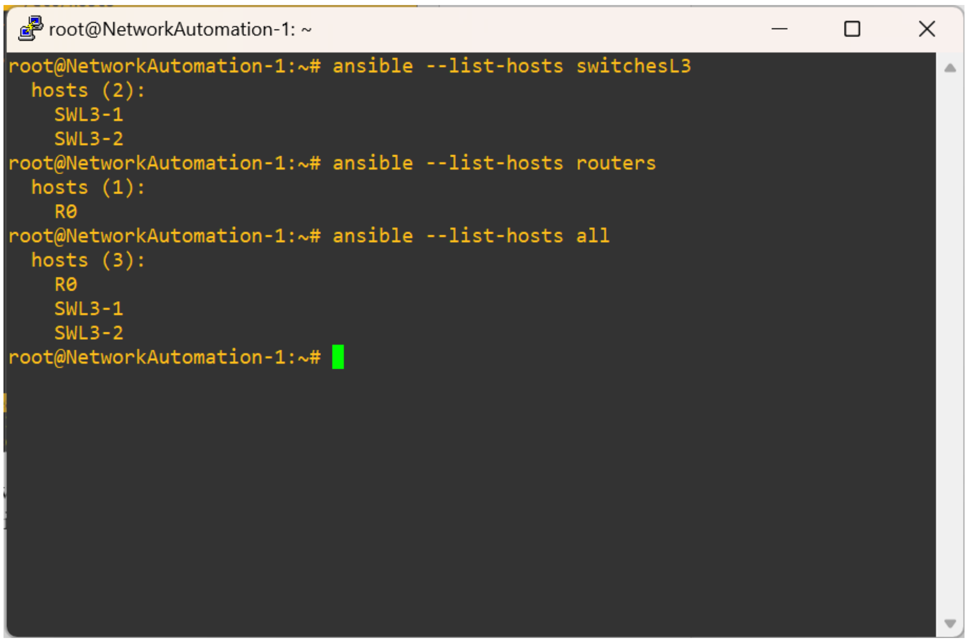
```
  GNU nano 4.8                        myhosts
[routers]
R0

[switchesL3]
SWL3-1
SWL3-2




                              [ Read 6 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

- Use the command #nano /etc/hosts to edit the /etc/hosts file and add the devices' IP addresses:


root@NetworkAutomation-1: ~

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

192.168.10.10 R0
192.168.10.20 SWL3-1
192.168.10.30 SWL3-2


                             [ Read 11 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

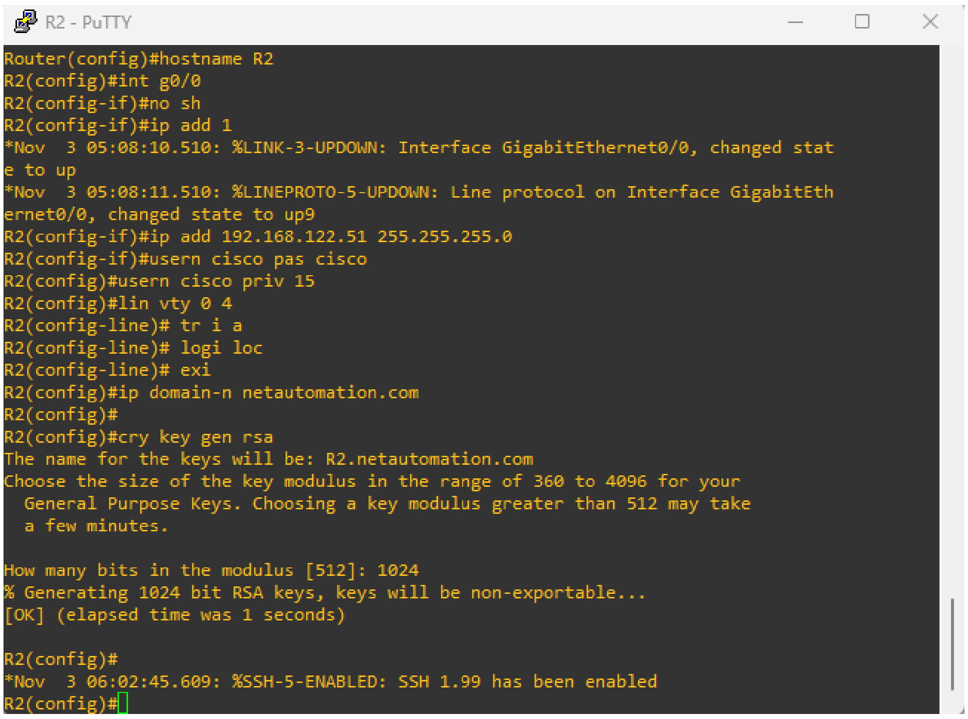
- The NetworkAutomation can now use the command #ansible –list-hosts all/routers/switches to check the list of devices that was added:



