Class 1 - Nornir Fundamentals

mardi, 27 août 2019

14:30

[root@NetAutoServer Staging]# ls

defaults.yaml groups.yaml hosts.yaml nornir.log

[root@NetAutoServer Staging]# cat hosts.yaml

---

EDGE:

hostname: RTRA

groups:

- RTR

DC:

hostname: NXSWA

groups:

- SW

[root@NetAutoServer Staging]# cat groups.yaml

---

RTR:

platform: IOS

SW:

platform: NXOS

[root@NetAutoServer Staging]# cat defaults.yaml

---

username: cisco

password: cisco

[root@NetAutoServer Staging]# ipython

Python 3.6.8 (default, May 2 2019, 20:40:44)

Type 'copyright', 'credits' or 'license' for more information

IPython 7.7.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from nornir import InitNornir

In [2]: nr = InitNornir()

In [3]: nr.inventory

Out[3]: <nornir.core.inventory.Inventory at 0x7fcaef30ba48>

In [3]: nr.inventory.hosts

Out[3]: {'EDGE': Host: EDGE, 'DC': Host: DC}

In [7]: nr.inventory.groups

Out[7]: {'RTR': Group: RTR, 'SW': Group: SW}

In [6]: edge\_dev = nr.inventory.hosts['EDGE']

In [14]: edge\_group = nr.inventory.groups['RTR']

In [6]: edge\_dev.hostname

Out[6]: 'RTRA'

In [12]: edge\_dev.platform

Out[12]: 'IOS'

In [14]: edge\_dev.username

Out[14]: 'cisco'

In [15]: edge\_dev.password

Out[15]: 'cisco'

Lets create a simple task

[root@AnsibleServer Staging]# cat task1.py

def task1(task):

print(task.host)

def main():

nr = InitNornir()

nr.run(task=task1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

[root@AnsibleServer Staging]# python task1.py

EDGE

DC

Now let see what happens if task can get executed randomly on the hosts defined

[root@AnsibleServer Staging]# cat task1.py

import random

import time

from nornir import InitNornir

def task1(task):

time.sleep(random.random())

print(task.host)

def main():

nr = InitNornir()

nr.run(task=task1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

[root@AnsibleServer Staging]# python task1.py

DC

EDGE

[root@AnsibleServer Staging]# python task1.py

EDGE

DC

You can see the tasks are executed randomly and not in sequential order.

To make the task get executed in sequential order of the hosts defined even with the random execution defined, we need to specify the "num\_workers" : 1 as shown below.

[root@AnsibleServer Staging]# cat task1.py

import random

import time

from nornir import InitNornir

def task1(task):

time.sleep(random.random())

print(task.host)

def main():

nr = InitNornir(core={"num\_workers": 1})

nr.run(task=task1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

[root@AnsibleServer Staging]# python task1.py

EDGE

DC

[root@AnsibleServer Staging]# python task1.py

EDGE

DC

Class 2 - Configuration Options and Netmiko plugins

jeudi, 12 septembre 2019

13:13

* 1. First, You need to have a inventory file, groups file and the defaults file.

root@NornirServer:~/Nornir# cat defaults.yaml

---

username: 'cisco'

password: '123!Cisco'

root@NornirServer:~/Nornir# cat groups.yaml

---

L2:

platform: cisco\_ios

root@NornirServer:~/Nornir# cat inventory.yaml

---

cat29SW1:

hostname: L2SWA

* 1. You need to have a config.yaml file or you should have the ENV variable set for the nornir to know where to find the inventory and host file and groups file.

Here I am showing that I have set that in the environment variable.

root@NornirServer:~/Nornir/Class2# printenv | grep NORNIR

NORNIR\_INVENTORY\_PLUGIN=nornir.plugins.inventory.simple.SimpleInventory

NORNIR\_INVENTORY\_OPTIONS={"host\_file": "/root/Nornir/inventory.yaml","group\_file": "/root/Nornir/groups.yaml","defaults\_file": "/root/Nornir/defaults.yaml"}

To set the env variable as shown above you need to do

export NORNIR\_INVENTORY\_PLUGIN=nornir.plugins.inventory.simple.SimpleInventory

export NORNIR\_INVENTORY\_OPTIONS='{"host\_file": "/root/Nornir/inventory.yaml","group\_file": "/root/Nornir/groups.yaml","defaults\_file": "/root/Nornir/defaults.yaml"}'

The problem with setting the options in ENV variable is, the env variable will be lost when the system is restarted. So need to find a way to permanently set this options in the ENV variable.

Or You can also specify this in a config.yaml file but note that instead of task.host, you need to use task.host.hostname in your scripts.

root@NornirServer:~/Nornir/# cat config.yaml

---

inventory:

plugin: nornir.plugins.inventory.simple.SimpleInventory

options:

host\_file: inventory.yaml

group\_file: groups.yaml

defaults\_file: defaults.yaml

Core:

num\_workers: 10

logging:

file: ""

* 1. Now let us try to run a simple python script against the inventory we have got. This script will display the output of "show ip int brief" command with the filter to show the interface with the IP 192.168

root@NornirServer:~/Nornir/Class2# cat netmiko\_sh\_ip\_int.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir()

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

Print()

root@NornirServer:~/Nornir/Class2# python netmiko\_sh\_ip\_int.py

cat29SW1

------------------------------

Vlan1 192.168.122.243 YES NVRAM up up

------------------------------

* 1. Now I am going to expand the inventory with a Nexus switch, L3 switch, L2 switch and a Router. Since the same command cannot give the output for Nexus due to the management vrf in Nexus, the output will be shown empty.

root@NornirServer:~/Nornir# cat inventory.yaml

---

L2SW:

hostname: L2SWA

groups:

- L2

RTR:

hostname: RTRA

groups:

- Edge

NX:

hostname: DCSW1

groups:

- DC

L3SW:

hostname: L3SWA

groups:

- L3

root@NornirServer:~/Nornir#

root@NornirServer:~/Nornir# cat groups.yaml

---

L2:

platform: cisco\_ios

Edge:

platform: cisco\_ios

DC:

platform: cisco\_nxos

L3:

platform: cisco\_ios

root@NornirServer:~/Nornir/Class2# cat int\_brief.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir()

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@NornirServer:~/Nornir/Class2# python int\_brief.py

L2SW

------------------------------

Vlan1 192.168.122.243 YES NVRAM up up

------------------------------

RTR

------------------------------

GigabitEthernet0/0 192.168.122.245 YES NVRAM up up

------------------------------

NX

------------------------------

------------------------------

L3SW

------------------------------

Vlan1 192.168.122.244 YES NVRAM up up

------------------------------

root@NornirServer:~/Nornir/Class2#

* 1. If you would like to get a output of a "sh run" with an admin account that has less privileges, then you need to get into enable mode and you need enable secret/password. For example I am going to create a test user with less privilege and configure a enable secret. I am going to do this only one device in the inventory as the process is similar on all devices.

I created a level 1 user account as the following

root@Nornir:~/Nornir/Class2# cat ../defaults.yaml

---

username: l1support

password: cisco

You can see below when I try to run the following script, it gives me an error because I used privilege level 1 user account to look at the sh run which is not allowed without the enable mode. Additionaly I am going to use print\_result plugin which gives the output in a different format. Though the format is not great it is useful at times.

root@Nornir:~/Nornir/Class2# cat sh\_run.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.plugins.functions.text import print\_result

nr = InitNornir()

results = nr.run(task = netmiko\_send\_command, command\_string = "show run int vlan1")

print\_result(results)

root@Nornir:~/Nornir/Class2# python sh\_run.py

netmiko\_send\_command\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* L2SW \*\* changed : False \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vvvv netmiko\_send\_command \*\* changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

^

% Invalid input detected at '^' marker.

^^^^ END netmiko\_send\_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Now I am going to add the enable password in the groups.yaml and the enable mode in the script

root@Nornir:~/Nornir/Class2# cat ../groups.yaml

---

L2:

platform: cisco\_ios

connection\_options:

netmiko:

extras:

secret: cisco

root@Nornir:~/Nornir/Class2# cat sh\_run.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.plugins.functions.text import print\_result

nr = InitNornir()

results = nr.run(task = netmiko\_send\_command, enable=True, command\_string = "show run int vlan1")

print\_result (results)

root@Nornir:~/Nornir/Class2# python sh\_run.py

netmiko\_send\_command\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* L2SW \*\* changed : False \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vvvv netmiko\_send\_command \*\* changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

Building configuration...

Current configuration : 65 bytes

!

interface Vlan1

ip address 192.168.122.243 255.255.255.0

end

^^^^ END netmiko\_send\_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

* 1. In this example, I am going to save the config of 3 devices in the inventory L2SW, L3SW, RTR.

root@Nornir:~/Nornir/Class2# cat wr\_mem.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_save\_config

from nornir.plugins.functions.text import print\_result

nr = InitNornir()

results = nr.run(task = netmiko\_save\_config)

print\_result (results)

root@Nornir:~/Nornir/Class2# python wr\_mem.py

netmiko\_save\_config\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* L2SW \*\* changed : True \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vvvv netmiko\_save\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

Building configuration...

Compressed configuration from 3714 bytes to 1700 bytes[OK]

^^^^ END netmiko\_save\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\* L3SW \*\* changed : True \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vvvv netmiko\_save\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

Building configuration...

Compressed configuration from 3656 bytes to 1673 bytes[OK]

^^^^ END netmiko\_save\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\* RTR \*\* changed : True \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vvvv netmiko\_save\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

Building configuration...

[OK]

^^^^ END netmiko\_save\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

* 1. Now lets us use napalm plugin instead of netmiko to achieve the desired things. For example, I would like to know the mac address table for L2SW, L3SW, RTR, Nexus. The nice part with napalm is the output is a commonly structured data for all kinds of operating system. In terms of inventory, the change is on the groups.yaml file. Napalm uses the platform type as ios,nxos,eos. In netmiko, the platform is specified as cisco\_ios, cisco\_nxos. Also for nxos we use api instead of SSH.

First we need to configure the api related features in the NXOS box.

DCSW1#

feature nxapi

nxapi https port 8443

nxapi sandbox

root@Nornir:~/Nornir# cat groups.yaml

---

L2:

platform: ios

Edge:

platform: ios

DC:

platform: nxos

port: 8443

L3:

platform: ios

root@Nornir:~/Nornir/Class2# cat napalm\_getters.py

import requests

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get

from nornir.plugins.functions.text import print\_result

from pprint import pprint

from urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable\_warnings(category=InsecureRequestWarning)

nr = InitNornir()

results = nr.run(task = napalm\_get, getters=["mac\_address\_table"])

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

pprint (v[0].result)

print ("-" \* 30)

print ()

print()

root@Nornir:~/Nornir/Class2# python napalm\_getters.py

L2SW

------------------------------

{'mac\_address\_table': [{'active': True,

'interface': 'Gi0/1',

'last\_move': -1.0,

'mac': '0C:F3:6F:34:7D:00',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/1',

'last\_move': -1.0,

'mac': '0C:F3:6F:9F:80:01',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/1',

'last\_move': -1.0,

'mac': '0C:F3:6F:9F:EC:00',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/3',

'last\_move': -1.0,

'mac': '0C:F3:6F:DF:00:00',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/0',

'last\_move': -1.0,

'mac': '6E:5C:91:B8:15:E7',

'moves': -1,

'static': False,

'vlan': 1}]}

------------------------------

RTR

------------------------------

{'mac\_address\_table': []}

------------------------------

NX

------------------------------

{'mac\_address\_table': [{'active': True,

'interface': 'sup-eth1(R)',

'last\_move': 0.0,

'mac': '0C:F3:6F:DF:00:2F',

'moves': 0,

'static': True,

'vlan': 0}]}

------------------------------

L3SW

------------------------------

{'mac\_address\_table': [{'active': True,

'interface': 'Gi0/1',

'last\_move': -1.0,

'mac': '0C:F3:6F:34:7D:00',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/0',

'last\_move': -1.0,

'mac': '0C:F3:6F:8C:74:01',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/0',

'last\_move': -1.0,

'mac': '0C:F3:6F:DF:00:00',

'moves': -1,

'static': False,

'vlan': 1},

{'active': True,

'interface': 'Gi0/0',

'last\_move': -1.0,

'mac': '6E:5C:91:B8:15:E7',

'moves': -1,

'static': False,

'vlan': 1}]}

------------------------------

root@Nornir:~/Nornir/Class2#

* 1. I am going to run getters against one specific host but I am going to combine multiple getters and specify which option to retrieve from which getter. In this example, I am going to "retrieve" the "startup" config from the "config" getter and also get the facts from the "facts getter". Here is the python script.

root@Nornir:~/Nornir/Class2# cat napalm\_getters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get

from pprint import pprint

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(hostname="RTRA")

results = nr.run(task = napalm\_get, getters=["config", "facts"] , getters\_options={"config": {"retrieve": "startup"}})

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

pprint (v[0].result)

print ("-" \* 30)

print ()

print()

root@Nornir:~/Nornir/Class2# python napalm\_getters.py

RTR

------------------------------

{'config': {'candidate': '',

'running': '',

'startup': 'Using 3204 out of 262144 bytes\n'

'!\n'

'version 15.6\n'

'service timestamps debug datetime msec\n'

'service timestamps log datetime msec\n'

'no service password-encryption\n'

'!\n'

'hostname RTRA\n'

'!\n'

'boot-start-marker\n'

'boot-end-marker\n'

'!\n'

'logging buffered 100000\n'

'!\n'

'no aaa new-model\n'

'end'},

'facts': {'fqdn': 'RTRA.vikiboy.net',

'hostname': 'RTRA',

'interface\_list': ['GigabitEthernet0/0',

'GigabitEthernet0/1',

'GigabitEthernet0/2',

'GigabitEthernet0/3'],

'model': 'IOSv',

'os\_version': 'IOSv Software (VIOS-ADVENTERPRISEK9-M), Version '

'15.6(2)T, RELEASE SOFTWARE (fc2)',

'serial\_number': '9HRUYI27C4B7T0GO42CGP',

'uptime': 16380,

'vendor': 'Cisco'}}

------------------------------

* 1. When a task against a specific host is failed, we can specify Nornir to either execute the other tasks against other host or can stop executing all the tasks against all the hosts. This is done with "raise\_on\_error" option the nornir core. You can set this under config file or can set it under env.

