

14/11/2025

# TP 03 Module Devnet

Automatisation Réseau

Ronan FOURNEUVE  
CPE LYON



## Table des matières

Partie 1 Générer et déployer des configurations réseau avec Python et Nornir .....	3
1-Automatiser le déploiement de configuration réseau sur des équipements Cisco....	3
1.8 -Génération automatique de la configuration à l'aide de Jinja2 (voir TP-01 et TP-02) .....	3
1.9 -Automatisation réseau avec Nornir .....	5

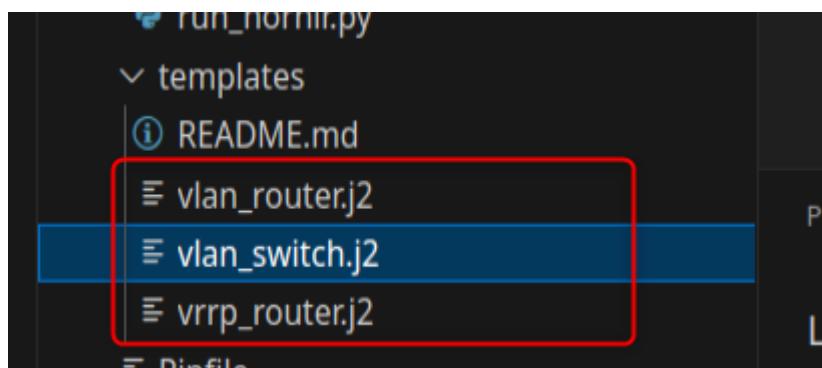
# Partie 1 Générer et déployer des configurations réseau avec Python et Nornir

## 1-Automatiser le déploiement de configuration réseau sur des équipements Cisco

1.8 -Génération automatique de la configuration à l'aide de Jinja2 (voir TP-01 et TP-02)

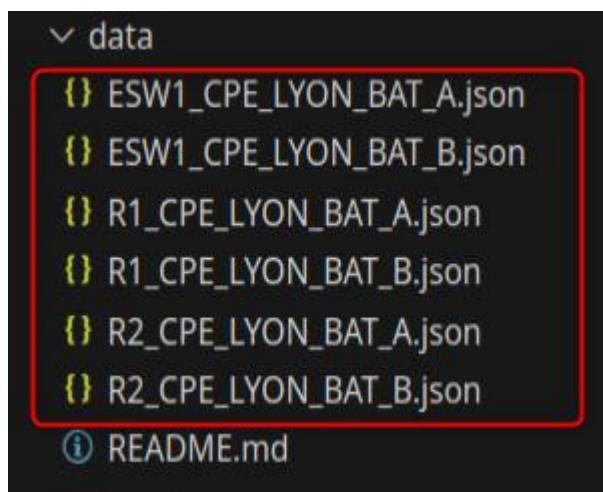
**Question 1.8.1 :** Développez l'ensemble des templates Jinja2 qui vous permettront de générer la configuration nécessaire pour le bon fonctionnement du bâtiment A. (Pour rappel, le vlan d'admin (99) est déjà configuré sur les équipements.) Les templates Jinja2 doivent être sauvegardés dans le dossier templates.

Les templates Jinja2 sont fournis dans le dossier *templates*.



**Question 1.8.2 :** Définissez les structures de données dans des fichiers JSON ou YAML (au choix) nécessaires pour remplir votre template Jinja2 développé à la question 1. (Pour rappel, le vlan d'admin (99) est déjà configuré sur les équipements.) Les fichiers doivent être stockés dans le dossier data

Les structures de données sont fournies dans le dossier *data*.



**Question 1.8.3 :** Développez les fonctions vous permettant de générer automatiquement les configurations à partir du template Jinja2 et de la structure de données que vous avez définis à la question 1 et 2. Écrivez votre code dans le fichier scripts/create\_config.py.

```
def create_config_cpe_lyon_batA():
    ESW1_CPE_LYON_BAT_A_data= load_json_data_from_file(file_path='data/ESW1_CPE_LYON_BAT_A.json')
    ESW1_CPE_LYON_BAT_A_config = render_network_config(template_name='vlan_switch.j2', data=ESW1_CPE_LYON_BAT_A_data)

    R2_LYON_BAT_A_data = load_json_data_from_file(file_path='data/R2_CPE_LYON_BAT_A.json')
    R2_LYON_BAT_A_config = render_network_config(template_name='vlan_router.j2', data=R2_LYON_BAT_A_data)

    R1_LYON_BAT_A_data = load_json_data_from_file(file_path='data/R1_CPE_LYON_BAT_A.json')
    R1_LYON_BAT_A_config = render_network_config(template_name='vlan_router.j2', data=R1_LYON_BAT_A_data)

    return ESW1_CPE_LYON_BAT_A_config,R1_LYON_BAT_A_config,R2_LYON_BAT_A_config
```