root@NetworkAutomation-1: ~

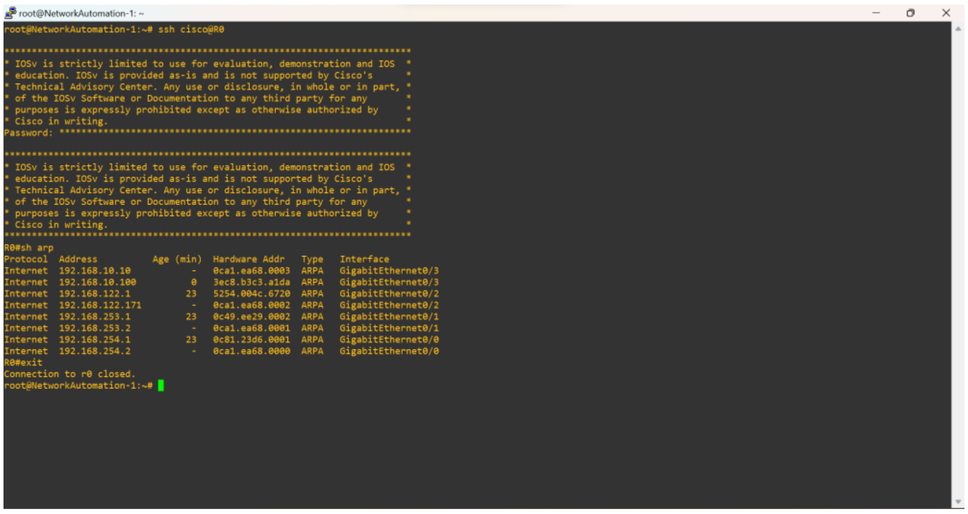
```
root@NetworkAutomation-1:~# ansible --list-hosts switchesL3
  hosts (2):
    SWL3-1
    SWL3-2
root@NetworkAutomation-1:~# ansible --list-hosts routers
  hosts (1):
    R0
root@NetworkAutomation-1:~# ansible --list-hosts all
  hosts (3):
    R0
    SWL3-1
    SWL3-2
root@NetworkAutomation-1:~#
```

- SSH Password Configuration: Run the following set of commands on each devices to configure SSH password protection with RSA1024 security (Password: ***cisco***)
  - usern cisco pas cisco
  - usern cisco priv 15
  - lin vty 0 4
  - tr i a
  - logi loc
  - exi
  - ip domain-n netautomation.com
  -
  - cry key gen rsa
  - 1024



