



PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA

Ciencias de la Ingeniería - Departamento de Ingeniería Telemática

Título:

Laboratorio Nornir-napalm

Estudiantes:

Wheliver Rivas

10140381

Asignatura:

CSTI-1870-5377

Automatización de

Redes

Docente:

Juan Alberto Corniel

Santiago de los Caballeros,

República Dominicana,

Octubre, 2024

Introducción

En esta práctica se utilizaron herramientas de automatización de redes como Nornir y NAPALM para administrar dispositivos de red con Python. Nornir es una biblioteca que permite ejecutar tareas simultáneamente en varios dispositivos, mientras que NAPALM brinda soporte para conectar y gestionar diversas plataformas de red. En este contexto, se definieron inventarios de dispositivos mediante archivos YAML y se usó Netmiko para ejecutar comandos en los dispositivos. El propósito de este laboratorio fue mostrar la eficiencia de la automatización en la gestión de infraestructuras de red y cómo Python puede simplificar la administración de dispositivos a gran escala.

Desarrollo

Paso 1: Instalación de Nornir y NAPALM

1. **Instalar Nornir y NAPALM** en tu entorno de Python ejecutando los siguientes comandos:

pip install nornir

pip install nornir-utils

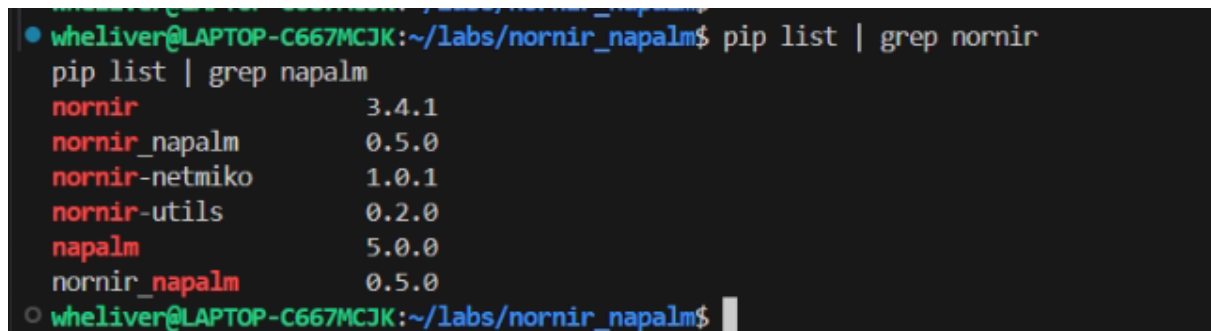
pip install nornir-napalm

pip install napalm

2. **Verifica la instalación:**

pip list | grep nornir

pip list | grep napalm

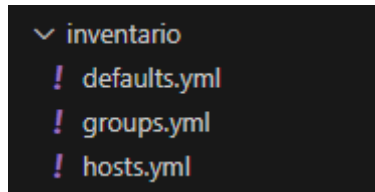


```
wheliver@LAPTOP-C667MCJK:~/labs/nornir_napalm$ pip list | grep nornir
pip list | grep napalm
nornir                3.4.1
nornir_napalm         0.5.0
nornir-netmiko        1.0.1
nornir-utils          0.2.0
napalm                5.0.0
nornir_napalm         0.5.0
wheliver@LAPTOP-C667MCJK:~/labs/nornir_napalm$
```

Captura de pantalla de las dependencias instaladas en el entorno virtual del laboratorio. Se confirma que las bibliotecas `nornir-napalm`, `nornir-netmiko` y `nornir-utils` están correctamente instaladas en las versiones necesarias, lo cual resulta esencial para la automatización de redes en Python mediante NAPALM y Nornir.

Paso 2: Crear la estructura de archivos de configuración para Nornir

Nornir utiliza varios archivos YAML para gestionar el inventario y la configuración. Vamos a crear los archivos necesarios: config.yaml, hosts.yaml, groups.yaml, y defaults.yaml.