For testing, I have added a wrong password for the router under the groups.yaml

I have created a nornir.yaml file which is the config\_file and it has the "raise\_on\_error" options set. When this option is set as True, if the first task against one host fails, then the second task will not be executed against any hosts. In our case, if first task fails against the host RTR, first task will continue on other hosts but the second tasks will not get executed against any of the hosts.

root@Nornir:~/Nornir/Class2# cat ../nornir.yaml

---

core:

num\_workers: 10

raise\_on\_error: True

logging:

file: ""

root@Nornir:~/Nornir/Class2# cat failed\_tasks.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir(config\_file="/root/Nornir/nornir.yaml")

results = nr.run(task = netmiko\_send\_command, command\_string = "show run | inc logging")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

results = nr.run(task = netmiko\_send\_command, command\_string = "show run | inc hostname")

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

You can see when I execute the script, the second task that is "sh run | inc hostname" is not executed.

root@Nornir:~/Nornir/Class2# python failed\_tasks.py

Host 'RTR': task 'netmiko\_send\_command' failed with traceback:

paramiko.ssh\_exception.AuthenticationException: Authentication failed.

During handling of the above exception, another exception occurred:

netmiko.ssh\_exception.NetMikoAuthenticationException: Authentication failure: unable to connect cisco\_ios RTRA:22

Authentication failed.

Traceback (most recent call last):

File "failed\_tasks.py", line 6, in <module>

results = nr.run(task = netmiko\_send\_command, command\_string = "show run | inc logging")

File "/usr/local/lib/python3.6/dist-packages/nornir/core/task.py", line 226, in raise\_on\_error

raise NornirExecutionError(self)

nornir.core.exceptions.NornirExecutionError:

########################################

# L2SW (succeeded)

########################################

\*\*\*\* netmiko\_send\_command

logging buffered 100000

########################################

# RTR (failed)

########################################

\*\*\*\* netmiko\_send\_command

Authentication failure: unable to connect cisco\_ios RTRA:22

Authentication failed.

########################################

# L3SW (succeeded)

########################################

\*\*\*\* netmiko\_send\_command

logging buffered 100000

Now I am going to change the raise\_on\_error option set as false.

root@Nornir:~/Nornir/Class2# cat ../nornir.yaml

---

core:

raise\_on\_error: False

You can see when I executed the tasks, task1 failed against RTR so task2 did not execute against failed host but the task2 has got executed on all other host.

root@Nornir:~/Nornir/Class2# python failed\_tasks.py

Host 'RTR': task 'netmiko\_send\_command' failed with traceback:

paramiko.ssh\_exception.AuthenticationException: Authentication failed.

During handling of the above exception, another exception occurred:

netmiko.ssh\_exception.NetMikoAuthenticationException: Authentication failure: unable to connect cisco\_ios RTRA:22

Authentication failed.

L2SW

------------------------------

logging buffered 100000

------------------------------

L3SW

------------------------------

logging buffered 100000

------------------------------

L2SW

------------------------------

hostname L2SWA

------------------------------

L3SW

------------------------------

hostname L3SWA

------------------------------

* 1. As you have seen when the raise\_on\_error is set as False, the first task failed so the second task does not get executed on the failed host. This is not a ideal behaviour in all situations because first task failed but it does not mean the second task should not get executed against failed host. To avoid that, we do "on\_failed=True" on the second task in the script as shown below. So the second task will get executed against the failed host. It is irrelevant it works or not but still the execution try should happen.

root@Nornir:~/Nornir/Class2# cat failed\_tasks.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir(config\_file="/root/Nornir/nornir.yaml")

results = nr.run(task = netmiko\_send\_command, command\_string = "show run | inc logging")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

results = nr.run(task = netmiko\_send\_command, on\_failed = True, command\_string = "show run | inc hostname")

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

You can see below the second task gets executed against the failed host but still there is an error and you know because we have set the wrong password for testing purpose. I have the highlighted the output for the first failed task in yellow and second failed task in green. This concludes both tasks are executed against the failed host.

root@Nornir:~/Nornir/Class2# python failed\_tasks.py

Host 'RTR': task 'netmiko\_send\_command' failed with traceback:

paramiko.ssh\_exception.AuthenticationException: Authentication failed.

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

netmiko.ssh\_exception.NetMikoAuthenticationException: Authentication failure: unable to connect cisco\_ios RTRA:22

Authentication failed.

L2SW

------------------------------

logging buffered 100000

------------------------------

L3SW

------------------------------

logging buffered 100000

------------------------------

Host 'RTR': task 'netmiko\_send\_command' failed with traceback:

AttributeError: 'UnestablishedConnection' object has no attribute 'send\_command'

L2SW

------------------------------

hostname L2SWA

------------------------------

L3SW

------------------------------

hostname L3SWA

------------------------------

Class 3 - Inventory Filtering

lundi, 16 septembre 2019

11:27

1. In some case you do not want the tasks to get executed on tsome hosts or some group of hosts. In that case you use filter to remove those hosts/groups from the task execution. In this example, let us see how to configure such filters by specifying the task should run on specific group. Here we will be using the Filter object so we can do the filtering on complex data structures

Here is my groups file and config file that says where the inventory can be found.

root@Nornir:~/Nornir/Class2# cat ../groups.yaml

---

L2:

platform: ios

Edge:

platform: ios

DC:

platform: nxos

port: 8443

L3:

platform: ios

root@Nornir:~/Nornir/Class2# cat ../nornir.yaml

---

inventory:

plugin: nornir.plugins.inventory.simple.SimpleInventory

options:

host\_file: "/root/Nornir/inventory.yaml"

group\_file: "/root/Nornir/groups.yaml"

defaults\_file: "/root/Nornir/defaults.yaml"

core:

num\_workers: 10

raise\_on\_error: False

logging:

file: ""

Here is my python script that executes the task on specific host/groups.

root@Nornir:~/Nornir/Class2# cat filters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.core.filter import F

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="Edge"))

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@Nornir:~/Nornir/Class2# python filters.py

RTR

------------------------------

GigabitEthernet0/0 192.168.122.245 YES NVRAM up up

1. We can also specify execute the task on one or more groups. This is done by Union ie OR method

root@Nornir:~/Nornir/Class2# cat filters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.core.filter import F

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="Edge") | F(groups\_\_contains="L3"))

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@Nornir:~/Nornir/Class2# python filters.py

RTR

------------------------------

GigabitEthernet0/0 192.168.122.245 YES NVRAM up up

------------------------------

L3SW

------------------------------

Vlan1 192.168.122.244 YES NVRAM up up

1. There is one more way to specify that execute the task on all hosts/groups but not on specific host/groups ie Unary way of saying this. In our case I do not want the task to execute the task on the group "DC". So I specified that in the unary way.

root@Nornir:~/Nornir/Class2# cat filters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.core.filter import F

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(**~**F(groups\_\_contains="DC"))

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@Nornir:~/Nornir/Class2# python filters.py

L2SW

------------------------------

Vlan1 192.168.122.243 YES NVRAM up up

------------------------------

RTR

------------------------------

GigabitEthernet0/0 192.168.122.245 YES NVRAM up up

------------------------------

L3SW

------------------------------

Vlan1 192.168.122.244 YES NVRAM up up

------------------------------

1. You can specify the filters directly using the inbuilt core attributes like hostname, platform, port, username, password and name. For this you do not need to import the "F" which is the filter object.

root@Nornir:~/Nornir/Class2# cat filters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(hostname="L2SWA")

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@Nornir:~/Nornir/Class2# python filters.py

L2SW

------------------------------

Vlan1 192.168.122.243 YES NVRAM up up

------------------------------

1. Here we can also combine the core attributes as filter to execute the task on filtered host. For example, I am going to execute task on a host which has the platform as "ios" and the port set as 22.

Here is my group file and you can see the hosts with platform and host set with port 22.

root@Nornir:~/Nornir/Class2# cat ../groups.yaml

---

L2:

platform: ios

Edge:

platform: ios

port: 22

DC:

platform: nxos

port: 8443

L3:

platform: ios

root@Nornir:~/Nornir/Class2# cat filters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios").filter(port=22)

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@Nornir:~/Nornir/Class2# python filters.py

RTR

------------------------------

GigabitEthernet0/0 192.168.122.245 YES NVRAM up up

------------------------------

1. Now if I want to execute the task on all hosts which runs the operating system as "ios".

root@Nornir:~/Nornir/Class2# cat filters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios")

results = nr.run(task = netmiko\_send\_command, command\_string = "show ip int brief | inc 192.168")

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print ()

root@Nornir:~/Nornir/Class2# python filters.py

L2SW

------------------------------

Vlan1 192.168.122.243 YES NVRAM up up

------------------------------

RTR

------------------------------

GigabitEthernet0/0 192.168.122.245 YES NVRAM up up

------------------------------

L3SW

------------------------------

Vlan1 192.168.122.244 YES NVRAM up up

1. Here is another example of using filters with napalm getters. In this example, we are going to get the output of "show ip int brief" using napalm getters on the devices running the operating system as IOS.

root@Nornir:~/Nornir/Class2# cat napalm\_getters.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios")

results = nr.run(task = napalm\_get, getters=["interfaces\_ip"])

print ()

for k,v in results.items():

print (k)

print ("-" \* 30)

print (v[0].result)

print ("-" \* 30)

print ()

print()

root@Nornir:~/Nornir/Class2# python napalm\_getters.py

L2SW

------------------------------

{'interfaces\_ip': {'Vlan1': {'ipv4': {'192.168.122.243': {'prefix\_length': 24}}}}}

------------------------------

RTR

------------------------------

{'interfaces\_ip': {'GigabitEthernet0/0': {'ipv4': {'192.168.122.245': {'prefix\_length': 24}}}}}

------------------------------

L3SW

------------------------------

{'interfaces\_ip': {'Vlan1': {'ipv4': {'192.168.122.244': {'prefix\_length': 24}}}}}

------------------------------

Class 4 - File Copy and Configurations

lundi, 16 septembre 2019

22:19

* 1. We will see how to transfer files to network devices. For example purpose I will transfer files named as os.txt to the network devices. To start with I will transfer the file to one device and if that works, we can simulate the same for other devices.

Here is my inventory, group and the image file that I need to transfer

root@Nornir:~/Nornir/Class4# cat ../inventory.yaml

---

L2SW:

hostname: L2SWA.vikiboy.net

groups:

- L2

RTR:

hostname: RTRA.vikiboy.net

groups:

- EDGE

CUBE:

hostname: RTRB.vikiboy.net

groups:

- VOIP

DCSW:

hostname: DCSW1.vikiboy.net

groups:

- DC

L3SW:

hostname: L3SWA.vikiboy.net

groups:

- L3

root@Nornir:~/Nornir/Class4# cat ../groups.yaml

---

EDGE:

platform: ios

VOIP:

platform: ios

DC:

platform: nxos

port: 8443

L3:

platform: ios

L2:

platform: ios

root@Nornir:~/Nornir/images# tree

.

|-- DC

| `-- os.txt

|-- EDGE

| `-- os.txt

|-- L2

| `-- os.txt

|-- L3

| `-- os.txt

`-- VOIP

`-- os.txt

5 directories, 5 files

Here is my python script, the first task is to enable the scp server on the remote device, second task is to transfer the file, third task is to disable the scp server on the remote device. If the file does not exist on the remote flash, then we print out a message transfer successful. If the file already exist on the flash, then we print out a message stating file exists already.

root@Nornir:~/Nornir/Class4# cat filecopy.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_file\_transfer,netmiko\_send\_config

from nornir.core.filter import F

def scp\_enabl(task):

# enable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="ip scp server enable"

)

print()

print("-" \* 30)

print("\*\*\* SCP Server Enabled on", (task.host), "\*\*\*")

def file\_copy(task):

# Obtain the groups

group\_name = task.host.groups[0]

# Set the file based on groups

base\_file = "os.txt"

src\_file = "/root/Nornir/images/"f"{group\_name}/{base\_file}"

dst\_file = base\_file

# Transfer the file

results = task.run(

netmiko\_file\_transfer,

source\_file = src\_file,

dest\_file = dst\_file,

direction = "put"

)

# Print the status

print("-" \* 30)

print("On",task.host)

if results[0].changed is True:

print("File Transfer successful")

if results[0].changed is False:

print("File exists already")

print("-" \* 30)

def scp\_remov(task):

# disable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="no ip scp server enable"

)

print("\*\*\* SCP Server disabled on", (task.host), "\*\*\*")

print("-" \* 30)

print()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="EDGE"))

results = nr.run(task=scp\_enabl)

results = nr.run(task=file\_copy)

results = nr.run(task=scp\_remov)

I am going to run the script and it will get executed on the host which belongs to the group Edge. Before that I am going to check if scp is enabled or not. Also if the file exists on the remote device or not manually.