```
Router(config)#hostname R2
R2(config)#int g0/0
R2(config-if)#no sh
R2(config-if)#ip add 1
*Nov  3 05:08:10.510: %LINK-3-UPDOWN: Interface GigabitEthernet0/0, changed stat
e to up
*Nov  3 05:08:11.510: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEth
ernet0/0, changed state to up9
R2(config-if)#ip add 192.168.122.51 255.255.255.0
R2(config-if)#usern cisco pas cisco
R2(config)#usern cisco priv 15
R2(config)#lin vty 0 4
R2(config-line)# tr i a
R2(config-line)# logi loc
R2(config-line)# exi
R2(config)#ip domain-n netautomation.com
R2(config)#
R2(config)#cry key gen rsa
The name for the keys will be: R2.netautomation.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)

R2(config)#
*Nov  3 06:02:45.609: %SSH-5-ENABLED: SSH 1.99 has been enabled
R2(config)#
```

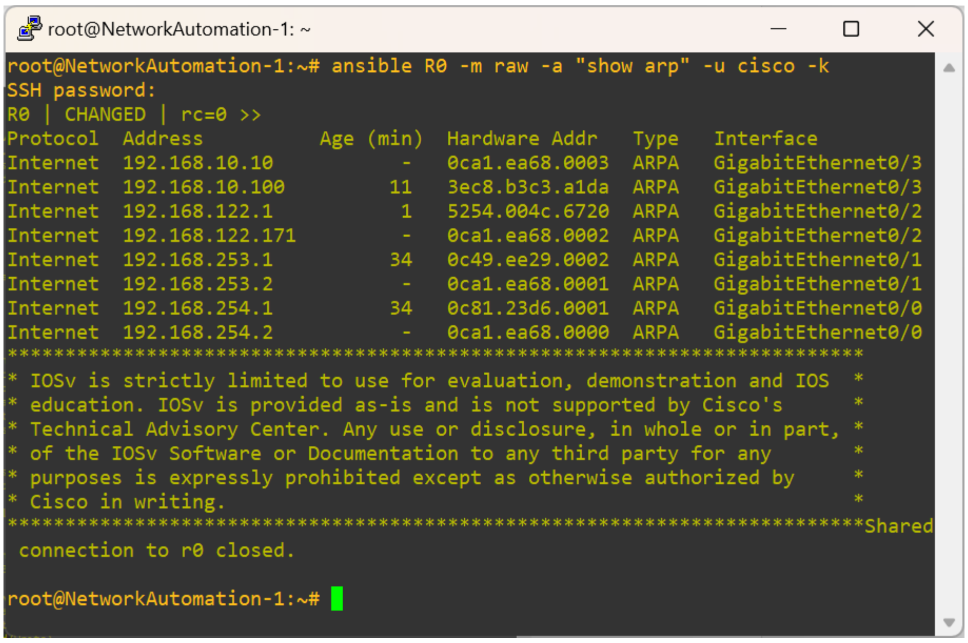
- R0, SWL3-1, SWL3-2 can now be remotely accessed through the NetworkAutomation-1 device with commands:



```
root@NetworkAutomation-1:~# ssh cisco@R0

**************************************************************************
* IOSv is strictly limited to use for evaluation, demonstration and IOS  *
* education. IOSv is provided as-is and is not supported by Cisco's      *
* Technical Advisory Center. Any use or disclosure, in whole or in part, *
* of the IOSv Software or Documentation to any third party for any       *
* purposes is expressly prohibited except as otherwise authorized by     *
* Cisco in writing.                                                      *
Password: ****************************************************************