**Question 1.8.4 :** Développez les fonctions vous permettant de sauvegarder automatiquement les configurations générées à la question 3 dans le dossier config de votre workspace (TP03).

```
def create_config_cpe_lyon_batA():
    ESW1_CPE_LYON_BAT_A_data= load_json_data_from_file(file_path='data/ESW1_CPE_LYON_BAT_A.json')
    ESW1_CPE_LYON_BAT_A_config = render_network_config(template_name='vlan_switch.j2', data=ESW1_CPE_LYON_BAT_A_data)

    R2_LYON_BAT_A_data = load_json_data_from_file(file_path='data/R2_CPE_LYON_BAT_A.json')
    R2_LYON_BAT_A_config = render_network_config(template_name='vlan_router.j2', data=R2_LYON_BAT_A_data)

    R1_LYON_BAT_A_data = load_json_data_from_file(file_path='data/R1_CPE_LYON_BAT_A.json')
    R1_LYON_BAT_A_config = render_network_config(template_name='vlan_router.j2', data=R1_LYON_BAT_A_data)

    return [
        'esw1': ESW1_CPE_LYON_BAT_A_config,
        'r1': R1_LYON_BAT_A_config,
        'r2': R2_LYON_BAT_A_config
    ]
```

```
#question 3:
config = create_config_cpe_lyon_batA()

#question 4:
save_built_config('config/R1_CPE_LYON_BAT_A.conf', config.get('r1'))
save_built_config('config/R2_CPE_LYON_BAT_A.conf', config.get('r2'))
save_built_config('config/ESW1_CPE_LYON_BAT_A.conf', config.get('esw1'))
```

config	
⚙️	ESW1_CPE_LYON_BAT_A.conf
⚙️	R1_CPE_LYON_BAT_A.conf
⚙️	R2_CPE_LYON_BAT_A.conf

**Question 1.8.5 :** Répétez les questions 1 à 4 pour le bâtiment B en vous appuyant sur les données du tableau ci-dessous

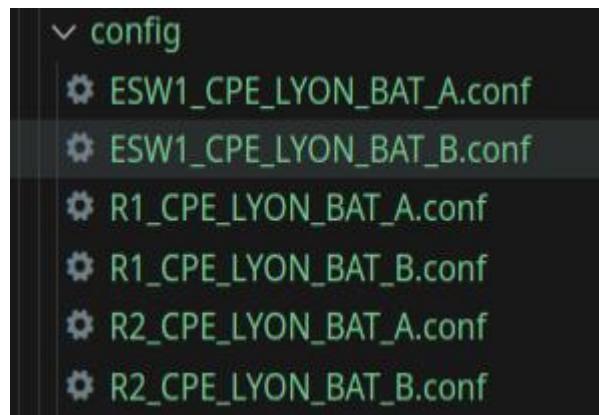
```
def create_config_cpe_lyon_batB():
    ESW1_CPE_LYON_BAT_B_data= load_json_data_from_file(file_path='data/ESW1_CPE_LYON_BAT_B.json')
    ESW1_CPE_LYON_BAT_B_config = render_network_config(template_name='vlan_switch.j2', data=ESW1_CPE_LYON_BAT_B_data)

    R2_LYON_BAT_B_data = load_json_data_from_file(file_path='data/R2_CPE_LYON_BAT_B.json')
    R2_LYON_BAT_B_config = render_network_config(template_name='vlan_router.j2', data=R2_LYON_BAT_B_data)

    R1_LYON_BAT_B_data = load_json_data_from_file(file_path='data/R1_CPE_LYON_BAT_B.json')
    R1_LYON_BAT_B_config = render_network_config(template_name='vlan_router.j2', data=R1_LYON_BAT_B_data)

    return {
        'esw1': ESW1_CPE_LYON_BAT_B_config,
        'r1': R1_LYON_BAT_B_config,
        'r2': R2_LYON_BAT_B_config
    }
```

```
#question 5:
config = create_config_cpe_lyon_batB()
save_builtin_config('config/R1_CPE_LYON_BAT_B.conf', config.get('r1'))
save_builtin_config('config/R2_CPE_LYON_BAT_B.conf', config.get('r2'))
save_builtin_config('config/ESW1_CPE_LYON_BAT_B.conf', config.get('esw1'))
```



## 1.9 -Automatisation réseau avec Nornir

**Question 1.9.6 :** Installez nornir dans votre workspace (TP03):

Nous installons le module Python Normir dans l'environnement virtuel géré par Pipenv

```
pipenv install nornir
```

**Question 1.9.7 :** Créez un fichier nommé run\_nornir.py dans le dossier scripts de votre workspace (TP03) et importez-le package nornir en en-tête du fichier

```
create_config.py run_nornir.py M X
devnet > TP_03 > scripts > run_nornir.py
1  from nornir import InitNornir
2
3  def question_13(nr):
```

**Question 1.9.8 :** Créez un fichier config.yaml dans le dossier inventory de votre workspace et ajoutez la config suivante