Pantallazo de la estructura del directorio inventario, el cual contiene los archivos YAML necesarios para la configuración del inventario en Nornir. Los archivos defaults.yaml, groups.yaml, y hosts.yaml definen las configuraciones por defecto, los grupos de dispositivos y la lista de hosts respectivamente. Estos archivos son esenciales para que Nornir pueda gestionar la automatización de dispositivos de red en el laboratorio.

2.1 Crear el archivo config.yaml

Este archivo define cómo Nornir gestionará el inventario y las tareas.

1. Crea el archivo config.yaml:

```
1 inventory:
2   plugin: SimpleInventory
3   options:
4     host_file: "/home/wheliver/labs/inventario/hosts.yaml"
5     group_file: "/home/wheliver/labs/inventario/groups.yaml"
6     defaults_file: "/home/wheliver/labs/inventario/defaults.yaml"
7
8 runner:
9   plugin: threaded
10  options:
11    num_workers: 100
12
```

Captura de pantalla del archivo de configuración `config.yaml`, donde se definen los parámetros de inventario y ejecución para Nornir. En la sección de `inventory`, se configura el uso del plugin `SimpleInventory`, especificando las rutas hacia los archivos `hosts.yaml`, `groups.yaml` y `defaults.yaml`, los cuales componen la estructura del inventario. En la sección `runner`, se establece el uso del plugin `threaded` con una configuración de 100 `workers`, permitiendo la ejecución concurrente de tareas en varios dispositivos de red.

2.2 Crear el archivo `hosts.yaml`

Este archivo contendrá la información específica de cada dispositivo en el inventario.

1. **Crea el archivo `hosts.yaml`:**

```

1  # Nornir Hosts File
2  ---
3
4  R53:
5      hostname: 192.168.100.121
6      port: 33333
7      platform: cisco_ios_telnet
8      username: cisco
9      password: cisco
10     groups:
11         - Routers
12         - Juan
13
14     SW32:
15         hostname: 192.168.100.121
16         port: 33332
17         platform: cisco_ios_telnet
18         username: cisco
19         password: cisco
20         groups:
21             - Switches
22             - Juan
23
24     ARISTA:
25         hostname: 192.168.100.121
26         port: 33335
27         platform: arista_eos_telnet
28         username: Arista
29         password: arista
30         groups:
31             - Switches
32             - Santiago
33         connection_options:
34             napalm:
35                 extras:
36                     optional_args:
37                         transport: "telnet"
38
39     Sandbox_Cisco_XE:
40         hostname: devnetsandboxiosxe.cisco.com
41         port: 22
42         platform: ios
43         username: admin
44         password: Cisco12345
45         groups:
46             - Switches
47             - Juan
48
49     Sandbox_Cisco_XR:
50         hostname: sandbox-iosxr-1.cisco.com
51         port: 22
52         platform: ios
53         username: admin
54         password: Cisco12345
55         groups:
56             - Switches
57             - Juan
58

```

Captura de pantalla del archivo `hosts.yaml`, que especifica los dispositivos administrados en el laboratorio de automatización de redes con Nornir. Este archivo incluye tres dispositivos: `R53`, `SW32` y `ARISTA`, cada uno con su configuración particular de `hostname`, `port`, `platform`, `username` y `password`. Los dispositivos están organizados en grupos, como `Routers` y `Switches`, lo cual facilita su administración en las tareas automatizadas. En el

caso del dispositivo `ARISTA`, se indican configuraciones adicionales de conexión para NAPALM, donde el transporte se establece como `telnet`.

2.3 Crear el archivo groups.yaml

Los grupos permiten definir configuraciones comunes para varios dispositivos.