**************************************************************************
* IOSv is strictly limited to use for evaluation, demonstration and IOS  *
* education. IOSv is provided as-is and is not supported by Cisco's      *
* Technical Advisory Center. Any use or disclosure, in whole or in part, *
* of the IOSv Software or Documentation to any third party for any       *
* purposes is expressly prohibited except as otherwise authorized by     *
* Cisco in writing.                                                      *
**************************************************************************
R0#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.10.10           -   0ca1.ea68.0003  ARPA   GigabitEthernet0/3
Internet  192.168.10.100          0   3ec8.b3c3.a1da  ARPA   GigabitEthernet0/3
Internet  192.168.122.1          23   5254.004c.6720  ARPA   GigabitEthernet0/2
Internet  192.168.122.171         -   0ca1.ea68.0002  ARPA   GigabitEthernet0/2
Internet  192.168.253.1          23   0c49.ee29.0002  ARPA   GigabitEthernet0/1
Internet  192.168.253.2           -   0ca1.ea68.0001  ARPA   GigabitEthernet0/1
Internet  192.168.254.1          23   0c81.23d6.0001  ARPA   GigabitEthernet0/0
Internet  192.168.254.2           -   0ca1.ea68.0000  ARPA   GigabitEthernet0/0
R0#exit
Connection to r0 closed.
root@NetworkAutomation-1:~#
```

III. Results

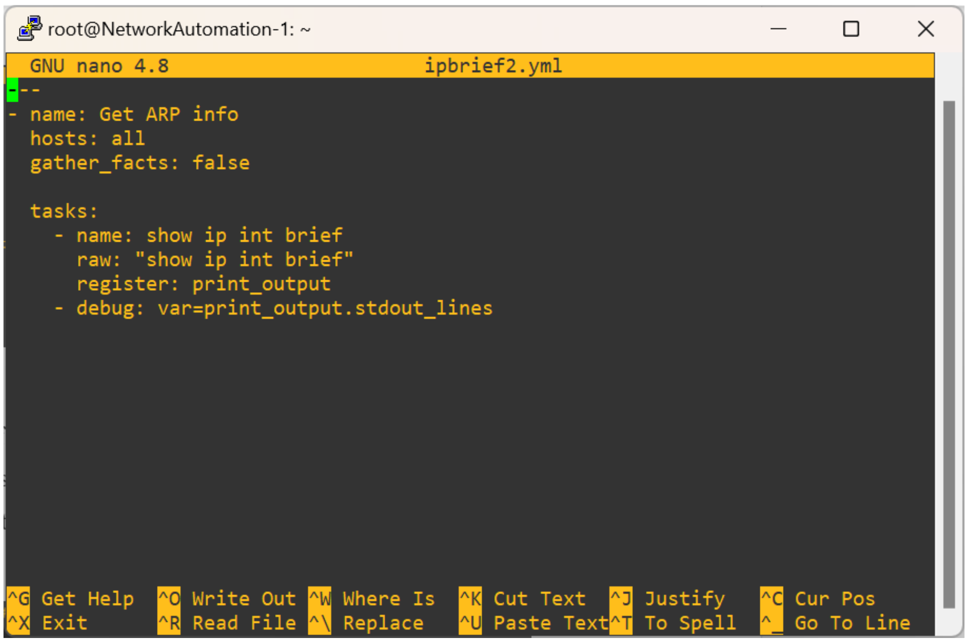
1. Ansible Ad Hoc



```
root@NetworkAutomation-1:~# ansible R0 -m raw -a "show arp" -u cisco -k
SSH password:
R0 | CHANGED | rc=0 >>
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.10.10           -   0ca1.ea68.0003  ARPA   GigabitEthernet0/3
Internet  192.168.10.100         11   3ec8.b3c3.a1da  ARPA   GigabitEthernet0/3
Internet  192.168.122.1           1   5254.004c.6720  ARPA   GigabitEthernet0/2
Internet  192.168.122.171         -   0ca1.ea68.0002  ARPA   GigabitEthernet0/2
Internet  192.168.253.1          34   0c49.ee29.0002  ARPA   GigabitEthernet0/1
Internet  192.168.253.2           -   0ca1.ea68.0001  ARPA   GigabitEthernet0/1
Internet  192.168.254.1          34   0c81.23d6.0001  ARPA   GigabitEthernet0/0
Internet  192.168.254.2           -   0ca1.ea68.0000  ARPA   GigabitEthernet0/0
**************************************************************************
* IOSv is strictly limited to use for evaluation, demonstration and IOS  *
* education. IOSv is provided as-is and is not supported by Cisco's      *
* Technical Advisory Center. Any use or disclosure, in whole or in part, *
* of the IOSv Software or Documentation to any third party for any       *
* purposes is expressly prohibited except as otherwise authorized by     *
* Cisco in writing.                                                      *
**************************************************************************Shared
 connection to r0 closed.