RTRA#dir flash:

Directory of flash0:/

1 drw- 0 Jan 30 2013 00:00:00 +00:00 boot

264 drw- 0 Oct 14 2013 00:00:00 +00:00 config

267 -rw- 143178592 Mar 22 2016 00:00:00 +00:00 vios-adventerprisek9-m

268 -rw- 524288 Nov 5 2018 16:18:48 +00:00 nvram

269 -rw- 79 Sep 13 2019 09:03:08 +00:00 e1000\_bia.txt

2142715904 bytes total (1994403840 bytes free)

RTRA#sh run | inc scp

RTRA#

You can see the scp server is not enabled and the file do not exist.

root@Nornir:~/Nornir/Class4# python filecopy.py

------------------------------

\*\*\* SCP Server Enabled on RTR \*\*\*

------------------------------

On RTR

File Transfer successful

------------------------------

\*\*\* SCP Server disabled on RTR \*\*\*

------------------------------

Now I am going to check manually if the script worked properly

RTRA#

\*Sep 16 20:58:44.353: %SYS-5-CONFIG\_I: Configured from console by cisco on vty0 (192.168.122.28)

RTRA#dir flash:

\*Sep 16 20:58:51.274: %SYS-5-CONFIG\_I: Configured from console by cisco on vty0 (192.168.122.28)

**RTRA#dir flash: | inc os.txt**

**272 -rw- 16 Sep 16 2019 20:58:46 +00:00 os.txt**

RTRA#sh run | inc scp

RTRA#

Since everything went well, now I am going to execute the same script against a group of device whose platform is "ios". For this I will changing the "nr.filter" to use the core attribute "platform"=ios".

Here is the python script.

root@Nornir:~/Nornir/Class4# cat filecopy.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_file\_transfer,netmiko\_send\_config

from nornir.core.filter import F

def scp\_enabl(task):

# enable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="ip scp server enable"

)

print("-" \* 30)

print("\*\*\* SCP Server Enabled on", (task.host), "\*\*\*")

def file\_copy(task):

# Obtain the groups

group\_name = task.host.groups[0]

# Set the file based on groups

base\_file = "os.txt"

src\_file = "/root/Nornir/images/"f"{group\_name}/{base\_file}"

dst\_file = base\_file

# Transfer the file

results = task.run(

netmiko\_file\_transfer,

source\_file = src\_file,

dest\_file = dst\_file,

direction = "put"

)

# Print the status

print("-" \* 30)

print("On",task.host)

if results[0].changed is True:

print("File Transfer successful")

if results[0].changed is False:

print("File exists already")

print("-" \* 30)

def scp\_remov(task):

# disable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="no ip scp server enable"

)

print("\*\*\* SCP Server disabled on", (task.host), "\*\*\*")

print("-" \* 30)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios")

print()

results = nr.run(task=scp\_enabl)

results = nr.run(task=file\_copy)

results = nr.run(task=scp\_remov)

print ()

root@Nornir:~/Nornir/Class4# python filecopy.py

------------------------------

\*\*\* SCP Server Enabled on CUBE \*\*\*

------------------------------

\*\*\* SCP Server Enabled on RTR \*\*\*

------------------------------

\*\*\* SCP Server Enabled on L3SW \*\*\*

------------------------------

\*\*\* SCP Server Enabled on L2SW \*\*\*

------------------------------

On RTR

File Transfer successful

------------------------------

------------------------------

On CUBE

File Transfer successful

------------------------------

------------------------------

On L3SW

File Transfer successful

------------------------------

------------------------------

On L2SW

File Transfer successful

------------------------------

\*\*\* SCP Server disabled on RTR \*\*\*

------------------------------

\*\*\* SCP Server disabled on CUBE \*\*\*

------------------------------

\*\*\* SCP Server disabled on L3SW \*\*\*

------------------------------

\*\*\* SCP Server disabled on L2SW \*\*\*

-----------------------------

We can check manually if the file has been transferred and the scp has been disabled. As you can see below the file is transferred and scp is disabled as expected

RTRA#dir flash: | inc os.txt

272 -rw- 16 Sep 16 2019 21:31:36 +00:00 os.txt

RTRA#sh run | inc scp

RTRA#

L3SWA#dir flash: | inc os.txt

270 -rw- 17 Sep 16 2019 21:10:42 +00:00 os.txt

L3SWA#sh run | inc scp

L3SWA

If I run the same script again, you will see file exist already message because the file has been already in place on the flash of the remote device.

root@Nornir:~/Nornir/Class4# python filecopy.py

------------------------------

\*\*\* SCP Server Enabled on CUBE \*\*\*

------------------------------

\*\*\* SCP Server Enabled on RTR \*\*\*

------------------------------

\*\*\* SCP Server Enabled on L3SW \*\*\*

------------------------------

\*\*\* SCP Server Enabled on L2SW \*\*\*

------------------------------

On RTR

File exists already

------------------------------

------------------------------

On CUBE

File exists already

------------------------------

------------------------------

On L3SW

File exists already

------------------------------

------------------------------

On L2SW

File exists already

------------------------------

\*\*\* SCP Server disabled on CUBE \*\*\*

------------------------------

\*\*\* SCP Server disabled on RTR \*\*\*

------------------------------

\*\*\* SCP Server disabled on L3SW \*\*\*

------------------------------

\*\*\* SCP Server disabled on L2SW \*\*\*

------------------------------

root@Nornir:~/Nornir/Class4#

* 1. We have seen that if the file exists we receive a message stating the "file exists already" if content in the file is same. Now let see what happens if the file name is same but the content of the file is different. For example, I am going to check what is there in the content of the file. Here we see the file os.txt contains the text "IOSXE L3SW image".

root@Nornir:~/Nornir/Class4# cat ../images/L3/os.txt

IOSXE L3SW image

I have modified the content of the same file as the following

root@Nornir:~/Nornir/Class4# cat ../images/L3/os.txt

IOS enterprise image

I am going to transfer the file and see what happens. You can see the transfer failed with a message stating, the overwrite failed.

root@Nornir:~/Nornir/Class4# python filecopy.py

------------------------------

\*\*\* SCP Server Enabled on L3SW \*\*\*

H**ost 'L3SW': task 'netmiko\_file\_transfer' failed with traceback:**

**ValueError: File already exists and overwrite\_file is disabled**

To enable overwrite, you have to set the "overwrite" argument as true in the task.

root@Nornir:~/Nornir/Class4# cat filecopy.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_file\_transfer,netmiko\_send\_config

from nornir.core.filter import F

def scp\_enabl(task):

# enable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="ip scp server enable"

)

print("-" \* 30)

print("\*\*\* SCP Server Enabled on", (task.host), "\*\*\*")

def file\_copy(task):

# Obtain the groups

group\_name = task.host.groups[0]

# Set the file based on groups

base\_file = "os.txt"

src\_file = "/root/Nornir/images/"f"{group\_name}/{base\_file}"

dst\_file = base\_file

# Transfer the file

results = task.run(

netmiko\_file\_transfer,

source\_file = src\_file,

dest\_file = dst\_file,

direction = "put",

overwrite\_file = True

)

# Print the status

print("-" \* 30)

print("On",task.host)

if results[0].changed is True:

print("File Transfer OK")

if results[0].changed is False:

print("File Exists already")

print("-" \* 30)

def scp\_remov(task):

# disable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="no ip scp server enable"

)

print("\*\*\* SCP Server disabled on", (task.host), "\*\*\*")

print("-" \* 30)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(name="L3SW")

print()

results = nr.run(task=scp\_enabl)

results = nr.run(task=file\_copy)

results = nr.run(task=scp\_remov)

print ()

I will run the code again and will see if the file transfer is working.

root@Nornir:~/Nornir/Class4# python filecopy.py

------------------------------

\*\*\* SCP Server Enabled on L3SW \*\*\*

------------------------------

On L3SW

File Transfer OK

------------------------------

\*\*\* SCP Server disabled on L3SW \*\*\*

------------------------------

* 1. Now lets get the file from the Network device to the SCP server. In both transfer and receive case you need to have scp server enabled on the Network device.

Also if the filename and content of the file is same then you need to have "overwrite\_file" argument set to true else the file cannot be received and you will see the error

"ValueError: File already exists and overwrite\_file is disabled"

root@Nornir:~/Nornir/Class4# cat fileget.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_file\_transfer,netmiko\_send\_config

from nornir.core.filter import F

def scp\_enabl(task):

# enable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="ip scp server enable"

)

print("-" \* 30)

print("\*\*\* SCP Server Enabled on", (task.host), "\*\*\*")

def get\_file(task):

# Obtain the groups

group\_name = task.host.groups[0]

# Set the file based on groups

src\_file = "os.txt"

dst\_file = "/root/Nornir/images/"f"{group\_name}/os.txt"

# Transfer the file

results = task.run(

netmiko\_file\_transfer,

source\_file = src\_file,

dest\_file = dst\_file,

direction = "get",

overwrite\_file = True

)

# Print the status

print("-" \* 30)

print("For",task.host)

if results[0].changed is True:

print("File Received from",task.host)

if results[0].changed is False:

print("File Exists already on SFTP Server")

print("-" \* 30)

def scp\_remov(task):

# disable scp server on remote device

results = task.run(netmiko\_send\_config,

config\_commands="no ip scp server enable"

)

print("\*\*\* SCP Server disabled on", (task.host), "\*\*\*")

print("-" \* 30)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(name="L3SW")

print()

results = nr.run(task=scp\_enabl)

results = nr.run(task=get\_file)

results = nr.run(task=scp\_remov)

print ()

root@Nornir:~/Nornir/Class4# python fileget.py

------------------------------

\*\*\* SCP Server Enabled on L3SW \*\*\*

------------------------------

For L3SW

File Received from L3SW

------------------------------

\*\*\* SCP Server disabled on L3SW \*\*\*

------------------------------

If I rerun the script again, it will receive a message the file exists already.

root@Nornir:~/Nornir/Class4# python fileget.py

------------------------------

\*\*\* SCP Server Enabled on L3SW \*\*\*

------------------------------

For L3SW

File Exists already on SFTP Server

------------------------------

\*\*\* SCP Server disabled on L3SW \*\*\*

------------------------------

* 1. Now let us see how to use netmiko configuration plugin to do configuration changes from a file. From the above example, you know how to do this via config\_commands but in this we do the configuration from the config file.

First we create base file for all devices based on the device group. You can file name based on the hostname of the "hostname-Lo99". The content of the base file is to create Lo99 on the devices.

root@Nornir:~/Nornir/newcfgs# pwd

/root/Nornir/newcfgs

root@Nornir:~/Nornir/newcfgs# ls

DC EDGE L2 L3 VOIP

root@Nornir:~/Nornir/newcfgs# cat DC/DCSW1-Lo99.txt

int loopback 99

ip address 10.99.99.6 255.255.255.255

root@Nornir:~/Nornir/newcfgs# cat L2/L2SWA-Lo99.txt

int loopback 99

ip address 10.99.99.3 255.255.255.255

root@Nornir:~/Nornir/newcfgs# cat L3/L3SWA-Lo99.txt

int loopback99

ip address 10.99.99.4 255.255.255.255

root@Nornir:~/Nornir/newcfgs# cat VOIP/CUBE-Lo99.txt

int loopback99

ip address 10.99.99.7 255.255.255.255

root@Nornir:~/Nornir/newcfgs# cat EDGE/RTRA-Lo99.txt

int loopback99

ip address 10.99.99.5 255.255.255.255

root@Nornir:~/Nornir/Class4# cat netmiko\_cfg\_file.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_config

from nornir.core.filter import F

def custom\_cfg(task):

groupname = task.host.groups[0]

hostname = task.host

base\_file = f"{hostname}""-Lo99.txt"

cfg\_file = "/root/Nornir/newcfgs/"f"{groupname}/{base\_file}"

print()

print("configuring",hostname)

print("-"\*30)

results = task.run(netmiko\_send\_config,

config\_file=cfg\_file,

)

print("#"\*50)

print()

print(results[0].result)

print()

if results[0].result is False:

print("\*"\*25,"config NOT changed on",hostname,"\*"\*25)

print()

if results[0].changed is True:

print("\*"\*25,"config changed on",hostname,"\*"\*25)

print()

print()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

print("#"\*50)

nr = nr.run(task=custom\_cfg)

print ()

print("#"\*50)

print()

I will run the above code to see if that works as expected. This code should create lo99 on the network devices.

root@Nornir:~/Nornir/Class4# python netmiko\_cfg\_file.py

##################################################

configuring L2SWA

------------------------------

configuring RTRA

------------------------------

configuring CUBE

------------------------------

configuring DCSW1

------------------------------

configuring L3SWA

------------------------------

##################################################

config term

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

DCSW1(config)# int loopback 99

DCSW1(config-if)# ip address 10.99.99.6 255.255.255.255

DCSW1(config-if)# end

DCSW1#

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* config changed on DCSW1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##################################################

config term

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

CUBE(config)#int loopback99

CUBE(config-if)#ip address 10.99.99.7 255.255.255.255

CUBE(config-if)#end

CUBE#

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* config changed on CUBE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##################################################

config term

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

RTRA(config)#int loopback99

RTRA(config-if)#ip address 10.99.99.5 255.255.255.255

RTRA(config-if)#end

RTRA#

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* config changed on RTRA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##################################################

config term

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

L3SWA(config)#int loopback99

L3SWA(config-if)#ip address 10.99.99.4 255.255.255.255

L3SWA(config-if)#end

L3SWA#

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* config changed on L3SWA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##################################################

config term

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

L2SWA(config)#int loopback 99

L2SWA(config-if)#ip address 10.99.99.3 255.255.255.255

L2SWA(config-if)#end

L2SWA#

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* config changed on L2SWA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##################################################

To verify if the configuration (creating the loopback 99 interface) has been done on the network devices, I will use netmiko-show commands

root@Nornir:~/Nornir# netmiko-show --cmd "sh run int lo99" ios

CUBE.txt:Building configuration...