1. Crea el archivo groups.yaml:

```
inventario > ! groups.yaml
1  ## Nornir Groups File
2  ---
3
4  Routers:
5    data:
6      ntp:
7        servers:
8          - 1.1.1.1
9
10 Switches:
11   data:
12     ntp:
13       servers:
14         - 2.2.2.2
15
16 Juan:
17   data:
18     site: Navarrete
19     syslog:
20       servers:
21         - 3.3.3.3
22
23 Santiago:
24   data:
25     site: Santiago
26     syslog:
27       servers:
28         - 1.1.1.1
29         - 4.4.4.4
30
```

Captura de pantalla del archivo `groups.yaml`, donde se definen los grupos de dispositivos en la configuración de Nornir. Los grupos `Routers`, `Switches`, `Juan` y `Santiago` incluyen información específica, como servidores NTP y Syslog. Cada grupo cuenta con configuraciones comunes que pueden aplicarse a todos los dispositivos dentro de dicho grupo, simplificando la automatización de configuraciones en múltiples dispositivos de red de manera eficiente.

2.4 Crear el archivo defaults.yaml

Define valores por defecto para los dispositivos.

1. Crea el archivo defaults.yaml:

```
inventario > ! defaults.yaml
1  ## Nornir Defaults File
2  ---
3
4  platform: cisco_ios_telnet
5  username: cisco
6  password: cisco
7
```

Captura de pantalla del archivo `defaults.yaml`, que establece los valores predeterminados para los dispositivos administrados en el laboratorio de automatización con Nornir. Este archivo define `cisco_ios_telnet` como la plataforma por defecto, y utiliza `cisco` tanto como nombre de usuario como contraseña por defecto. Esto facilita la configuración de dispositivos al aplicar valores estándar cuando no se proporcionan otros específicos en los archivos de inventario.

Paso 3: Crear un script Python para cargar el inventario

1. Crea el archivo nornir_inventory.py:


```
lab_nornir > nornir_inventory.py > ...
1  from nornir import InitNornir
2  from nornir_utils.plugins.functions import print_result
3
4  # Inicializa Nornir con el archivo de configuración
5  nr = InitNornir(config_file="config.yaml")
6
7  # Muestra el inventario de dispositivos
8  print(nr.inventory.hosts)
9

● wheliver@LAPTOP-C667MCJK:~/labs/nornir_napalm$ python3 nornir_inventory.py
{'R53': Host: R53, 'SW32': Host: SW32, 'ARISTA': Host: ARISTA, 'Sandbox_Cisco_XE': Host: Sandbox_Cisco_XE, 'Sandbox_Cisco_XR': Host: Sandbox_Cisco_XR}
○ wheliver@LAPTOP-C667MCJK:~/labs/nornir_napalm$
```

Captura de pantalla del script `nornir_inventory.py`, que inicializa Nornir empleando el archivo de configuración `config.yaml` y muestra el inventario de dispositivos. Se importa la función `print_result` del módulo `nornir_utils.plugins.functions` y, al ejecutar el script, se imprime el inventario usando `print(nr.inventory.hosts)`. En la salida del script, se visualizan los dispositivos configurados, como `R53`, `SW32`, `ARISTA`, entre otros.

Paso 4: Conectar a los dispositivos en EVE-NG y ejecutar comandos

En este paso, vamos a ejecutar un comando simple en los dispositivos conectados.

1. **Crea el archivo `nornir_commands.py`:**

```
1 from nornir import InitNornir
2 from nornir_utils.plugins.functions import print_result
3 from nornir_netmiko import netmiko_send_command
4
5 # Inicializa Nornir con el archivo de configuración
6 nr = InitNornir(config_file="config.yaml")
7
8 # Ejecuta el comando en todos los dispositivos
9 result = nr.run(
10     task=netmiko_send_command,
11     command_string="show ip int br"
12 )
13
14 # Muestra el resultado
15 print_result(result)
16
```

```

^END netmiko_send_command *****
* R53 ** changed : False *****
vvvv netmiko_send_command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Interface          IP-Address      OK? Method Status           Protocol
Ethernet0/0        unassigned     YES unset   administratively down down
Ethernet0/1        unassigned     YES unset   administratively down down
Ethernet0/2        unassigned     YES unset   administratively down down
Ethernet0/3        unassigned     YES unset   administratively down down
R53>
^^^^ END netmiko_send_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* SW32 ** changed : False *****
vvvv netmiko_send_command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Interface          IP-Address      OK? Method Status           Protocol
Ethernet0/0        unassigned     YES unset   up                up
Ethernet0/1        unassigned     YES unset   up                up
Ethernet0/2        unassigned     YES unset   up                up
Ethernet0/3        unassigned     YES unset   up                up
S52>
^^^^ END netmiko_send_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* Sandbox_Cisco_XE ** changed : False *****
vvvv netmiko_send_command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Interface          IP-Address      OK? Method Status           Protocol
GigabitEthernet1   10.10.20.48    YES NVRAM   up                up
GigabitEthernet2   192.168.1.100  YES manual administratively down down
GigabitEthernet3   unassigned     YES NVRAM   administratively down down
Loopback0          10.0.0.1       YES NVRAM   up                up
Loopback10         unassigned     YES unset   up                up
Loopback11         10.11.1.1      YES other  up                up
VirtualPortGroup0  192.168.1.1    YES NVRAM   up                up
^^^^ END netmiko send command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```