root@NetworkAutomation-1:~#
```

- The syntax would start with ‘ansible’ and the device which will be applied to, in this case is ‘ansible R0’. The ‘–m’ argument stands for module, ‘-a’ stands for action, ‘-u’ is the username and ‘-k’ option will prompt for password if the command is executed.

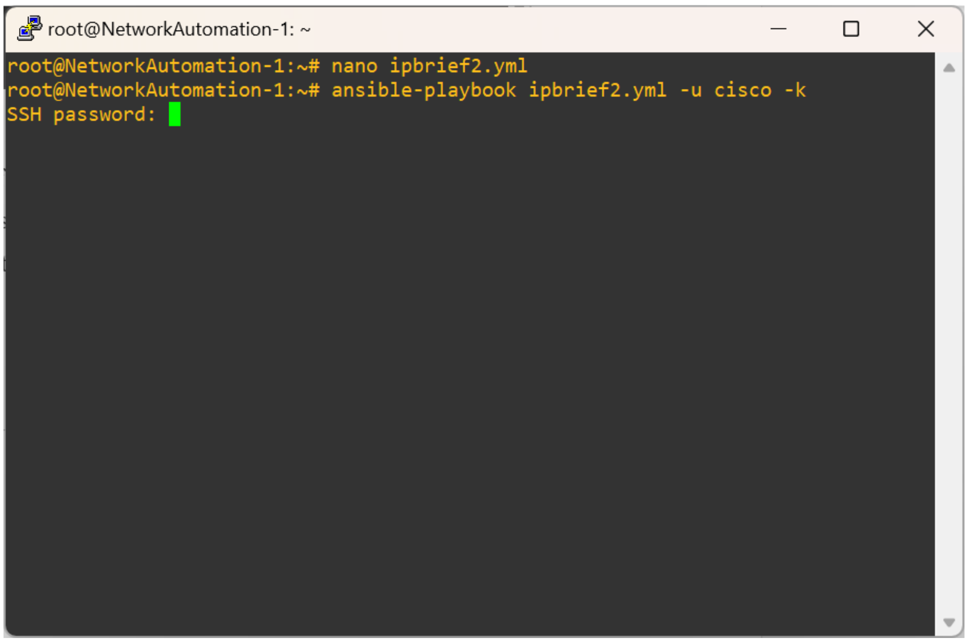
2. Ansible Playbook



root@NetworkAutomation-1: ~

```
  GNU nano 4.8                         ipbrief2.yml
---
- name: Get ARP info
  hosts: all
  gather_facts: false

  tasks:
    - name: show ip int brief
      raw: "show ip int brief"
      register: print_output
    - debug: var=print_output.stdout_lines











^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

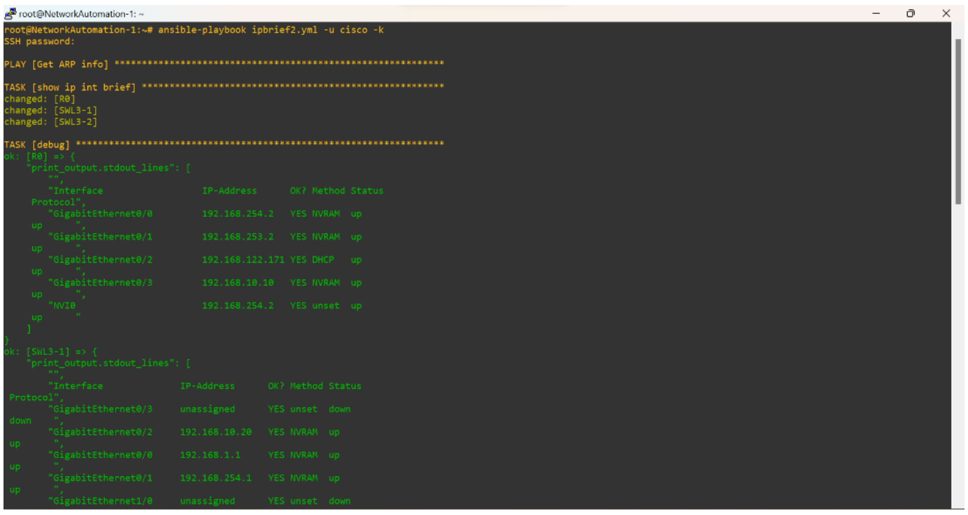
- The file above is a .yaml file called "ipbrief2.yml". On the top section, we define the group of hosts on which to run the playbook and its name. After that, we define a list of tasks. Each of the tasks contains some information about the task and the module to be executed along with the necessary arguments.