CUBE.txt:Current configuration : 67 bytes

CUBE.txt:!

CUBE.txt:interface Loopback99

CUBE.txt: ip address 10.99.99.7 255.255.255.255

CUBE.txt:end

L2SWA.txt:Building configuration...

L2SWA.txt:Current configuration : 67 bytes

L2SWA.txt:!

L2SWA.txt:interface Loopback99

L2SWA.txt: ip address 10.99.99.3 255.255.255.255

L2SWA.txt:end

L3SWA.txt:Building configuration...

L3SWA.txt:Current configuration : 67 bytes

L3SWA.txt:!

L3SWA.txt:interface Loopback99

L3SWA.txt: ip address 10.99.99.4 255.255.255.255

L3SWA.txt:end

RTRA.txt:Building configuration...

RTRA.txt:Current configuration : 67 bytes

RTRA.txt:!

RTRA.txt:interface Loopback99

RTRA.txt: ip address 10.99.99.5 255.255.255.255

RTRA.txt:end

* 1. Now let us see how to use napalm configuration plugin to do configuration changes. First let us use the commands directly on the script and later let us see how to pull the commands from a file. First thing to note with napalm is napalm uses scp but netmiko uses SSH. If you use napalm only then you need to have "ip scp server enable" on all your devices. To make it easy for us I am going to use netmiko to create the SSH channel to enable SCP and use napalm to configure the network device and then disable scp. There is other way to use napalm only without netmiko which is called inline\_transfer which is shown in the point 9 below.

root@Nornir:~/Nornir/Class4# cat napalm\_cfg\_cmd.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_config,napalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

commands = "ip scp server enable"

scp\_eble = task.run(task=netmiko\_send\_config,

config\_commands=commands

)

print\_result(scp\_eble)

print("#" \* 50)

commands = "logging buffered 333555"

cfg = task.run(task=napalm\_configure,

configuration=commands

)

print\_result(cfg)

if cfg[0].changed is True:

print("\*\*\* config changed on",hostname,"\*\*\*")

if cfg[0].changed is False:

print("\*\*\* config NOT changed on",hostname,"\*\*\*")

print("?" \* 30)

print("#" \* 50)

commands = "no ip scp server enable"

scp\_dble = task.run(task=napalm\_configure,

configuration=commands

)

print\_result(scp\_dble)

print("#" \* 50)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

Now, I am going to run the script and see what happens.

root@Nornir:~/Nornir/Class4# python napalm\_cfg\_cmd.py

------------------------------

\*\*\* configuring L2SWA \*\*\*

------------------------------

\*\*\* configuring CUBE \*\*\*

------------------------------

\*\*\* configuring L3SWA \*\*\*

------------------------------

\*\*\* configuring RTRA \*\*\*

------------------------------

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

CUBE(config)#ip scp server enable

CUBE(config)#end

CUBE#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

RTRA(config)#ip scp server enable

RTRA(config)#end

RTRA#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

L3SWA(config)#ip scp server enable

L3SWA(config)#end

L3SWA#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

L2SWA(config)#ip scp server enable

L2SWA(config)#end

L2SWA#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 333555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on CUBE \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 333555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on RTRA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 333555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L2SWA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 333555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L3SWA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

Let us verify if the logging has been changed using the netmiko-show.

root@Nornir:~/Nornir# netmiko-show --cmd "sh run | inc buffer" ios

CUBE.txt:logging buffered 333555

L2SWA.txt:logging buffered 333555

L3SWA.txt:logging buffered 333555

RTRA.txt:logging buffered 333555

* 1. In this example, let us use configure interface lo99 on the ios devices using a file. This means the configuration is done in a file and placed at the respective groups/hosts. The filename is "hostname-Lo99.txt" and placed in the path respective to the group.

root@Nornir:~/Nornir# tree newcfgs/

newcfgs/

|-- DC

| `-- DCSW1-Lo99.txt

|-- EDGE

| `-- RTRA-Lo99.txt

|-- L2

| `-- L2SWA-Lo99.txt

|-- L3

| `-- L3SWA-Lo99.txt

|-- VOIP

| `-- CUBE-Lo99.txt

Here is the python script to create loopback99 on 4 ios devices.

root@Nornir:~/Nornir/Class4# cat napalm\_cfg\_file.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_config,napalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

commands = "ip scp server enable"

scp\_eble = task.run(task=netmiko\_send\_config,

config\_commands=commands

)

print\_result(scp\_eble)

print("#" \* 50)

base\_file = f"{hostname}""-Lo99.txt"

cfg\_file = "/root/Nornir/newcfgs/"f"{group\_name}/{base\_file}"

cfg = task.run(task=napalm\_configure,

filename=cfg\_file

)

print\_result(cfg)

if cfg[0].changed is True:

print("\*\*\* config changed on",hostname,"\*\*\*")

if cfg[0].changed is False:

print("\*\*\* config NOT changed on",hostname,"\*\*\*")

print("?" \* 30)

print("#" \* 50)

commands = "no ip scp server enable"

scp\_dble = task.run(task=napalm\_configure,

configuration=commands

)

print\_result(scp\_dble)

print("#" \* 50)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

I will run this script now and let us look at the output

root@Nornir:~/Nornir/Class4# python napalm\_cfg\_file.py

------------------------------

\*\*\* configuring L2SWA \*\*\*

------------------------------

\*\*\* configuring RTRA \*\*\*

------------------------------

\*\*\* configuring L3SWA \*\*\*

------------------------------

\*\*\* configuring CUBE \*\*\*

------------------------------

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

RTRA(config)#ip scp server enable

RTRA(config)#end

RTRA#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

CUBE(config)#ip scp server enable

CUBE(config)#end

CUBE#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

L3SWA(config)#ip scp server enable

L3SWA(config)#end

L3SWA#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv netmiko\_send\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

config term

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

L2SWA(config)#ip scp server enable

L2SWA(config)#end

L2SWA#

^^^^ END netmiko\_send\_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+int loopback99

+ip address 10.99.99.7 255.255.255.255

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on CUBE \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+int loopback99

+ip address 10.99.99.5 255.255.255.255

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on RTRA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+int loopback99

+ip address 10.99.99.4 255.255.255.255

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L3SWA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+int loopback 99

+ip address 10.99.99.3 255.255.255.255

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L2SWA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no ip scp server enable

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

root@Nornir:~/Nornir# netmiko-show --cmd "sh run int lo99" ios

CUBE.txt:Building configuration...

CUBE.txt:Current configuration : 67 bytes

CUBE.txt:!

CUBE.txt:interface Loopback99

CUBE.txt: ip address 10.99.99.7 255.255.255.255

CUBE.txt:end

L2SWA.txt:Building configuration...

L2SWA.txt:Current configuration : 67 bytes

L2SWA.txt:!

L2SWA.txt:interface Loopback99

L2SWA.txt: ip address 10.99.99.3 255.255.255.255

L2SWA.txt:end

L3SWA.txt:Building configuration...

L3SWA.txt:Current configuration : 67 bytes

L3SWA.txt:!

L3SWA.txt:interface Loopback99

L3SWA.txt: ip address 10.99.99.4 255.255.255.255

L3SWA.txt:end

RTRA.txt:Building configuration...

RTRA.txt:Current configuration : 67 bytes

RTRA.txt:!

RTRA.txt:interface Loopback99

RTRA.txt: ip address 10.99.99.5 255.255.255.255

RTRA.txt:end

* 1. If you want to test the configuration without committing the configuration to the device use the "dry\_run=True" argument. For example, I am setting the logging buffer to 444555 in this configuration but making it to run with dry\_run set to true.

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_config,napalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

commands = "ip scp server enable"

scp\_eble = task.run(task=netmiko\_send\_config,

config\_commands=commands

)

print\_result(scp\_eble)

print("#" \* 50)

commands = "logging buffered 444555"

cfg = task.run(task=napalm\_configure,

configuration=commands,

**dry\_run=True**

)

print\_result(cfg)

if cfg[0].changed is True:

print("\*\*\* config changed on",hostname,"\*\*\*")

if cfg[0].changed is False:

print("\*\*\* config NOT changed on",hostname,"\*\*\*")

print("?" \* 30)

print("#" \* 50)

commands = "no ip scp server enable"

scp\_dble = task.run(task=napalm\_configure,

configuration=commands

)

print\_result(scp\_dble)

print("#" \* 50)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(platform="ios")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

I will run the above script and I am showing only the respective output from the screen. You can it say the configuration is changed but though it says the configuration is changed, the reality with dry\_run is commit is not done.

root@Nornir:~/Nornir/Class4# python napalm\_cfg\_cmd.py

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 444555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on CUBE \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 444555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on RTRA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 444555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L2SWA \*\*\*

##################################################

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 444555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L3SWA \*\*\*

##################################################

we can verify if the configuration of the buffer is changed to 444555 on the devices under the ios platform.

You can see from the output below the configuration is not changed.

root@Nornir:~/Nornir# netmiko-grep buffer ios

CUBE.txt:logging buffered 333555

L2SWA.txt:logging buffered 333555

L3SWA.txt:logging buffered 333555

RTRA.txt:logging buffered 333555

* 1. If you do not wish to use netmiko and use only napalm, you need to enable the inline\_transfer to be true. This is done under the host level or groups or at the defaults level. I have set the inline\_transfer under defaults.yaml file

root@Nornir:~/Class4# cat ../defaults.yaml

---

username: cisco

password: 123!Cisco

connection\_options:

netmiko:

extras:

inline\_transfer: True

napalm:

extras:

optional\_args:

inline\_transfer: True

root@Nornir:~/Class4# cat napalm\_cfg\_cmd.py

from nornir import InitNornir

from nornir.plugins.tasks.networking importnapalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

commands = "logging buffered 100555"

cfg = task.run(task=napalm\_configure,

configuration=commands,

)

print\_result(cfg)

if cfg[0].changed is True:

print("\*\*\* config changed on",hostname,"\*\*\*")

if cfg[0].changed is False:

print("\*\*\* config NOT changed on",hostname,"\*\*\*")

print("?" \* 30)

print("#" \* 50)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(name="L2SWA")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

root@Nornir:~/Class4# python napalm\_cfg\_cmd.py

------------------------------

\*\*\* configuring L2SWA \*\*\*

------------------------------

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 300555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

\*\*\* config changed on L2SWA \*\*\*

##################################################

* 1. In this example, we are going to create a scenario where we get the running configuration from a Cisco Router, edit the Running configuration so that it can be merged back to the device if there is any changes that was unexpected. For simulating the unexpected change, we are removing a loopback interface from the device. For remediation, we will merge the certain part of the running config back to the device to make it back to its working state. In this example, we are merging the old configuration to the new configuration which turn will revert only the changes made and not the whole running configuration.

root@Nornir:~/Class4# cat napalm\_cfg\_merge.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get,napalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

get\_run = task.run(task=napalm\_get,

getters = ["config"],

retrieve = "running"

)

cfg\_run = get\_run.result["config"]["running"]

cfg\_run = cfg\_run.splitlines()

cfg\_run = cfg\_run[7:106]

cfg\_run = "\n".join(cfg\_run)

print("!" \* 50)

print("Simulating the Unexpected Change")

cfg = task.run(task=napalm\_configure,

configuration ="no int lo99"

)

print\_result(cfg)

print("!" \* 50)

print("Applying Remediation")

result = task.run(task=napalm\_configure,

configuration=cfg\_run,

replace=False

)

print\_result(result)

print("#" \* 50)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(name="RTRA")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

root@Nornir:~/Class4# python napalm\_cfg\_merge.py

------------------------------

\*\*\* configuring RTRA \*\*\*

------------------------------

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Simulating the Unexpected Change

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

-no int lo99

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Applying Remediation

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+interface Loopback99

+ ip address 10.99.99.5 255.255.255.255

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

##################################################

* 1. Here we are going to see the replace operation. When performing replace operation, you have to be aware of the caveats mentioned here for the napalm,

<https://napalm.readthedocs.io/en/latest/support/ios.html> . The napalm\_replace operation entirely replaces the running configuration of the system completely. Due to the napalm caveats related to the banner configuration on the router, I am using a different system here which does not have banner configuration.

(py3\_venv) [sethuvignesh@norn1a Class4]$ cat napalm\_cfg\_replace.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get,napalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

get\_run = task.run(task=napalm\_get,

getters = ["config"],

retrieve = "running"

)

cfg\_run = get\_run.result["config"]["running"]

cfg\_run = cfg\_run.splitlines()

cfg\_run = cfg\_run[7:]

cfg\_run = "\n".join(cfg\_run)

print("!" \* 50)

print("Simulating the Unexpected Change")

cfg = task.run(task=napalm\_configure,

configuration="logging buffered 333555"

)

print\_result(cfg)

print("!" \* 50)

print("Applying Remediation")

result = task.run(task=napalm\_configure,

configuration=cfg\_run,

replace=True

)

print\_result(result)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file="/home/sethuvignesh/config.yaml")

nr = nr.filter(name="cisco3")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

(py3\_venv) [sethuvignesh@norn1a Class4]$ python napalm\_cfg\_replace.py

------------------------------

\*\*\* configuring cisco3 \*\*\*

------------------------------

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Simulating the Unexpected Change

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 333555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Applying Remediation

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 444555

-logging buffered 333555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

* 1. In some cases you might need to replace or merge only few lines on the top and few lines on the bottom. To do that, here is the script. In this example, any lines from 7 till 41 and any lines between 50 till 166 and any lines between 178 till 198 will be sent back to the device.