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a workspace named "devnet" containing a "TP\_03" project. Inside "TP\_03", there are "config", "data", and "scripts" folders. The "inventory" folder is expanded, showing files like ESW1\_CPE\_LYON\_BAT\_A.json, R1\_CPE\_LYON\_BAT\_A.json, etc. A "config.yaml" file is visible in the "inventory" folder.
- Code Editor:** Displays the content of "config.yaml". The code is highlighted with a red border.

```
1 inventory:
2   plugin: SimpleInventory
3   options:
4     host_file: "inventory/hosts.yaml"
5     group_file: "inventory/groups.yaml"
6     defaults_file: "inventory/defaults.yaml"
7   runner:
8     plugin: threaded
9     options:
10       num_workers: 20
```

**Question 1.9.9 :** Créez le fichier hosts.yaml dans le dossier inventory et ajoutez la structure de données suivante :

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a workspace named "devnet" containing a "TP\_03" project. Inside "TP\_03", there are "config", "data", and "scripts" folders. The "inventory" folder is expanded, showing files like "config.yaml" and "hosts.yaml".
- Code Editor:** Displays the content of "hosts.yaml". The code is highlighted with a red border.

```
1 R1-CPE-BAT-A:
2   hostname: 172.16.100.125
3   port: 22
4   groups:
5     - ios
6   data: # Anything under this key is custom data
7     device_name: R1-CPE-BAT-A
8     device_type: router
9     device_model: C7200
10    locality: lyon
11    building: A
12
13 R2-CPE-BAT-A:
14   hostname: 172.16.100.126
15   port: 22
16   groups:
17     - ios
18   data: # Anything under this key is custom data
19     device_name: R2-CPE-BAT-A
20     device_type: router
21     device_model: C7200
22     locality: lyon
```

**Question 1.9.10 :** Créez le fichier groups.yaml dans le dossier inventory et ajoutez la structure de données suivante.

The screenshot shows a dark-themed VS Code interface. In the top right, there are tabs for 'hosts.yaml' and 'groups.yaml'. The 'groups.yaml' tab is active, showing the following YAML content:

```
! hosts.yaml u ! groups.yaml x
devnet > TP_03 > inventory > ! groups.yaml
1   ios:
2     platform: ios
3     data:
4       vendor: Cisco
```

In the bottom left, the Explorer sidebar shows a project structure under 'WORKSPACE' named 'devnet'. It includes a 'TP\_03' folder containing 'config' and 'data' subfolders, and a 'scripts' folder. The 'inventory' folder is expanded, showing 'config.yaml', 'groups.yaml' (which is selected), 'hosts.yaml', and 'README.md'. A red box highlights the 'groups.yaml' file in the Explorer, and a red arrow points from it to the 'groups.yaml' tab in the top right.

**Question 1.9.11 :** Créez le fichier defaults.yaml dans le dossier inventory et ajoutez la structure de données suivante.

The screenshot shows a dark-themed VS Code interface. In the top right, there are tabs for 'hosts.yaml', 'groups.yaml', and 'defaults.yaml'. The 'defaults.yaml' tab is active, showing the following YAML content:

```
! hosts.yaml u ! groups.yaml u ! defaults.yaml x
devnet > TP_03 > inventory > ! defaults.yaml
1   username: cisco
2   password: cisco
```

In the bottom left, the Explorer sidebar shows a project structure under 'WORKSPACE' named 'devnet'. It includes a 'TP\_03' folder containing 'config' and 'data' subfolders, and a 'scripts' folder. The 'inventory' folder is expanded, showing 'config.yaml', 'defaults.yaml' (which is selected), 'groups.yaml', 'hosts.yaml', and 'README.md'. A red box highlights the 'defaults.yaml' file in the Explorer, and a red arrow points from it to the 'defaults.yaml' tab in the top right.

**Question 1.9.12 :** Vous pouvez à présent initialiser Nornir dans le \_\_main\_\_ du script run\_nornir.py de la manière suivante

```

! hosts.yaml   ! groups.yaml   ! defaults.yaml   run_nornir.py X
devnet > TP_03 > scripts > run_nornir.py
82
83     def question_40(nr):
84         pass
85
86
87     if __name__ == "__main__":
88         nr = InitNornir(config_file="inventory/config.yaml")
89

```

**Question 1.9.13 :** Quels sont les attributs de l'objet nr ? Pour connaître les attributs de l'objet nr utilisez l'attribut `__dict__` (utilisable sur tout objet python). Quel est le format de données de sortie ? Quelles sont les attributs de l'objet nr ? A votre avis lequel nous permettrait d'aller lire l'inventaire que nous avons précédemment défini (question 9) ?

```

def question_13(nr):
    print(nr.__dict__)

```

```

{'data': <nornir.core.state.GlobalState object at 0x78909b3a77c0>, 'inventory': <nornir.core.inventory.Inventory object at 0x78909af0ab00>, 'config': <nornir.core.configuration.Config object at 0x78909a6e9580>, 'processors': [], '_runner': <nornir.plugins.runners.ThreadedRunner object at 8909af96240>}
<class 'nornir.core.Nornir'>
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
{'data': <nornir.core.state.GlobalState object at 0x7f6e86de35b0>, 'inventory': <nornir.core.inventory.Inventory object at 0x7f6e8627d240>, 'config': <nornir.core.configuration.Config object at 0x7f6e86243010>, 'processors': [], '_runner': <nornir.plugins.runners.ThreadedRunner object at 0xf6e868f5a90>}
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ 

```

L'objet nr est une instance de la classe Nornir, et son attribut spécial `__dict__` retourne un dictionnaire Python dont les valeurs sont des instances de classes internes à Nornir ; parmi ces attributs, celui qui permet d'accéder à l'inventaire précédemment défini est `inventory`.