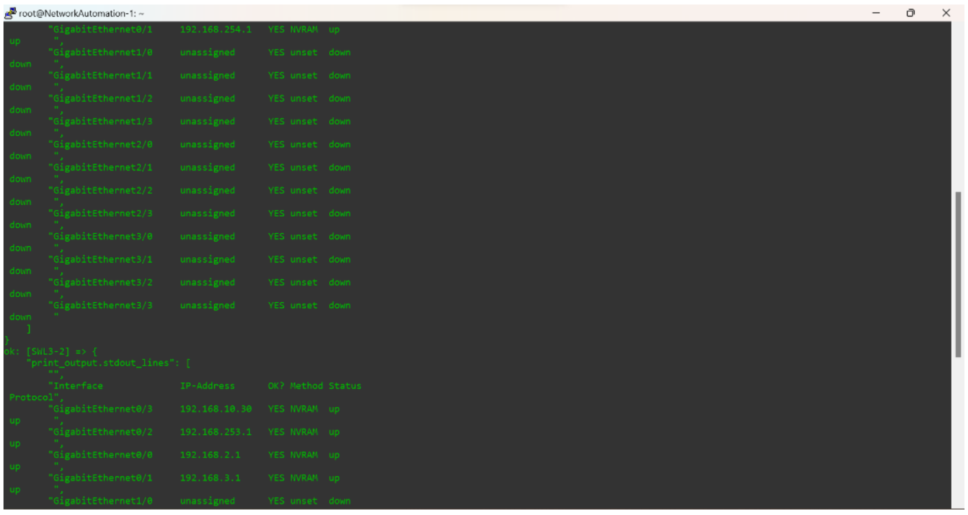
root@NetworkAutomation-1: ~

```
root@NetworkAutomation-1:~# nano ipbrief2.yml
root@NetworkAutomation-1:~# ansible-playbook ipbrief2.yml -u cisco -k
SSH password:
```