(py3\_venv) [sethuvignesh@norn1a Class4]$ cat napalm\_cfg\_replace.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get,napalm\_configure

from nornir.core.filter import F

from nornir.plugins.functions.text import print\_result

from pprint import pprint

def custom\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* configuring",hostname,"\*\*\*")

print("-" \* 30)

get\_run = task.run(task=napalm\_get,

getters = ["config"],

retrieve = "running"

)

cfg\_run = get\_run.result["config"]["running"]

cfg\_run = cfg\_run.splitlines()

cfg\_run = cfg\_run[7:41] + cfg\_run[50:166] + cfg\_run[178:198]

cfg\_run = "\n".join(cfg\_run)

print("!" \* 50)

print("Simulating the Unexpected Change")

cfg = task.run(task=napalm\_configure,

configuration="logging buffered 444555"

)

print\_result(cfg)

print("!" \* 50)

print("Applying Remediation")

result = task.run(task=napalm\_configure,

configuration=cfg\_run,

replace=True

)

print\_result(result)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file="/home/sethuvignesh/config.yaml")

nr = nr.filter(name="cisco3")

print("-" \* 30)

nr = nr.run(task=custom\_cfg)

print()

(py3\_venv) [sethuvignesh@norn1a Class4]$ python napalm\_cfg\_replace.py

------------------------------

\*\*\* configuring cisco3 \*\*\*

------------------------------

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Simulating the Unexpected Change

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 444555

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Applying Remediation

vvvv napalm\_configure \*\* changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

+logging buffered 30000

-logging buffered 444555

-banner login ^CC

testing new banner

this is a very long banner

multi line banner

seriously long banner

add on even more banner

add more words so longer

adding even more words

last line le im so bored

^C

^^^^ END napalm\_configure ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Class 5 - Device Config Generation with Jinja2 Templating

lundi, 7 octobre 2019

18:48

1. In this first example, I am going to generate the device config for a L2SW which has access ports. Here is my groups file which has needed informations as variables.

root@Nornir:~/Class5# cat ../groups.yaml

---

EDGE:

platform: ios

VOIP:

platform: ios

DC:

platform: nxos

L3:

platform: ios

data:

ns1: 8.8.8.8

ns2: 8.8.4.4

ntp1: 81.94.123.16

interfaces:

GigabitEthernet3/0:

mode: trunk

state: no shut

descr: Free

GigabitEthernet3/1:

mode: trunk

state: no shut

descr: Free

GigabitEthernet3/2:

mode: trunk

state: no shut

descr: Free

GigabitEthernet3/3:

mode: trunk

state: no shut

descr: Free

L2:

platform: ios

data:

ns1: 8.8.8.8

ns2: 8.8.4.4

ntp1: 81.94.123.16

interfaces:

GigabitEthernet3/0:

mode: access

state: no shut

descr: Free

vlan: 1

GigabitEthernet3/1:

mode: access

state: no shut

descr: Free

vlan: 1

GigabitEthernet3/2:

mode: access

state: no shut

descr: Free

vlan: 1

GigabitEthernet3/3:

mode: access

state: no shut

descr: Free

vlan: 1

I have created the templates under each groups. For example under the group L2 and L3.

root@Nornir:~/templates# tree

.

|-- DC

|-- EDGE

|-- L2

| |-- jjtemp.j2

|

|-- L3

| |-- jjtemp.j2

|

|-- VOIP

Here is my template which is basic but I am planning to improve the template as like the one I did for Ansible.

root@Nornir:~/templates# cat L2/jjtemp.j2

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

!

hostname {{ host.hostname }}

!

ip name-server {{ ns1 }} {{ ns2 }}

!

no aaa new-model

clock timezone EST

!

ntp server {{ ntp1 }}

!

{% for intf, data in interfaces.items() %}

interface {{ intf }}

description {{ data.descr }}

switchport access vlan {{ data.vlan }}

switchport mode access

{{ data.state }}

{% endfor %}

Here is my script that specifies where the template can be found and what task has to be run. The \*\*hostname is specify from which host/group the variables has to be referenced.

root@Nornir:~/Class5# cat cfg\_temp.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_configure

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "jjtemp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

print(prep\_result)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="L2"))

print("-" \* 30)

prep\_result = nr.run(task=prep\_cfg)

Here I run the script and you can see a template is generated for the L2 group, meaning for the switch L2SWA.

root@Nornir:~/Class5# python cfg\_temp.py

------------------------------

\*\*\* Render template for L2SWA \*\*\*

------------------------------

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

!

hostname L2SWA

!

ip name-server 8.8.8.8 8.8.4.4

!

no aaa new-model

clock timezone EST

!

ntp server 81.94.123.16

!

interface GigabitEthernet3/0

description Free

switchport access vlan 1

switchport mode access

no shut

interface GigabitEthernet3/1

description Free

switchport access vlan 1

switchport mode access

no shut

interface GigabitEthernet3/2

description Free

switchport access vlan 1

switchport mode access

no shut

interface GigabitEthernet3/3

description Free

switchport access vlan 1

switchport mode access

no shut

1. In this example, I am going iterate on the above example. It means in the previous example, I have created the template and printed it out but in this example I will create the template but will store it in the respective folder. The template is the same but the script has additional task.

Here is my script.

root@Nornir:~/Class5# cat cfg\_temp\_store.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_configure

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.hostname

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "jjtemp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

def store\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.hostname

print("\*\*\* Store template for",hostname,"\*\*\*")

print("-" \* 30)

cfg\_path = "/root/Nornir/newcfgs/"f"{group\_name}/"

filename = f"{cfg\_path}/{hostname}\_basic"

cfg\_prepared = task.host['prep\_result']

result = task.run(task=files.write\_file, filename=filename, content=cfg\_prepared)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="L2"))

print("-" \* 30)

prep\_result = nr.run(task=prep\_cfg)

print()

store\_result = nr.run(task=store\_cfg)

Here I am running the script and I am not going to print only what the task does but not the result.

root@Nornir:~/Class5# python cfg\_temp\_store.py

------------------------------

\*\*\* Render template for L2SWA \*\*\*

------------------------------

\*\*\* Store template for L2SWA \*\*\*

------------------------------

You can see a new file is stored in the respective L2 group and you can see the content of that file.

root@Nornir:~/newcfgs# tree L2

L2

`-- L2SWA\_basic

root@Nornir:~/newcfgs# cat L2/L2SWA\_basic

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

!

hostname L2SWA

!

ip name-server 8.8.8.8 8.8.4.4

!

no aaa new-model

clock timezone EST

!

ntp server 81.94.123.16

!

interface GigabitEthernet3/0

description Free

switchport access vlan 1

switchport mode access

no shut

interface GigabitEthernet3/1

description Free

switchport access vlan 1

switchport mode access

no shut

interface GigabitEthernet3/2

description Free

switchport access vlan 1

switchport mode access

no shut

interface GigabitEthernet3/3

description Free

switchport access vlan 1

switchport mode access

no shut

!

In this example, I am going iterate more on the above example by pushing the prepared config to the device.

Here is my script that has template preparation, template storing, deploying the generated configs.

root@Nornir:~/Class5# cat cfg\_temp\_deploy.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_configure

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.hostname

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "jjtemp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

def store\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.hostname

print("\*\*\* Store template for",hostname,"\*\*\*")

print("-" \* 30)

cfg\_path = "/root/Nornir/newcfgs/"f"{group\_name}/"

filename = f"{cfg\_path}/{hostname}\_basic"

cfg\_prepared = task.host['prep\_result']

result = task.run(task=files.write\_file, filename=filename, content=cfg\_prepared)

def deploy\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.hostname

print("\*\*\* Deploy template for",hostname,"\*\*\*")

print("-" \* 30)

cfg\_path = "/root/Nornir/newcfgs/"f"{group\_name}/"

filename = f"{cfg\_path}/{hostname}\_basic"

with open(filename, 'r') as f:

cfg\_new = f.read()

result = task.run(task=napalm\_configure, configuration=cfg\_new)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="L2"))

print("-" \* 30)

prep\_result = nr.run(task=prep\_cfg)

print()

store\_result = nr.run(task=store\_cfg)

print()

deploy\_result = nr.run(task=deploy\_cfg)

Here I am going to run the script I will show what task does not the content of results.

root@Nornir:~/Class5# python cfg\_temp\_deploy.py

------------------------------

\*\*\* Render template for L2SWA \*\*\*

------------------------------

\*\*\* Store template for L2SWA \*\*\*

------------------------------

\*\*\* Deploy template for L2SWA \*\*\*

------------------------------

Here you can see the template has been pushed successfully.

root@Nornir:~/Class5# netmiko-show --cmd "sh run | b 3/0" L2SWA

interface GigabitEthernet3/0

description Free

switchport mode access

media-type rj45

negotiation auto

!

interface GigabitEthernet3/1

description Free

switchport mode access

media-type rj45

negotiation auto

!

interface GigabitEthernet3/2

description Free

switchport mode access

media-type rj45

negotiation auto

!

interface GigabitEthernet3/3

description Free

switchport mode access

media-type rj45

negotiation auto

root@Nornir:~/Class5# netmiko-grep ntp L2SWA

ntp server 81.94.123.16

root@Nornir:~/Class5# netmiko-grep name-server L2SWA

ip name-server 8.8.8.8

ip name-server 8.8.4.4

1. In this example, I am going to use macros and loops to generate the interfaces config and include that macros template into the generic template. I am not going to stote or push this config but just to create and print it out to the terminal.

Here is my templates for macros and generic.

root@Nornir:~/Class5# cat ../templates/L2/jjtemp.j2

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

!

hostname {{ host.hostname }}

!

ip name-server {{ ns1 }} {{ ns2 }}

!

no aaa new-model

clock timezone EST

!

ntp server {{ ntp1 }}

!

{% include "macro\_loops.j2" %}

!

ip default-gateway {{ dfg }}

Here is my macros and loop template.

root@Nornir:~/Class5# cat ../templates/L2/macro\_loops.j2

{% macro intf\_trk(native\_vlan=1, allowed\_vlan=1) -%}

switchport mode trunk

switchport trunk native vlan {{ native\_vlan }}

switchport trunk allowed vlan {{ allowed\_vlan }}

{%- endmacro %}

!

{% macro intf\_acc(vlan=1) -%}

switchport mode access

switchport access vlan {{ vlan }}

{%- endmacro %}

!

{% for intf, data in interfaces.items() %}

!

interface {{ intf }}

{% if data.mode == "trunk"%}

description {{ data.descr }}

{{ intf\_trk(native\_vlan=1, allowed\_vlan= "1,200") }}

{{ data.state }}

{% elif data.mode == "access"%}

description {{ data.descr }}

{{ intf\_acc(vlan=1) }}

{{ data.state }}

{% endif %}

{% endfor %}

Here is my script

root@Nornir:~/Class5# cat cfg\_temp\_render.py

from nornir import InitNornir

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.hostname

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "jjtemp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

print(prep\_result)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/nornir.yaml")

nr = nr.filter(F(groups\_\_contains="L2"))

print("-" \* 30)

prep\_result = nr.run(task=prep\_cfg)

print()

Here I will run the script and see if that works as expected.

root@Nornir:~/Class5# python cfg\_temp\_render.py

------------------------------

\*\*\* Render template for L2SWA \*\*\*

------------------------------

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

hostname L2SWA

!

ip name-server 8.8.8.8 8.8.4.4

!

no aaa new-model

clock timezone EST

!

ntp server 81.94.123.16

!

!

interface GigabitEthernet3/0

description Free

switchport mode access

switchport access vlan 1

no shut

!

interface GigabitEthernet3/1

description Free

switchport mode access

switchport access vlan 1

no shut

!

interface GigabitEthernet3/2

description Free

switchport mode access

switchport access vlan 1

no shut

!

interface GigabitEthernet3/3

description Free

switchport mode trunk

switchport trunk native vlan 1

switchport trunk allowed vlan 1,200

no shut

!

ip default-gateway 192.168.122.1

Class 6 - Failed Tasks and Exception, Troubleshooting, Storing Passwords, Session logging

samedi, 19 octobre 2019

07:14

* 1. When troubleshooting set the num\_workers to 1.
  2. When running the script simultaneously on all device (ie) num\_workers!=1, the outputs being sent back by the devices to the standard out (terminal screen) will be out of order. To avoid this, use all your print statements in the main() function.
  3. The defaults options of the Nornir session logging to the console is false but Nornir has a bug so the session log is still on the console
  4. You can also have the session outputs from the network devices cane stored in a file. The session outputs from the network devices and Nornir session logs are different.
  5. When you refer a group as "task.host.groups[0]", the output is a string, but if you refer it as "task.host.groups.refs[0]" then it is a group object

Class 7 - Bonus Lessons - Tasks using Ansible Inventory

jeudi, 10 octobre 2019

19:11

1. In this example, I am going to use ansible inventory instead of simple inventory

root@Nornir:~# cat config.yaml

---

inventory:

plugin: nornir.plugins.inventory.ansible.AnsibleInventory

options:

hostsfile: "/root/Nornir/hosts.yaml"

core:

num\_workers: 25

logging:

enabled: False

In the below file, you can see it is a yaml file but "---" is not used because it is not supported. This should be similar to hosts file in ansible (ini type file).

root@Nornir:~# cat hosts.yaml

[all:vars]

ansible\_connection=local

host\_key\_checking=false

[EDGE]

RTRA ansible\_host=192.168.122.245

RTRB ansible\_host=192.168.122.237

[L2]

L2SWA ansible\_host=192.168.122.243

L2SWB ansible\_host=192.168.122.139

[L3]