Pour voir le contenu du dictionnaire je réalise une boucle For

```

def question_13(nr):
    for key in nr.__dict__.keys():
        print(f" - {key}")

```

```
<class 'nornir.core.Nornir'>
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
- data
- inventory
- config
- processors
- _runner
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

- **data** : contient l'état global de Nornir, partagé entre les tâches durant l'exécution.
- **inventory** : contient tout l'inventaire (hosts, groupes, valeurs par défaut) chargé depuis les fichiers de configuration.
- **config** : regroupe la configuration complète de Nornir (chemins, options d'exécution, paramètres de l'inventaire, etc.).
- **processors** : liste les processeurs qui peuvent intercepter et modifier l'exécution des tâches.
- **\_runner** : définit la manière dont les tâches sont exécutées, par exemple en parallèle via un ThreadedRunner.

Parmi eux, l'attribut qui permet d'accéder à l'inventaire défini précédemment est `inventory`, car c'est lui qui contient la structure des hosts, groupes et valeurs par défaut chargés par Nornir.

**Question 1.9.14 :** Affichez l'attribut `hosts` de l'attribut que vous avez trouvé à la question 13. Quelles sont les données retournées ? Quel est le format de données retourné.

```
def question_14(nr):
    print(nr.inventory.hosts)
    print(type(nr.inventory.hosts))
```

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
{'R1-CPE-BAT-A': Host: R1-CPE-BAT-A, 'R2-CPE-BAT-A': Host: R2-CPE-BAT-A, 'ESW1-CPE-BAT-A': Host: ESW1-CPE-BAT-A, 'R1-CPE-BAT-B': Host: R1-CPE-BAT-B,
 'R2-CPE-BAT-B': Host: R2-CPE-BAT-B, 'ESW1-CPE-BAT-B': Host: ESW1-CPE-BAT-B}
<class 'nornir.core.inventory.Hosts'>
```

L'attribut `hosts` retourne un dictionnaire où chaque clé est le nom d'un équipement (ex : R1-CPE-BAT-A) et chaque valeur est un objet de type `Host` représentant cet équipement. Le format de données retourné est donc un dictionnaire spécialisé, de type `nornir.core.inventory.Hosts`, qui se comporte comme un dict Python contenant des objets `Host`.

**(Liste de hosts)**

**Question 1.9.15 :** Affichez la valeur du premier élément de l'objet à la question 14. Quelle est la valeur retournée ? Quel est le type de cette valeur (fonction `type()`) ?

```

def question_15(nr):
    print(nr.inventory.hosts["R1-CPE-BAT-A"])
    print(type(nr.inventory.hosts["R1-CPE-BAT-A"]))

```

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
R1-CPE-BAT-A
<class 'nornir.core.inventory.Host'>
```

La valeur du premier élément retourné est R1-CPE-BAT-A, et son type est nornir.core.inventory.Host, c'est-à-dire un objet représentant un hôte de l'inventaire dans Nornir.

**Question 1.9.16 :** Utilisez la méthode dir() sur l'objet de la question 15. Parmis les attributs affichés, lequel permet d'afficher l'adresse ip et le username / password du host en question? Affichez les valeurs de ces attributs dans la console. Depuis quel fichier ces données ont été récupérées ? Dans quelle section de ce fichier plus précisément.