```

1  from nornir import InitNornir
2  from nornir.core.task import Task, Result
3  from nornir_utils.plugins.functions import print_result
4  from nornir_napalm.plugins.tasks import napalm_get
5
6
7
8  # Función para enviar un comando a través de NAPALM
9  def send_command(task: Task) -> Result:
10     # Recuperar información usando NAPALM
11     result = task.run(task=napalm_get, getters=["facts", "interfaces"])
12     return Result(host=task.host, result=result)
13
14 # Inicializar Nornir
15 nr = InitNornir(config_file="config.yaml")
16
17 # Ejecutar la tarea en todos los hosts
18 result = nr.run(task=send_command)
19
20 # Mostrar los resultados
21 print_result(result)
22

```

```
* Sandbox_Cisco_XE ** changed : False *****  
www send_command ** changed : False ~~~~~ INFO  
MultiResult: [Result: "napalm_get"]  
---- napalm_get ** changed : False ----- INFO  
{ 'facts': { 'fqdn': 'Lenkov.cisco.com',  
             'hostname': 'Lenkov',  
             'interface_list': [ 'GigabitEthernet1',  
                                  'GigabitEthernet2',  
                                  'GigabitEthernet3',  
                                  'Loopback0',  
                                  'Loopback10',  
                                  'Loopback11',  
                                  'VirtualPortGroup0'],  
             'model': 'C8000V',  
             'os_version': 'Virtual XE Software '  
                             '(X86_64_LINUX_IOSD-UNIVERSALK9-M), Version '  
                             '17.12.2, RELEASE SOFTWARE (fc2)',  
             'serial_number': '9OBXJHNNU5V',  
             'uptime': 6840.0,  
             'vendor': 'Cisco'},  
  'interfaces': { 'GigabitEthernet1': { 'description': 'MANAGEMENT INTERFACE - '  
                                         '"DON'T TOUCH ME",  
                                         'is_enabled': True,  
                                         'is_up': True,  
                                         'last_flapped': -1.0,  
                                         'mac_address': '00:50:56:BF:BF:E7',  
                                         'mtu': 1500,  
                                         'speed': 1000.0},  
    'GigabitEthernet2': { 'description': 'Network Interface',  
                          'is_enabled': False,  
                          'is_up': False,  
                          'last_flapped': -1.0,  
                          'mac_address': '00:50:56:BF:82:F3',  
                          'mtu': 1500,  
                          'speed': 1000.0},  
    'GigabitEthernet3': { 'description': 'Network Interface',
```