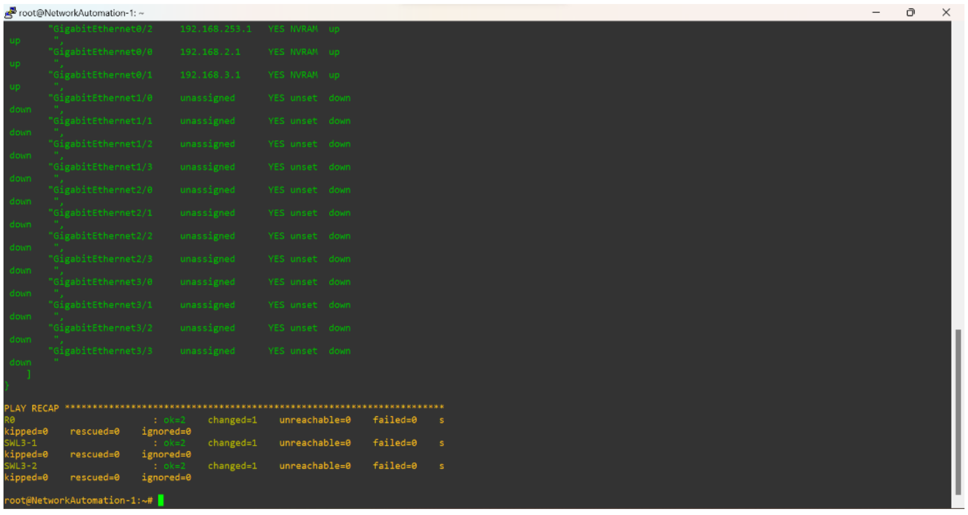
- So now we could use the syntax above to run playbook with the corresponding yaml file.



```
root@NetworkAutomation-1: ~
root@NetworkAutomation-1:~# ansible-playbook ipbrief2.yml -u cisco -k
SSH password:

PLAY [Get ARP info] *****************************************************************

TASK [show ip int brief] ************************************************************
changed: [R0]
changed: [SWL3-1]
changed: [SWL3-2]

TASK [debug] ************************************************************************
ok: [R0] => {
    "print_output.stdout_lines": [
        "",
        "Interface                  IP-Address      OK? Method Status
     Protocol",
        "GigabitEthernet0/0         192.168.254.2   YES NVRAM  up
     up      ",
        "GigabitEthernet0/1         192.168.253.2   YES NVRAM  up
     up      ",
        "GigabitEthernet0/2         192.168.122.171 YES DHCP   up
     up      ",
        "GigabitEthernet0/3         192.168.10.10   YES NVRAM  up
     up      ",
        "NVI0                       192.168.254.2   YES unset  up
     up      "
    ]
}
ok: [SWL3-1] => {
    "print_output.stdout_lines": [
        "",
        "Interface              IP-Address      OK? Method Status
 Protocol",
        "GigabitEthernet0/3     unassigned      YES unset  down
 down   ",
        "GigabitEthernet0/2     192.168.10.20   YES NVRAM  up
 up     ",
        "GigabitEthernet0/0     192.168.1.1     YES NVRAM  up
 up     ",
        "GigabitEthernet0/1     192.168.254.1   YES NVRAM  up
 up     ",
        "GigabitEthernet1/0     unassigned      YES unset  down
```



```
root@NetworkAutomation-1: ~
        "GigabitEthernet0/1     192.168.254.1   YES NVRAM  up
 up     ",
        "GigabitEthernet1/0     unassigned      YES unset  down
 down   ",
        "GigabitEthernet1/1     unassigned      YES unset  down
 down   ",
        "GigabitEthernet1/2     unassigned      YES unset  down
 down   ",
        "GigabitEthernet1/3     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/0     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/1     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/2     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/3     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/0     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/1     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/2     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/3     unassigned      YES unset  down
 down   "
    ]
}
ok: [SWL3-2] => {
    "print_output.stdout_lines": [
        "",
        "Interface              IP-Address      OK? Method Status
 Protocol",
        "GigabitEthernet0/3     192.168.10.30   YES NVRAM  up
 up     ",
        "GigabitEthernet0/2     192.168.253.1   YES NVRAM  up
 up     ",
        "GigabitEthernet0/0     192.168.2.1     YES NVRAM  up
 up     ",
        "GigabitEthernet0/1     192.168.3.1     YES NVRAM  up
 up     ",
        "GigabitEthernet1/0     unassigned      YES unset  down
```



```
root@NetworkAutomation-1: ~
        "GigabitEthernet0/2     192.168.253.1   YES NVRAM  up
 up     ",
        "GigabitEthernet0/0     192.168.2.1     YES NVRAM  up
 up     ",
        "GigabitEthernet0/1     192.168.3.1     YES NVRAM  up
 up     ",
        "GigabitEthernet1/0     unassigned      YES unset  down
 down   ",
        "GigabitEthernet1/1     unassigned      YES unset  down
 down   ",
        "GigabitEthernet1/2     unassigned      YES unset  down
 down   ",
        "GigabitEthernet1/3     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/0     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/1     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/2     unassigned      YES unset  down
 down   ",
        "GigabitEthernet2/3     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/0     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/1     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/2     unassigned      YES unset  down
 down   ",
        "GigabitEthernet3/3     unassigned      YES unset  down
 down   "
    ]
}

PLAY RECAP **************************************************************************
R0                         : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
SWL3-1                     : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
SWL3-2                     : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

root@NetworkAutomation-1:~#
```

3. Conclusion

- Network automation helps reduce the risk of misconfiguration by eliminating the repetitive manual tasks and it also saves time and money. It makes changes faster, has better scaling and builds a reliable network.
- In this report, Ansible and GNS3 are presented platforms that support an effective workplace for network management and automation from remote devices with vivid symbols and features that allow various modules, image files to be installed and imported for designing network topologies. The network automation is carried out mainly by manipulating with console applications and yaml language, which makes it easy to interact with each device in a system.