```

def question_16(nr):
    print(dir(nr.inventory.hosts["R1-CPE-BAT-A"]))

```

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
['__bool__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__slots__', '__str__', '__subclasshook__', 'get_connection_options_recursively', 'has_parent_group_by_name', 'has_parent_group_by_object', 'close_connection', 'close_connections', 'connection_options', 'connections', 'data', 'defaults', 'dict', 'extended_data', 'extended_groups', 'get', 'get_connection', 'get_connection_parameters', 'groups', 'has_parent_group', 'hostname', 'items', 'keys', 'name', 'open_connection', 'password', 'platform', 'port', 'schema', 'username', 'values']
[TP_03] cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

Dans un inventaire Nornir, les credentials et l'adresse IP proviennent du fichier hosts.yaml

dir() est une fonction intégrée de Python qui permet d'afficher la liste complète des attributs et méthodes accessibles pour un objet donné.

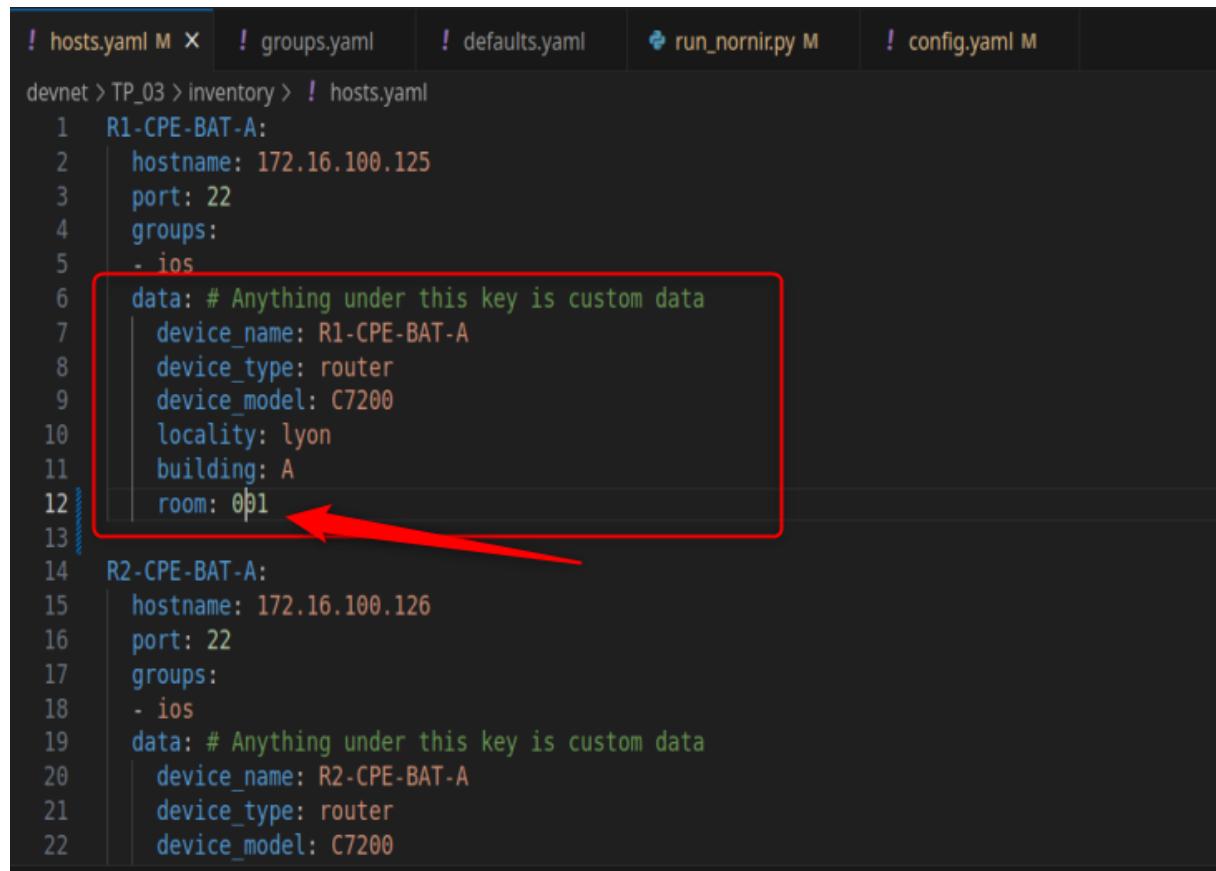
```

def question_16(nr):
    print(dir(nr.inventory.hosts["R1-CPE-BAT-A"]))
    first_host = list(nr.inventory.hosts.values())[0]
    print(f"Adresse IP (hostname): {first_host.hostname}")
    print(f"Username: {first_host.username}")
    print(f>Password: {first_host.password}")

```

Ces informations proviennent de plusieurs fichiers de l'inventaire Nornir : l'adresse IP de chaque hôte se trouve dans le fichier inventory/hosts, tandis que le username et le mot de passe sont définis dans inventory/defaults.yaml, et la configuration globale de l'inventaire est dans inventory/config.yaml.

**Question 1.9.17 :** Ajoutez une nouvelle entrée à la section de ce fichier, par exemple: room: 001 et exécuter de nouveau le script.



```
! hosts.yaml M X ! groups.yaml ! defaults.yaml ⚡ run_nornir.py M ! config.yaml M

devnet > TP_03 > inventory > ! hosts.yaml

1   R1-CPE-BAT-A:
2     hostname: 172.16.100.125
3     port: 22
4     groups:
5       - ios
6     data: # Anything under this key is custom data
7       device_name: R1-CPE-BAT-A
8       device_type: router
9       device_model: C7200
10      locality: lyon
11      building: A
12      room: 001
13
14 R2-CPE-BAT-A:
15   hostname: 172.16.100.126
16   port: 22
17   groups:
18     - ios
19   data: # Anything under this key is custom data
20     device_name: R2-CPE-BAT-A
21     device_type: router
22     device_model: C7200
```

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
['__bool__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__slots__', '__str__', '__subclasshook__', 'get_connection_options_recursively', 'has_parent_group_by_name', 'has_parent_group_by_object', 'close_connection', 'close_connections', 'connection_options', 'connections', 'data', 'defaults', 'dict', 'extended_data', 'extended_groups', 'get', 'get_connection', 'get_connection_parameters', 'groups', 'has_parent_group', 'hostname', 'items', 'keys', 'name', 'open_connection', 'password', 'platform', 'port', 'schema', 'username', 'values']
Adresse IP (hostname): 172.16.100.125
```