L3SWA ansible\_host=192.168.122.244

[DC]

DCSW1 ansible\_host=192.168.122.246

[VOIP]

CUBE ansible\_host=192.168.122.247

H323 ansible\_host=192.168.122.138

You can also use yaml type hosts file.

root@Nornir:~# cat hosts.yaml

---

all:

 children:

   L2:

     hosts:

       L2SWA:

         ansible\_host: 192.168.122.243

root@Nornir:~# tree ./group\_vars/

./group\_vars/

|-- L2.yaml

`-- all.yaml

To check if inventory is correctly working, try

root@Nornir:~# ansible-inventory -i ./hosts.yaml --graph

@all:

|--@DC:

| |--DCSW1

|--@EDGE:

| |--RTRA

| |--RTRB

|--@L2:

| |--L2SWA

| |--L2SWB

|--@L3:

| |--L3SWA

|--@VOIP:

| |--CUBE

| |--H323

|--@ungrouped:

root@Nornir:~# cat group\_vars/L2.yaml

---

platform: ios

connection\_options:

netmiko:

platform: cisco\_ios

extras: {}

napalm:

platform: ios

extras:

optional\_args: {}

Note that **group\_vars/L2/L2.yaml** that won't work with the Nornir Ansible plugin. You have to use group\_vars/L2.yaml (i.e. Nornir-Ansible Inventory) does NOT support a directory of YAML files.

root@Nornir:~# cat group\_vars/all.yaml

---

username: cisco

password: 123!Cisco

Here is my simple script to see if inventory is invoked.

root@Nornir:~/Class7# cat inven\_test1.py

from nornir import InitNornir

def main():

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

L2 = nr.inventory.hosts

print(L2)

if \_\_name\_\_ == "\_\_main\_\_":

main()

root@Nornir:~/Class7# python inven\_test1.py

{'rtra': Host: rtra, 'rtrb': Host: rtrb, 'l2swa': Host: l2swa, 'l2swb': Host: l2swb, 'l3swa': Host: l3swa, 'dcsw1': Host: dcsw1, 'cube': Host: cube, 'h323': Host: h323}

You can the output are in lower case, this means when you filter the hosts use lower case letters when filtering the hosts.

1. In this example, I will iterate to do some task executed on the ansible inventory. In this basic example, I am going to run a script to print the hostname of a device.

root@Nornir:~/Class7# cat netmiko\_send\_cmd.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.plugins.functions.text import print\_result

def custom(task):

results = task.run(task = netmiko\_send\_command, command\_string = "show run | inc hostname")

print\_result(results)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file = "/root/Nornir/config.yaml")

nr = nr.filter(name="l2swa")

nr = nr.run(task=custom)

print()

root@Nornir:~/Class7# python netmiko\_send\_cmd.py

netmiko\_send\_command\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* l2swa \*\* changed : False \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vvvv netmiko\_send\_command \*\* changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

hostname L2SWA

^^^^ END netmiko\_send\_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

1. I am going to do some templating for bgp configuration but will use the ansible type inventory and group\_vars and host\_vars.

Here is my hosts inventory, group\_vars, host\_vars.

root@Nornir:~/Class7# cat ../hosts.yaml

[all:vars]

ansible\_connection=local

host\_key\_checking=false

[EDGE]

RTRA ansible\_host=192.168.122.245

RTRB ansible\_host=192.168.122.237

[L2]

L2SWA ansible\_host=192.168.122.243

L2SWB ansible\_host=192.168.122.139

[L3]

L3SWA ansible\_host=192.168.122.244

[DC]

DCSW1 ansible\_host=192.168.122.246

[VOIP]

CUBE ansible\_host=192.168.122.247

H323 ansible\_host=192.168.122.138

root@Nornir:~/Class7# cat ../group\_vars/EDGE.yaml

---

platform: ios

connection\_options:

netmiko:

platform: cisco\_ios

extras: {}

napalm:

platform: ios

extras:

optional\_args: {}

bgp\_peers:

- {ip: 172.16.122.100, as: 100}

- {ip: 172.16.122.200, as: 200}

root@Nornir:~/Class7# cat ../host\_vars/rtra.yaml

---

bgp\_local\_ip: 172.16.122.100

bgp\_local\_as: 100

router\_id: 10.100.100.245

connected\_local1: 10.100.100.0

connected\_local2: 172.16.122.0

net\_mask: 255.255.255.0

dfg: 192.168.122.1

root@Nornir:~/Class7# cat ../host\_vars/rtrb.yaml

---

bgp\_local\_ip: 172.16.122.200

bgp\_local\_as: 200

router\_id: 10.200.200.237

connected\_local1: 10.200.200.0

connected\_local2: 172.16.122.0

net\_mask: 255.255.255.0

dfg: 192.168.122.1

Here is my BGP jinja2 template.

root@Nornir:~/Class7# cat ../templates/EDGE/bgp.j2

router bgp {{ bgp\_local\_as }}

bgp router-id {{ router\_id }}

{%- for peer in bgp\_peers %}

{%- if peer.ip != bgp\_local\_ip %}

neighbor {{ peer.ip }} remote-as {{ peer.as }}

neighbor {{ peer.ip }} maximum-prefix 100

{%- endif %}

{%- endfor %}

network {{ connected\_local1 }} mask {{ net\_mask }}

network {{ connected\_local2 }} mask {{ net\_mask }}

exit

!

ip route {{ connected\_local1 }} {{ net\_mask }} null0

ip route {{ connected\_local2 }} {{ net\_mask }} null0

Here is my python script

root@Nornir:~/Class7# cat bgp\_cfg\_render.py

from nornir import InitNornir

from nornir.core.filter import F

from nornir.plugins.tasks import text

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "bgp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

print(prep\_result)

print()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(F(groups\_\_contains="EDGE"))

print()

prep\_result = nr.run(task=prep\_cfg,num\_workers=1)

root@Nornir:~/Class7# python bgp\_cfg\_render.py

\*\*\* Render template for rtra \*\*\*

------------------------------

router bgp 100

bgp router-id 10.100.100.245

neighbor 172.16.122.200 remote-as 200

neighbor 172.16.122.200 maximum-prefix 100

network 10.100.100.0 mask 255.255.255.0

network 172.16.122.0 mask 255.255.255.0

exit

!

ip route 10.100.100.0 255.255.255.0 null0

ip route 172.16.122.0 255.255.255.0 null0

\*\*\* Render template for rtrb \*\*\*

------------------------------

router bgp 200

bgp router-id 10.200.200.237

neighbor 172.16.122.100 remote-as 100

neighbor 172.16.122.100 maximum-prefix 100

network 10.200.200.0 mask 255.255.255.0

network 172.16.122.0 mask 255.255.255.0

exit

!

ip route 10.200.200.0 255.255.255.0 null0

ip route 172.16.122.0 255.255.255.0 null0

1. In this example, I am going to prepare a router config for RTRA and RTRB and store it. This config includes both basic and bgp config in a single file.

root@Nornir:~/templates/EDGE# cat basic\_bgp.j2

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

hostname {{ host.name }}

!

ip name-server {{ ns1 }} {{ ns2 }}

!

no aaa new-model

clock timezone EST

!

{% include "bgp.j2" %}

!

ip route 0.0.0.0 0.0.0.0 {{ dfg }}

!

ntp server {{ ntp1 }}

Here is my script that renders the config for rtra and rtrb and stores it.

root@Nornir:~/Class7# cat bgp\_cfg\_store.py

from nornir import InitNornir

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "basic\_bgp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

def store\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* Store template for",hostname,"\*\*\*")

print("-" \* 30)

cfg\_path = "/root/Nornir/newcfgs/"f"{group\_name}/"

filename = f"{cfg\_path}/{hostname}-bgp"

cfg\_prepared = task.host['prep\_result']

result = task.run(task=files.write\_file, filename=filename, content=cfg\_prepared)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(F(groups\_\_contains="EDGE"))

print("-" \* 30)

prep\_result = nr.run(task=prep\_cfg)

print()

store\_result = nr.run(task=store\_cfg)

print()

I am running the script and show the results of the script.

root@Nornir:~/Class7# python bgp\_cfg\_\_store.py

------------------------------

\*\*\* Render template for rtra \*\*\*

------------------------------

\*\*\* Render template for rtrb \*\*\*

------------------------------

\*\*\* Store template for rtra \*\*\*

------------------------------

\*\*\* Store template for rtrb \*\*\*

------------------------------

Here is the content of the file store in the newcfgs folder.

root@Nornir:~/Class7# ls ../newcfgs/EDGE/ | grep bgp

rtra-bgp

rtrb-bgp

root@Nornir:~/Class7# cat ../newcfgs/EDGE/rtra-bgp

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

hostname rtra

!

ip name-server 8.8.8.8 8.8.4.4

!

no aaa new-model

clock timezone EST

!

router bgp 100

bgp router-id 10.100.100.245

neighbor 172.16.122.200 remote-as 200

neighbor 172.16.122.200 maximum-prefix 100

network 10.100.100.0 mask 255.255.255.0

network 172.16.122.0 mask 255.255.255.0

exit

!

ip route 10.100.100.0 255.255.255.0 null0

ip route 172.16.122.0 255.255.255.0 null0

!

ip route 0.0.0.0 0.0.0.0 192.168.122.1

!

ntp server 82.197.188.130

root@Nornir:~/Class7# cat ../newcfgs/EDGE/rtrb-bgp

service timestamps debug datetime msec localtime

service timestamps log datetime msec

service sequence-numbers

!

hostname rtrb

!

ip name-server 8.8.8.8 8.8.4.4

!

no aaa new-model

clock timezone EST

!

router bgp 200

bgp router-id 10.200.200.237

neighbor 172.16.122.100 remote-as 100

neighbor 172.16.122.100 maximum-prefix 100

network 10.200.200.0 mask 255.255.255.0

network 172.16.122.0 mask 255.255.255.0

exit

!

ip route 10.200.200.0 255.255.255.0 null0

ip route 172.16.122.0 255.255.255.0 null0

!

ip route 0.0.0.0 0.0.0.0 192.168.122.1

!

ntp server 82.197.188.130

1. In this example, I am just going to verify if the BGP peer adjacency formed and BGP routes received from the peers are valid. If this is valid we print a BGP peers OK msg. If the BGP adjacency or the routes received is not valid, we raise a error.

Here is my script

root@Nornir:~/Class7# cat bgp\_adj\_verify.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_configure,netmiko\_send\_command

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def verify\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* verify BGP adjancecy on",hostname,"\*\*\*")

result = task.run(task=netmiko\_send\_command, command\_string=" show ip bgp summary | inc 172.16.122")

bgp\_neighbor = result[0].result

bgp\_peers = bgp\_neighbor.count("172.16.122")

print("\*\*\* verify BGP routes on",hostname,"\*\*\*")

result = task.run(task=netmiko\_send\_command, command\_string= "sh ip route bgp")

received\_routes = result[0].result

bgp\_routes = received\_routes.count("10.")

if bgp\_peers < 1:

raise ValueError(f"Check BGP Config: only { bgp\_peers } adjacency formed")

elif bgp\_routes < 2:

raise ValueError(f"Check BGP advertised routes: only { bgp\_routes } routes received")

else:

msg = f"""

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On {task.host.name}: BGP peers OK

On {task.host.name}: BGP routes OK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

"""

print(msg)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(F(groups\_\_contains="EDGE"))

print("-" \* 30)

deploy\_result = nr.run(task=verify\_cfg)

root@Nornir:~/Class7# python bgp\_adj\_verify.py

------------------------------

\*\*\* verify BGP adjancecy on rtra \*\*\*

\*\*\* verify BGP adjancecy on rtrb \*\*\*

\*\*\* verify BGP routes on rtrb \*\*\*

\*\*\* verify BGP routes on rtra \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On rtrb: BGP peers OK

On rtrb: BGP routes OK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On rtra: BGP peers OK

On rtra: BGP routes OK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

For testing purpose, I changed the received routes value to 3 in the script. You can see below I receive the value error because normally in my topology I should receive only 2 routes.

root@Nornir:~/Class7# python bgp\_adj\_verify.py

------------------------------

\*\*\* verify BGP adjancecy on rtra \*\*\*

\*\*\* verify BGP adjancecy on rtrb \*\*\*

\*\*\* verify BGP routes on rtra \*\*\*

\*\*\* verify BGP routes on rtrb \*\*\*

Host 'rtrb': task 'verify\_cfg' failed with traceback:

Traceback (most recent call last):

File "/usr/local/lib/python3.6/dist-packages/nornir/core/task.py", line 85, in start

r = self.task(self, \*\*self.params)

File "bgp\_adj\_verify.py", line 26, in verify\_cfg

raise ValueError(f"Check BGP advertised routes: only { bgp\_routes } routes received")

ValueError: Check BGP advertised routes: only 2 routes received

Host 'rtra': task 'verify\_cfg' failed with traceback:

Traceback (most recent call last):

File "/usr/local/lib/python3.6/dist-packages/nornir/core/task.py", line 85, in start

r = self.task(self, \*\*self.params)

File "bgp\_adj\_verify.py", line 26, in verify\_cfg

raise ValueError(f"Check BGP advertised routes: only { bgp\_routes } routes received")

ValueError: Check BGP advertised routes: only 2 routes received

1. In the below example, I am going to have full script starting from rendering of BGP config, storing the config, deploying the config to the device, verify the BGP peers and received routes.