```
        'speed': 1000.0},
'GigabitEthernet3': { 'description': 'Network Interface',
                      'is_enabled': False,
                      'is_up': False,
                      'last_flapped': -1.0,
                      'mac_address': '00:50:56:BF:32:CF',
                      'mtu': 1500,
                      'speed': 1000.0},
'Loopback0': { 'description': '',
               'is_enabled': True,
               'is_up': True,
               'last_flapped': -1.0,
               'mac_address': '',
               'mtu': 1514,
               'speed': 8000.0},
'Loopback10': { 'description': '',
                'is_enabled': True,
                'is_up': True,
                'last_flapped': -1.0,
                'mac_address': '',
                'mtu': 1514,
                'speed': 8000.0},
'Loopback11': { 'description': 'Lenkov',
                'is_enabled': True,
                'is_up': True,
                'last_flapped': -1.0,
                'mac_address': '',
                'mtu': 1514,
                'speed': 8000.0},
'VirtualPortGroup0': { 'description': '',
                       'is_enabled': True,
                       'is_up': True,
                       'last_flapped': -1.0,
                       'mac_address': '00:1E:7A:85:31:BD',
                       'mtu': 1500,
                       'speed': 333.0}}}
```

```

#### END send_command #####
* Sandbox_Cisco_XR ** changed : False *****
vvvv send_command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
MultiResult: [Result: "napalm_get"]
---- napalm_get ** changed : False ----- INFO
{ 'facts': { 'fqdn': 'System.not set',
             'hostname': 'System',
             'interface_list': [ 'Bundle-Ether1',
                                  'Bundle-Ether6',
                                  'Bundle-Ether6.6477',
                                  'Loopback0',
                                  'Loopback1',
                                  'Loopback2',
                                  'Loopback4',
                                  'Loopback6',
                                  'Loopback10',
                                  'tunnel-ip3',
                                  'MgmtEth0/RP0/CPU0/0',
                                  'GigabitEthernet0/0/0/0',
                                  'GigabitEthernet0/0/0/1',
                                  'GigabitEthernet0/0/0/2',
                                  'GigabitEthernet0/0/0/3',
                                  'GigabitEthernet0/0/0/4',
                                  'GigabitEthernet0/0/0/5',
                                  'GigabitEthernet0/0/0/6'],
             'model': 'Unknown',
             'os_version': 'Unknown',
             'serial_number': 'Unknown',
             'uptime': 1209600.0,
             'vendor': 'Cisco'},
  'interfaces': { 'Bundle-Ether1': { 'description': '',
                                     'is_enabled': True,
                                     'is_up': False,
                                     'last_flapped': -1.0,
                                     'mac_address': '00:17:95:00:8F:35',
                                     'mtu': 1514,
                                     'speed': 0.0},

```

En la primera imagen, se observa la salida de la ejecución de un script en Nornir que utiliza NAPALM para obtener información detallada de las interfaces de un dispositivo llamado `Sandbox_Cisco_XR`. Los datos recuperados incluyen una lista de interfaces, como `Loopback` y `GigabitEthernet`, con detalles adicionales sobre el estado y características de cada interfaz. Algunas interfaces aparecen como administrativamente apagadas, mientras que otras, como ciertas `Loopback`, están activas.

En la segunda imagen, se muestra el código del script `nornir_napalm.py`. Este script inicializa Nornir con un archivo de configuración (`config.yaml`), define una función `send_command` que utiliza `napalm_get` para recuperar información de "facts" e "interfaces" de los dispositivos en el inventario, y finalmente ejecuta esta función en todos los hosts configurados. El resultado se imprime utilizando la función `print result`, lo cual

permite visualizar la información de cada dispositivo, como se muestra en la primera imagen.

Conclusión

A lo largo de esta práctica, se presentaron varios desafíos técnicos, como problemas de autenticación en los dispositivos de red y errores en la configuración de los archivos YAML. Sin embargo, estos obstáculos se resolvieron ajustando las credenciales de acceso y la estructura de los archivos de configuración, lo cual permitió ejecutar correctamente los comandos en los dispositivos. Estas dificultades resaltan la importancia de una configuración precisa al trabajar con herramientas de automatización de redes.

En el futuro, la automatización de redes ofrece grandes posibilidades para optimizar la gestión y configuración de dispositivos en entornos complejos. La integración de más herramientas y plataformas, como NAPALM, permitirá llevar a cabo tareas de manera más eficiente, reduciendo la intervención manual y minimizando los errores humanos. Este laboratorio refuerza la necesidad de seguir explorando y mejorando las capacidades de automatización para optimizar la administración de infraestructuras de red a gran escala.