On constate ici que le champ room n'apparaît pas dans le résultat

**Question 1.9.18 :** Affichez la valeur de l'attribut "room" crée à la question 17 sur le terminal

Pour afficher sur le terminal la valeur de l'attribut *room* que vous avez créé à la question 17, il suffit d'utiliser l'instruction Python suivante, qui récupère cet attribut dans les données de l'hôte

```
def question_18(nr):
    print(nr.inventory.hosts["R1-CPE-BAT-A"].data["room"])
```

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
1
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

**Question 1.9.19 :** Nornir nous fournit également la possibilité d'accéder aux données du fichier groups.yaml à l'aide de l'attribut groups de l'objet inventory. Affichez les groupes définis dans le fichier groups.yaml à l'aide du code suivant

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
{'ios': Group: ios, 'data': Group: data}
```

Ou en fonction des tabulations dans notre fichiers groups.yaml

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
{'ios': Group: ios}
```

```
def question_19(nr):
    print(nr.inventory.groups)
```

Le fichier groups.yaml contient clairement deux groupes distincts : ios et data.

**Question 1.9.20 :** Il est possible d'afficher les groupes rattaché à un host à l'aide de l'attribut groups de l'objet

```
def question_20(nr):
    print(nr.inventory.hosts.get('R1-CPE-BAT-A').groups)
```

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
[Group: ios]
```

nr.inventory.hosts.get('R1-CPE-BAT-A') permet de récupérer l'objet Host correspondant au nom R1-CPE-BAT-A dans l'inventaire.

L'appel à .get() permet d'accéder à un élément du dictionnaire des hosts

Une fois que l'on dispose de cet objet Host, on peut accéder à son attribut .groups, qui contient la liste des groupes auxquels cet hôte appartient.

**Question 1.9.21 :** Par exemple, si on veut afficher les keys définis dans le fichier groups.yaml du host RP1-CPE-BAT-A

La commande `nr.inventory.hosts.get('R1-CPE-BAT-A').groups[0].keys()` renvoie `dict_keys(['vendor'])`. Cela signifie que le groupe associé à l'hôte R1-CPE-BAT-A dans le fichier `groups.yaml` contient une seule clé définie : vendor.

The screenshot shows a terminal window with the following content:

```
35 def question_21(nr):
36     | print(nr.inventory.hosts.get('R1-CPE-BAT-A').groups[0].keys())
37
38 def question_22(nr):
39     | pass
40
41 def question_23(nr):
42     | pass
43
44 . . . . .
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict_keys([])
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict_keys([])
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ ^C
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict_keys(['vendor'])
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

A red arrow points to the terminal output line `dict_keys(['vendor'])`, which is highlighted with a red box.

**Question 1.9.22 :** Affichez le vendor de l'host R1-CPE-BAT-A en partant du code de la question 21

The screenshot shows a terminal window with the following content:

```
37
38 def question_22(nr):
39     | print(nr.inventory.hosts.get('R1-CPE-BAT-A').groups[0].get('vendor'))
40
41 def question_23(nr):
42     | pass
43
44 def question_24(nr):
45     | pass
46
47 def question_25(nr):
48     | pass
49
50 def question_26(nr):
51     | pass
52
53 def question_27(nr):
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict_keys([])
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ ^C
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict_keys(['vendor'])
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
Cisco
```

La ligne `nr.inventory.hosts.get('R1-CPE-BAT-A').groups[0].get('vendor')` permet de récupérer l'objet hôte R1-CPE-BAT-A, d'accéder à son premier groupe associé, puis d'extraire et d'afficher la valeur du champ vendor (ici, "Cisco").

**Question 1.9.23 :** Affichez le hostname (adresse ip) de chaque host définis dans le fichier hosts.yaml en utilisant l'objet nr définis à la question 12.