root@Nornir:~/Class7# cat bgp\_cfg\_full.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_configure,netmiko\_send\_command

from nornir.core.filter import F

from nornir.plugins.tasks import text,files

from nornir.plugins.functions.text import print\_result

def prep\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* Render template for",hostname,"\*\*\*")

print("-" \* 30)

template = "basic\_bgp.j2"

template\_path = "/root/Nornir/templates/"f"{group\_name}"

result = task.run(task=text.template\_file, path=template\_path, template=template, \*\*task.host)

prep\_result = result[0].result

task.host['prep\_result'] = prep\_result

def store\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* Store template for",hostname,"\*\*\*")

print("-" \* 30)

cfg\_path = "/root/Nornir/newcfgs/"f"{group\_name}/"

filename = f"{cfg\_path}/{hostname}-bgp"

cfg\_prepared = task.host['prep\_result']

result = task.run(task=files.write\_file, filename=filename, content=cfg\_prepared)

def deploy\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* Deploy template for",hostname,"\*\*\*")

print("-" \* 30)

cfg\_path = "/root/Nornir/newcfgs/"f"{group\_name}/"

filename = f"{cfg\_path}/{hostname}-bgp"

with open(filename, 'r') as f:

cfg\_new = f.read()

result = task.run(task=napalm\_configure, configuration=cfg\_new)

def verify\_cfg(task):

group\_name = task.host.groups[0]

hostname = task.host.name

print("\*\*\* verify BGP adjancecy on",hostname,"\*\*\*")

result = task.run(task=netmiko\_send\_command, command\_string=" show ip bgp summary | inc 172.16.122")

bgp\_neighbor = result[0].result

bgp\_peers = bgp\_neighbor.count("172.16.122")

print("\*\*\* verify BGP routes on",hostname,"\*\*\*")

result = task.run(task=netmiko\_send\_command, command\_string= "sh ip route bgp")

received\_routes = result[0].result

bgp\_routes = received\_routes.count("10.")

if bgp\_peers < 1:

raise ValueError(f"Check BGP Config: only { bgp\_peers } adjacency formed")

elif bgp\_routes < 2:

raise ValueError(f"Check BGP advertised routes: only { bgp\_routes } routes received")

else:

msg = f"""

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On {task.host.name}: BGP peers OK

On {task.host.name}: BGP routes OK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

"""

print(msg)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(F(groups\_\_contains="EDGE"))

print("-" \* 30)

prep\_result = nr.run(task=prep\_cfg)

print()

store\_result = nr.run(task=store\_cfg)

print()

deploy\_result = nr.run(task=deploy\_cfg)

print()

deploy\_result = nr.run(task=verify\_cfg)

root@Nornir:~/Class7# python bgp\_cfg\_full.py

------------------------------

\*\*\* Render template for rtra \*\*\*

------------------------------

\*\*\* Render template for rtrb \*\*\*

------------------------------

\*\*\* Store template for rtra \*\*\*

------------------------------

\*\*\* Store template for rtrb \*\*\*

------------------------------

\*\*\* Deploy template for rtra \*\*\*

------------------------------

\*\*\* Deploy template for rtrb \*\*\*

------------------------------

\*\*\* verify BGP adjancecy on rtra \*\*\*

\*\*\* verify BGP adjancecy on rtrb \*\*\*

\*\*\* verify BGP routes on rtrb \*\*\*

\*\*\* verify BGP routes on rtra \*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On rtrb: BGP peers OK

On rtrb: BGP routes OK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

On rtra: BGP peers OK

On rtra: BGP routes OK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class 7 - Bonus Lesson - Result Objects

lundi, 14 octobre 2019

15:07

1. In this example I am going to show you what are Result object we have and how to access their attributes. Here is my group\_vars and Python script

root@Nornir:~# cat group\_vars/DC.yaml

---

platform: nxos

connection\_options:

netmiko:

platform: cisco\_nxos

extras: {}

napalm:

port: 8443

extras:

optional\_args: {}

root@Nornir:~/Class7# cat napalm\_nxos\_ssh.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get

from pprint import pprint

def main():

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(name="dcsw1")

result = nr.run(task=napalm\_get, getters=["get\_mac\_address\_table"])

pprint(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

root@Nornir:~/Class7# python napalm\_nxos\_ssh.py

AggregatedResult (napalm\_get): {'dcsw1': MultiResult: [Result: "napalm\_get"]}

You can see I received a Aggregated result object which stores the Multiresult object and Multiresult object stores the Result. All these objects will have a "result", "failed" etc attributes. To know what are attributes, you need to dir().

To explain our output, "result" is an attribute of the "Result object" that is stored in the "MultiResult object".

To access the result attribute we need to access "Aggregatedresult[key].result", so in our case it will be "pprint(result['dcsw1'].result)"

So we have the script as

root@Nornir:~/Class7# cat napalm\_nxos\_ssh.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get

from nornir.plugins.functions.text import print\_result

from pprint import pprint

def main():

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(name="dcsw1")

result = nr.run(task=napalm\_get, getters=["get\_mac\_address\_table"])

pprint(result['dcsw1'].result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

root@Nornir:~/Class7# python napalm\_nxos\_ssh.py

{'get\_mac\_address\_table': [{'active': False,

'interface': 'sup-eth1(R)',

'last\_move': -1.0,

'mac': '0C:F3:6F:DF:00:2F',

'moves': -1,

'static': True,

'vlan': 0}]}

The easy way is frequently to loop over an aggregated result object (otherwise you need to know the key name).

for multi\_result in agg\_result:

  print(multi\_result[0].result)

Class 7 - Bonus Lesson - Direct Connection, SSH keys

lundi, 14 octobre 2019

18:56

1. In this example, I am going to show how to use nx-os SSH instead of nx-api when using the napalm\_plugin. To connect with a nxos device, normally we use nx-api with port 8443 when we use the napalm plugin but we have change this to use SSH. To do that we need to specify the platform as nxos\_ssh as the platform. We do that in the groups.yaml file but this can also be done at hosts.yaml file. This is applicable only for napalm because netmiko do not use nx-api, netmiko always use ssh.

root@Nornir:~# cat group\_vars/DC.yaml

---

platform: nxos

connection\_options:

netmiko:

platform: cisco\_nxos

extras: {}

napalm:

platform: nxos\_ssh

port: 22

extras:

optional\_args: {}

root@Nornir:~/Class7# cat napalm\_nxos\_ssh.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import napalm\_get

from pprint import pprint

def main():

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(name="dcsw1")

result = nr.run(task=napalm\_get, getters=["get\_mac\_address\_table"])

pprint(result['dcsw1'].result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

You can see I am able to access the Nexus using ssh and retrieve information and I did NOT use NX-API (port 8443)

root@Nornir:~/Class7# python napalm\_nxos\_ssh.py

{'get\_mac\_address\_table': [{'active': False,

'interface': 'sup-eth1(R)',

'last\_move': -1.0,

'mac': '0C:F3:6F:DF:00:2F',

'moves': -1,

'static': True,

'vlan': 0}]}

1. In this example, I am going to show how to use a direct task to achieve something without any napalm or netmiko task plugins imported to run a task. First let us look at a netmiko task without a netmiko plugin.

root@Nornir:~/Class7# cat netmiko\_direct.py

from nornir import InitNornir

def direct(task):

result = task.host.get\_connection("netmiko",task.nornir.config)

connect = result.send\_command("show run int vlan1")

print(connect)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(name="l2swa")

nr.run(task=direct)

You can see above I am sending a command to the device instead of using the plugins and you see below execution of this script.

root@Nornir:~/Class7# python netmiko\_direct.py

Building configuration...

Current configuration : 65 bytes

!

interface Vlan1

ip address 192.168.122.243 255.255.255.0

end

Here you will see a napalm task without a plugin and you will see a minor difference in the code.

root@Nornir:~/Class7# cat napalm\_direct.py

from nornir import InitNornir

def direct(task):

result = task.host.get\_connection("napalm",task.nornir.config)

connect = result.device

output = connect.send\_command("show run int mgmt0")

print(output)

connect.disconnect()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(name="dcsw1")

nr.run(task=direct)

root@Nornir:~/Class7# python napalm\_direct.py

!Command: show running-config interface mgmt0

!Time: Tue Oct 15 11:16:09 2019

version 7.3(0)D1(1)

interface mgmt0

vrf member management

ip address 192.168.122.246/24

1. I would like to use above netmiko example to send SSH keys to my devices so that going forward I do not use passwords for the login. This is going to iterate from the above example (ie) not using any plugins. When sending SSH keys we need to have some delay for the keys to be sent line by line because the keys are folded to fit the 254 character limit on the Cisco devices. You can see my script, I am sending SSH keys as multiline string (key-string) to the network device. The delay to send the input is achieved by the send\_command\_timing.

First let me run the following script to configure the SSH keys on my network devices. I have set num\_workers=1 to see how much time it takes to send the keys to all my four devices (L2SWA, L3SWA, RTRA, RTRB).

root@Nornir:~/BonusClass# cat netmiko\_send\_keys.py

from nornir import InitNornir

def prompt(task):

connect = task.host.get\_connection("netmiko",task.nornir.config)

print("\*\*\* send SSH Keys to",task.host.name,"\*\*\*")

print("-" \* 50)

cfg = """

conf t

ip ssh pubkey-chain

username cisco

key-string

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVutO1qnBGu5

/VJdE1OFXUouu3AsZ/vdqCr7j9QW8SBKagy5ZVr2ES2MoTGMlcgCzMMCLDItYG30Zg4Oysrp

S511Hb7vUs85Sy2eMvAdYic+HksnD2NcxcrN1tDN7jzQsjgY06sx2YsOGV9TLpYdjYsalc/H

UexSu1oVtIQGS5fXvrYb4f1QRmYy9fd6Xm78nyc3NZ7uD8fqCzdfJax9h9PPPYw7UoN4iiI5

ZW03zKbrlfgpLHpbSFMfw/a3ggRk2i20/aXhtmzDVu9e05zCyNsfh5C18BcG6tNUWNuAT2q8

w8Xx3wEkKvNd+2zC3D3R root@Nornir

exit

end

"""

output = connect.send\_command\_timing(cfg, strip\_prompt=False, strip\_command=False)

print(output)

print()

print("\*\*\* verify the SSH keys on",task.host.name,"\*\*\*")

print("-" \* 50)

verify = connect.send\_command(command\_string = "sh run | s key-chain", strip\_prompt=False, strip\_command=False)

print(verify)

print()

print("#" \* 20,"SSH KEYS OK on",task.host.name,"#" \* 20)

print()

print()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

print()

nr.run(task=prompt)

From the below screen shot, you can see my SSH keys are pushed successfully and I have verified if the keys are pushed to the Network device. You can see the time it took to complete this task is 1 minute and 1 seconds. For this scenario, I have kept the num\_workers=1.

Machine generated alternative text:
python 
*** send SSH Keys to rtra *** 
rtra#conf t 
• Nomir 
- ssh- 
- ssh- pubkey- 
- ssh- pubkey- 
• RTRB 
netmiko 
L2SWA 
send _ keys . py 
End with CNTL/Z. 
L3SWA 
• RTRA 
Alarms & Clock 
01. 
'O I .02 
Enter configuration commands, one per line. 
rtra(config)#ip ssh pubkey-chain 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra# 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
*** verify the SSH keys on rtra *** 
sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
rtra# 
#################### SSH KEYS OK on rtra #################### 
*** send SSH Keys to rtrb *** 
rtrb#conf t 
Enter configuration commands, one per line. 
rtrb(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb# 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
*** verify the SSH keys on rtrb *** 
sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
rtrb# 
#################### SSH KEYS OK on rtrb #################### 
*** send SSH Keys to 12swa *** 
L2SWA#conf t 
Enter configuration commands, one per line. 
L2SWA(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA# 
verify 
sh run 
s 
-ssh 
the 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
-ssh-pubkey-user)#end 
SSH keys on 12swa *** 
key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L2SWA# 
#################### SSH KEYS OK on 12swa #################### 
*** send SSH Keys to 13swa *** 
L3SWA#conf t 
Enter configuration commands, one per line. 
L3SWA(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA# 
verify 
sh run 
s 
-ssh 
the 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
SSH keys on 13swa *** 
key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L3SWA# 
#################### SSH KEYS OK on 13swa #################### 

Now let me run the same task by increasing the num\_workers to 100 and see the time it takes. It took only 29.86 seconds to complete the task.

Machine generated alternative text:
Nornir 
L2SWA#conf t 
• RTRB 
L2SWA 
End with CNTL/Z. 
L3SWA 
• RTRA 
x 
Enter configuration commands, one per line. 
L2SWA(config)#ip ssh pubkey-chain 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA# 
verify 
sh run 
s 
-ssh 
the 
-pubkey)#username cisco 
Alarms & Clock 
00:00: 
- ssh- pubkey- string 
-ssh-pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
-ssh-pubkey 
-ssh-pubkey 
-ssh-pubkey 
-ssh-pubkey 
-ssh-pubkey-data)#w8Xx3wEkKvNd+2zC3D3R root@Nornir 
-ssh-pubkey-user)#end 
SSH keys on 12swa *** 
key-chain 
29.86 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L3SWA# 
#################### SSH KEYS OK on 13swa #################### 
sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L2SWA# 
#################### SSH KEYS OK on 12swa #################### 

1. In this example, I am going to use the similar scenario, meaning sending the SSH keys but I am going to use cmd\_list and loop over to send command with a delay. I have configured the num\_workers=100 to spin up. You can see it took 31.28 seconds to complete the task.

root@Nornir:~/BonusClass# cat netmiko\_send\_keys\_cmdloop.py

from nornir import InitNornir

def prompt(task):

connect = task.host.get\_connection("netmiko",task.nornir.config)

print("\*\*\* send SSH Keys to",task.host.name,"and verify \*\*\*")

print("-" \* 50)

cfg = """

conf t

ip ssh pubkey-chain

username cisco

key-string

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVutO1qnBGu5

/VJdE1OFXUouu3AsZ/vdqCr7j9QW8SBKagy5ZVr2ES2MoTGMlcgCzMMCLDItYG30Zg4Oysrp

S511Hb7vUs85Sy2eMvAdYic+HksnD2NcxcrN1tDN7jzQsjgY06sx2YsOGV9TLpYdjYsalc/H

UexSu1oVtIQGS5fXvrYb4f1QRmYy9fd6Xm78nyc3NZ7uD8fqCzdfJax9h9PPPYw7UoN4iiI5

ZW03zKbrlfgpLHpbSFMfw/a3ggRk2i20/aXhtmzDVu9e05zCyNsfh5C18BcG6tNUWNuAT2q8

w8Xx3wEkKvNd+2zC3D3R root@Nornir

exit

end

"""

verify = "sh run | s key-chain"

cmd\_list = [cfg, "\n", verify, "\n"]

output = ""

for cmd in cmd\_list:

output += connect.send\_command\_timing(cmd, strip\_prompt=False, strip\_command=False)

print(output)

print("#" \* 20,"SSH KEYS OK on",task.host.name,"#" \* 20)

print()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

print()

nr.run(task=prompt)

Here I am running the above script. You can see it took 31.28 seconds to complete the task.