Appendix

1. Self-evaluation

Report format	Presentation	Theory	Demonstration
1 point	1 point	2 point	4.5 point

2. Task assignment

Members	Tasks	Percentage of completion
Huỳnh Anh Nguyễn	Implementation, configuration for Ansible and presentation	100%
Bùi Nguyên Phúc	Implementation, configuration, demonstration for Ansible and presentation	100%
Trần Tấn Hải	Report, theory, configuration, and write playbooks for Ansible and presentation	100%

3. Questions and answers

Q. Why do we need to create groups in inventory file in ansible?

A. For easier management, for example: when we have many subnets, we should create a group for each subnet so that when we want to automate a specific subnet, we only need to target the group that contains all the devices in the targeted subnet (or subnets).

Q. Why did we use GNS3 instead of others app?

A. Because GNS3 is one of the most popular automation tools, it is easy to use and has a big community for answering any questions we may have counter. And it has many tutorials online.

Q. What are the requirements for node devices and IOS?

A. Our project is about network automation, so we only concern about the network devices like switch layer 2/layer 3, router and others network devices. And all the nodes are required to have SSH services enabled.

Q. Why we should not use ansible for configuring the network devices?

A. Because the main purpose of network automation is to do the repetitive tasks automatically. But you only need to configure the network devices one time, and each time you configure it, it will be different, it won't be the same. So, it is not highly recommended using ansible for automatic configuration.

Q. Explain the error that we encountered when presentation (with the given cause: the network)

A. So, the error that we encountered in the presentation is that the PC0 ping 8.8.8.8 with only 50% success. The reason is that the hardware of the laptop that demonstrated it, the internet connection is not very good so that when it takes too long to get response from 8.8.8.8 will count that as not receive. So that is why we only have 50% success rate.

Q. If there is a large network can ansible have performance issues? If so, please choose a solution?

A. No there won't be any issues when using ansible in a larger network. There are a couple of reasons for that. Firstly, Ansible uses a technology called **Automation mesh** is an overlay network intended to ease the distribution of automation across a collection of execution nodes using existing connectivity. Secondly, Ansible is agentless, so it won't need to install any monitoring software on the network devices. Thirdly, Ansible will send the automation tasks to the network devices to execute those tasks.

Q. We already have configured the SSH, so can other people connect with SSH to the network automation device?

A. No, other people can't connect with SSH to the network automation device because we only configured SSH service on the routers and switches (managed node). Ansible use SSH to connect to the managed nodes, so it does not require to configure SSH services on the network automation devices.

Q. How can network automation tools contribute to maintaining a secure network infrastructure?

A. One of the most popular risks in a network is caused by misconfiguration, human error. So, by automating repetitive tasks we can minimize these risks and save time, money, …

Q. In deploying network automation with Ansible, how can we maintain flexibility and extensibility when working with many providers (network devices, technology)?

A. Ansible has many tools for these problems. First, it has inventory file here you can create many groups each group have many devices, and each group represent a zone that we define it as we like. Second, Ansible is agentless, so it won't need any monitor software on the managed nodes, it only needs SSH to connect to managed nodes.

Q. Is there a limit to how many devices are in the network?

A. Yes there is a limit to how many devices are in the network. It depends on the hardware of the automation devices. Because Ansible need a connection between devices so that it can send the automation tasks to managed node.

Q. How to ensure security when using Ansible?

A. Ansible has many tools for these problems. First, it uses SSH for communication between devices so it can ensure that data travel between devices is secure (cause SSH basically stands for secure socket shell). Second, it has vault technology. This is the way to encrypt sensitive data in the automation task for example password, etc.