The screenshot shows a code editor and a terminal window. The code editor contains Python functions for questions 23 through 26. The function for question 23 is highlighted with a red box around the loop body. The terminal window shows the command `python -m scripts.run_nornir` being run, followed by a list of IP addresses: 172.16.100.125, 172.16.100.126, 172.16.100.123, 172.16.100.189, 172.16.100.190, and 172.16.100.187. These last five IP addresses are also highlighted with a red box.

```
40
41 def question_23(nr):
42     for host_name in nr.inventory.hosts:
43         print(nr.inventory.hosts.get(host_name).hostname)
44
45 def question_24(nr):
46     pass
47
48 def question_25(nr):
49     pass
50
51 def question_26(nr):
52     pass
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
172.16.100.125
172.16.100.126
172.16.100.123
172.16.100.189
172.16.100.190
172.16.100.187
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

La fonction parcourt chaque hôte défini dans `nr.inventory.hosts` et affiche son hostname (adresse IP) en récupérant directement l'attribut depuis l'objet associé à chaque nom d'hôte.

**Question 1.9.24 :** Nornir nous donne la possibilité d'appliquer des filtres sur notre inventory pour récupérer par exemple un host à partir de son hostname par exemple. A l'aide du code suivant affichez la liste des hosts de type router

The screenshot shows a VS Code interface with two tabs open:

- run\_nornir.py**: Contains Python code for generating inventory hosts. A red box highlights the line `print(nr.filter(device\_type='router').inventory.hosts.keys())` (line 47). A red arrow points from this line to the corresponding host entry in the hosts.yaml file.
- hosts.yaml**: Shows the generated inventory configuration. It includes entries for R1-CPE-BAT-A and R2-CPE-BAT-A, each with its hostname, port, groups (ios), and device details (name, type, model).

Terminal output at the bottom shows the command `python3 -m scripts.run\_nornir` being run, followed by the output of `dict keys(['R1-CPE-BAT-A', 'R2-CPE-BAT-A', 'R1-CPE-BAT-B', 'R2-CPE-BAT-B'])`.

La fonction utilise la méthode `.filter(device_type='router')` pour sélectionner uniquement les hôtes dont le type est défini comme "router", puis affiche la liste des noms de ces hôtes avec `.inventory.hosts.keys()`.

**Question 1.9.25 :** Affichez à présent la liste des hosts de type router\_switch

The screenshot shows a terminal window with several sections of code and output.

**Python Code:**

```
4 def question_24(nr):
5     print(nr.filter(device_type='router').inventory.hosts.keys())
6
7 def question_25(nr):
8     print(nr.filter(device_type='router_switch').inventory.hosts.keys())
9
10 def question_26(nr):
11     pass
12
13 def question_27(nr):
14     result = nr.run(task=hello_world)
15     print(type(result))
16
17 def question_29(nr):
18     result = nr.run(task=hello_world)
19     print_result(result)
20
21 def question_30(nr):
22     pass
23
24 def question_32(nr):
25
```

**Configuration Dump:**

```
14 R2-CPE-BAT-A:
15     data: # Anything under this key
16
17 ESW1-CPE-BAT-A:
18     hostname: 172.16.100.123
19     port: 22
20     groups:
21         - ios
22     data: # Anything under this key
23         device_name: ESW1-CPE-BAT-A
24         device_type: router_switch
25         device_model: C3725
26         locality: lyon
27         building: A
28
29 R1-CPE-BAT-B:
30     hostname: 172.16.100.189
31     port: 22
32     groups:
33         - ios
34     data: # Anything under this key
35         device_name: R1-CPE-BAT-B
36
37
38
39
40
41
42
43
44
```

**Terminal Bottom:**

```
2.16.100.187
TP_03] cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
[nan_router,j2 R1-CPE-BAT-A', 'R2-CPE-BAT-A', 'R1-CPE-BAT-B', 'R2-CPE-BAT-B'])
[TP_03] cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
ict_keys(['R1-CPE-BAT-A', 'R2-CPE-BAT-A', 'R1-CPE-BAT-B', 'R2-CPE-BAT-B'])
[TP_03] cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
ict_keys(['ESW1-CPE-BAT-A', 'ESW1-CPE-BAT-B'])
[TP_03] cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

Je filtre maintenant le device\_type sur "router\_switch" pour afficher uniquement les switches présents dans l'inventaire.

**Question 1.9.26 :** Importez les classes Task et Result et définissez une première task nommée hello\_world

```
51 | def hello_world(task: Task) -> Result:
52 |     return Result(
53 |         host=task.host,
54 |         result=f"{task.host.name} says hello world!"
55 |     )
56 |
57 |
58 | def question_26(nr):
59 |     result = nr.run(task=hello_world)
60 |     print(result)
61 |
62 | def question_27(nr):
63 |     result = nr.run(task=hello_world)
64 |     print(type(result))
65 |
66 | def question_29(nr):
67 |     result = nr.run(task=hello_world)
68 |     print_result(result)
69 |
70 | def question_30(nr):
71 |     pass
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
python3-TP_03 + v ⌂ 會 … | [] x
```

```
from nornir import Task, Result
ImportError: cannot import name 'Task' from 'nornir' (/home/cpe/.local/share/virtualenvs/TP_03-khchFeCv/lib/python3.12/site-packages/nornir/_init_.py)
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
AggregatedResult (hello_world): {'R1-CPE-BAT-A': MultiResult: [Result: "hello_world"], 'R2-CPE-BAT-A': MultiResult: [Result: "hello_world"], 'ESW1-CPE-BAT-A': MultiResult: [Result: "hello_world"], 'R1-CPE-BAT-B': MultiResult: [Result: "hello_world"], 'R2-CPE-BAT-B': MultiResult: [Result: "hello_world"], 'ESW1-CPE-BAT-B': MultiResult: [Result: "hello_world"]}
```

Cette instruction permet d'afficher l'identifiant de la tâche en cours d'exécution, qui, dans ce cas précis, correspond à hello\_world."