Machine generated alternative text:
rtrb#sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
rtrb# 
rtrb# 
#################### SSH KEYS OK on rtrb #################### 
send SSH 
Keys to rtra 
send SSH 
Keys to rtrb 
send SSH 
send SSH 
rtrb#conf t 
- ssh- pubkey- 
- ssh- pubkey- 
- ssh- 
- ssh- 
python py 
and * 
and * 
Alarms & Clock 
00:00: 
Keys to 13swa and verify *** 
Keys to 12swa and verify *** 
Enter configuration commands, one per line. 
rtrb(config)#ip ssh pubkey-chain 
31.28 
End with CNTL/Z. 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb# 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
rtra#conf t 
Enter configuration commands, one per line. 
rtra(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra# 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
rtra#sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
rtra# 
rtra# 
#################### SSH KEYS OK on rtra #################### 
L3SWA#conf t 
Enter configuration commands, one per line. 
L3SWA(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA# 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
L3SWA#sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L3SWA# 
L3SWA# 
#################### SSH KEYS OK on 13swa #################### 
L2SWA#conf t 
Enter configuration commands, one per line. 
L2SWA(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA# 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
L2SWA#sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L2SWA# 
L2SWA# 
#################### SSH KEYS OK on 12swa #################### 

I kept the num\_workers=1 and ran the same script that has cmd\_list and loops over to the next command with a delay.

You can see the time it took as 1min and 19 secs.

Machine generated alternative text:
Alarms & Clock 
*** send SSH Keys to rtra and verify *** 
rtra#conf t 
Enter configuration commands, one per line. 
rtra(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
01:19.12 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra(conf 
rtra# 
-ssh 
-pubkey)#username cisco 
rtra#sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
rtra# 
rtra# 
#################### SSH KEYS OK on rtra #################### 
*** send SSH Keys to rtrb and verify *** 
rtrb#conf t 
Enter configuration commands, one per line. 
rtrb(config)#ip ssh pubkey-chain 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb(conf 
rtrb# 
rtrb#sh run 
s key-chain 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
rtrb# 
rtrb# 
#################### SSH KEYS OK on rtrb #################### 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA(conf 
L2SWA# 
L2SWA#sh run 
ip ssh pubkey-chain 
username cisco 
key-hash ssh-rsa 7FAIF697CBA9B21D88BCF4B5BAB2E5EB root@Nornir 
L2SWA# 
L2SWA# 
#################### SSH KEYS OK on 12swa #################### 
*** send SSH Keys to 13swa and verify *** 
L3SWA#conf t 
Enter configuration commands, one per line. 
L3SWA(config)#ip ssh pubkey-chain 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA(conf 
L3SWA# 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
-ssh-pubkey-user)#end 
python py 
- ssh- 
- ssh- pubkey- 
- ssh- 
End with CNTL/Z. 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
*** send SSH Keys to 12swa and verify *** 
L2SWA#conf t 
Enter configuration commands, one per line. 
L2SWA(config)#ip ssh pubkey-chain 
End with CNTL/Z. 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- user)#key- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
- ssh- pubkey- 
s key-chain 
End with CNTL/Z. 
-ssh 
-pubkey)#username cisco 
- ssh- pubkey- string 
- ssh- pubkey 
AAAAB3NzaC1yc2EAAAADAQABAAABAQCwoSQbBsq1H4qOAqJFckp+ZVut01qnBGu5 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- ssh- pubkey 
- root@Nornir 
-ssh-pubkey-user)#end 
L3SWA#sh run 
s key-chain 
#################### SSH KEYS OK on 13swa #################### 

1. In this example, we will show how to delete files from the device flash. This is bit special because the router prompts for your input to confirm if the file can be deleted. To enter the required input such as yes [y] or no [n] we need some delay in sending that yes/no to the router, this is achieved with the netmiko\_send\_command\_timing" option. We are going to see how to use that to our benefit.

In our example, I am going to delete a file named os.txt from the network devices flash. You can see below from my netmiko-show output, I have that file in the flash of my network devices that belongs to the ios group.

root@Nornir:~/BonusClass# netmiko-show --cmd "dir" ios | grep os.txt

L2SWA.txt: 273 -rw- 17 Oct 17 2019 15:16:46 +00:00 os.txt

L3SWA.txt: 270 -rw- 21 Oct 17 2019 15:16:00 +00:00 os.txt

RTRA.txt: 272 -rw- 17 Oct 17 2019 15:37:58 +00:00 os.txt

RTRB.txt: 285 -rw- 17 Oct 17 2019 15:36:54 +00:00 os.txt

root@Nornir:~/BonusClass# cat netmiko\_prompting.py

from nornir import InitNornir

def prompt(task):

connect = task.host.get\_connection("netmiko",task.nornir.config)

filename = "os.txt"

del\_cmd = f"delete flash:/{filename}"

cmd\_list = ["\n", del\_cmd, "y"]

output = ""

for cmd in cmd\_list:

output += connect.send\_command\_timing(cmd, strip\_prompt=False, strip\_command=False)

print(output)

print("-" \* 30)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr.run(task=prompt)

You can see I am running the script below which has deleted the file os.txt automatically.

root@Nornir:~/BonusClass# python netmiko\_prompting.py

rtrb#delete flash:/os.txt

Delete flash0:/os.txt? [confirm]y

rtrb#

rtrb#

------------------------------

rtra#delete flash:/os.txt

Delete flash0:/os.txt? [confirm]y

rtra#

rtra#

------------------------------

L3SWA#delete flash:/os.txt

Delete flash0:/os.txt? [confirm]y

L3SWA#

L3SWA#

------------------------------

L2SWA#delete flash:/os.txt

Delete flash0:/os.txt? [confirm]y

L2SWA#

L2SWA#

------------------------------

To verify if the file has been deleted we run the netmiko-show command again and you need file has been deleted and no such file exist.

root@Nornir:~/BonusClass# netmiko-show --cmd "dir" ios | grep os.txt

root@Nornir:~/BonusClass#

1. In this example, I will show you how to use SSH keys instead of password. Before start using SSH keys we need to do the following steps.

root@Nornir:~# chmod 600 /root/.ssh/config

root@Nornir:~# chown root /root/.ssh/config

First prepare the router for SSH connection

hostname RTRA

ip domain-name vikiboy.net

username cisco privilege 15 password cisco

crypto key generate rsa

How many bits in the modulus [512]: 2048

After the above config, you will see the following displayed on your screen

%SSH-5-ENABLED: SSH 1.99 has been enabled

Ensure SSH connections are allowed on your Network Device.

line vty 0 4

login local

transport input ssh

On your Automation Control Server do this

[root@Nornir ~]# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory ‘/root/.ssh/id\_rsa’.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Enter password and confirm it. The password will used for the safety of the key. If you do not want any then you can simply leave this blank. Press enter.

The key is saved in /root/.ssh folder. .ssh is a hidden folder. In .ssh folder there is a file call id\_rsa.pub which is actually the public key.

Cisco CLI can contain maximum 254 characters in one line so it is not possible to paste the entire key in one line so you need to break the key in multiple line.

[root@Nornir ~]# fold -b -w 72 /root/.ssh/id\_rsa.pub

Copy the key into a notepad

On the Cisco router/switch do this,

RTRA(config)#ip ssh pubkey-chain

RTRA(conf-ssh-pubkey)#username cisco

RTRA(conf-ssh-pubkey-user)#key-string

After the key-string command press enter and then paste the copied key. After that again, press enter then type exit.

Now on the router, if you check, you will see the

RTRA(config)#do sh running-config | begin pubkey

ip ssh pubkey-chain

username cisco

key-hash ssh-rsa D6402CXXXXXXXXXXXXXXXXX root@Nornir

You cannot compare the key output shown by the router and the output show by the Automation server because router shows the fingerprint of that key and not the actual key.

[root@Nornir ~]# ssh-keygen -l -f .ssh/id\_rsa.pub

2048 SHA256:7XXMed2XXXXXXXXXXXXXXXXXXXXXXXXX root@Nornir (RSA)

Now when I try to login to the router, I do not need to enter the password.

[root@Nornir ~]# ssh cisco@RTRA

RTRA#

Now that we have prepared the SSH keys for the login, let us use that for the login in the scripts. The SSH keys are stored in the /root/.ssh/id\_rsa, so this path will be specified in the group\_vars.

root@Nornir:~# cat group\_vars/L2.yaml

---

platform: ios

connection\_options:

netmiko:

platform: cisco\_ios

extras:

use\_keys: True

key\_file: "/root/.ssh/id\_rsa"

I will not have my password in the defaults file or on the all.yaml file, you can see only username is specified

root@Nornir:~# cat group\_vars/all.yaml

---

username: cisco

Now I will run the same Nornir script that we used in previous example. You can see I am successfully able to execute the script which logins and deletes the os.txt file on the L2SWA.

root@Nornir:~/BonusClass# cat netmiko\_prompting.py

from nornir import InitNornir

def prompt(task):

connect = task.host.get\_connection("netmiko",task.nornir.config)

filename = "os.txt"

del\_cmd = f"delete flash:/{filename}"

cmd\_list = ["\n", del\_cmd, "y"]

output = ""

for cmd in cmd\_list:

output += connect.send\_command\_timing(cmd, strip\_prompt=False, strip\_command=False)

print(output)

print("-" \* 30)

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file= "/root/Nornir/config.yaml")

nr = nr.filter(name="l2swa")

nr.run(task=prompt)

root@Nornir:~/BonusClass# python netmiko\_prompting.py

L2SWA#delete flash:/os.txt

Delete flash0:/os.txt? [confirm]y

L2SWA#

L2SWA#

------------------------------

root@Nornir:~/BonusClass#

1. In this example, I am going to show what is the best way to print statements. Is it good to have in custom function or is it good to have in the main function.

First let us have the print statement in the custom function. I am just going to use very basic script.

root@Nornir:~/BonusClass# cat print\@main.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.core.filter import F

def custom (task):

host = task.host.hostname

print("\*\*\* Testing on",host,"\*\*\*")

print()

output = task.run(task=netmiko\_send\_command, command\_string = "sh run | inc hostname")

**print(output[0].result)**

print()

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file = "/root/Nornir/nornir.yaml")

nr = nr.filter(F(hostname="L2SWA") | F(hostname="L3SWA"))

print("-" \* 30)

nr = nr.run(task=custom)

You can see from the below output, due to the number of threads we are spunning, we receive output of L3SWA (hostname) first before the L2SWA output. This means the order of the output received/displayed in the terminal is not in the order of the thread that was sent to respective devices. In some huge configs it is annoying because we might get confused between the outputs received from the devices. For example, we see the first output as "Testing on L2SWA" but the subsequent output for L2SWA Ie hostname of L2SWA is received after the hostname of L3SWA.

root@Nornir:~/BonusClass# python print\@main.py

------------------------------

\*\*\* Testing on L2SWA \*\*\*

\*\*\* Testing on L3SWA \*\*\*

hostname L3SWA

hostname L2SWA

To avoid the above said issue in relation to the output displayed to the terminal, it is better to have the print statement in the main function.

root@Nornir:~/BonusClass# cat print\@main.py

from nornir import InitNornir

from nornir.plugins.tasks.networking import netmiko\_send\_command

from nornir.core.filter import F

def custom (task):

host = task.host.hostname

print("\*\*\* Testing on",host,"\*\*\*")

print()

output = task.run(task=netmiko\_send\_command, command\_string = "sh run | inc hostname")

if \_\_name\_\_ == "\_\_main\_\_":

nr = InitNornir(config\_file = "/root/Nornir/nornir.yaml")

nr = nr.filter(F(hostname="L2SWA") | F(hostname="L3SWA"))

print("-" \* 30)

nr = nr.run(task=custom)

**for hostname,output in nr.items():**

**print(output[1].result)**

print()

You can see from the below output the we are able get the output for relevant devices in order when we have used the print statement in main function. For example the first output "\*\*\* Testing on L2SWA \*\*\*" and subsequent output from the switch "hostname L2SWA" is displayed in the order as the threads are spun out to the devices.

root@Nornir:~/BonusClass# python print\@main.py

------------------------------

\*\*\* Testing on L2SWA \*\*\*

\*\*\* Testing on L3SWA \*\*\*

hostname L2SWA

hostname L3SWA