**Question 1.9.27 :** Que retourne la variable result à la question 27 ? Aidez-vous de la méthode type()

```
61 |
62 | def question_27(nr):
63 |     result = nr.run(task=hello_world)
64 |     print(type(result))
65 |
66 | def question_29(nr):
67 |     result = nr.run(task=hello_world)
68 |     print_result(result)
69 |
70 | def question_30(nr):
71 |     pass
72 |
73 | def question_32(nr):
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
_.py
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
AggregatedResult (hello_world): {'R1-CPE-BAT-A': MultiResult: [Result: "hello_world"], 'R2-CPE-BAT-A': MultiResult: [Result: "hello_world"], 'ESW1-CPE-BAT-B': MultiResult: [Result: "hello_world"]}
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
<class 'nornir.core.task.AggregatedResult'>
```

La variable result est une instance de AggregatedResult qui stocke les résultats consolidés de l'exécution des tâches Nornir sur l'ensemble des hôtes ciblés.

**Question 1.9.28 :** Nornir fournit une fonction print\_result qui permet d'afficher des logs sur les événements retournés par la task. Utilisez print\_result pour afficher la variable result.

```
devnet > TP_03 > scripts > run_nornir.py
1   from nornir import InitNornir
2 | from nornir.core.task import Task, Result
3
```

**Question 1.9.29 :** Après avoir affiché result à l'aide de la fonction print\_result, qu'avez-vous remarqué ? Sur quel équipement la task s'est exécutée par défaut ?

```
00
67  def question_29(nr):
68      result = nr.run(task=hello_world)
69      print_result(result)
70
71  def question_30(nr):
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
hello_world*****
* ESW1-CPE-BAT-A ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
ESW1-CPE-BAT-A says hello world!
~~~~ END hello_world ~~~~~
* ESW1-CPE-BAT-B ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
ESW1-CPE-BAT-B says hello world!
~~~~ END hello_world ~~~~~
* R1-CPE-BAT-A ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
R1-CPE-BAT-A says hello world!
~~~~ END hello_world ~~~~~
* R1-CPE-BAT-B ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
R1-CPE-BAT-B says hello world!
~~~~ END hello_world ~~~~~
* R2-CPE-BAT-A ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
R2-CPE-BAT-A says hello world!
~~~~ END hello_world ~~~~~
```

La tâche hello\_world a été exécutée avec succès sur l'ensemble des équipements de l'inventaire, à savoir les commutateurs (ESW1-CPE-BAT-A/B) et les routeurs (R1-CPE-BAT-A/B, R2-CPE-BAT-A/B). Par défaut, Nornir applique la tâche à tous les devices en l'absence de filtre spécifique.

**Question 1.9.30 :** Faites en sorte que la task hello\_world s'exécute uniquement sur les équipements de type router\_switch.

The screenshot shows a terminal window with the following content:

```
70
71 def question_30(nr):
72     host = nr.filter(device_type='router_switch')
73     print(result(host.run(task=hello_world)))
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
```

TERMINAL

```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
hello_world*****
* ESW1-CPE-BAT-A ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
ESW1-CPE-BAT-A says hello world!
~~~~ END hello_world ~~~~~
* ESW1-CPE-BAT-B ** changed : False ****
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
ESW1-CPE-BAT-B says hello world!
~~~~ END hello_world ~~~~~
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$
```

La task hello\_world s'exécute uniquement sur les équipements de type router\_switch en filtrant les hôtes avec nr.filter(device\_type='router\_switch') avant d'appeler run.

**Question 1.9.31 :** Installez les packages nornir\_napalm et nornir\_netmiko à votre environnement de développement (TP03)

```
from nornir import InitNornir
from nornir.core.task import Task, Result
from nornir_utils.plugins.functions import print_result
from nornir_napalm.plugins.tasks import napalm_get, napalm_configure, napalm_cli
from nornir_netmiko.tasks import netmiko_send_config, netmiko_send_command, netmiko_save_config, netmiko_commit
```

**Question 1.9.32 :** Développez une task permettant d'afficher l'état des interfaces de chaque routeur à l'aide de la fonction napalm\_cli.

Cette tâche utilise napalm\_cli avec la commande "show ip interface brief" pour récupérer et résumé concis de l'état des interfaces IP (adresse, statut, up ?) sur chaque routeur ciblé par Nornir.

**Question 1.9.33 :** Développez une task permettant d'afficher la table arp de chaque switch référencé dans l'inventory. Utilisez la fonction napalm\_get pour cela.

```

88 def get_arp_table(task):
89     task.run(
90         task=napalm_get,
91         getters=["arp_table"]
92     )
93
94 def question_33(nr):
95     switches = nr.filter(device_type="router_switch")
96     result = switches.run(task=get_arp_table)
97     print_result(result)
98
99 def question_34(nr):

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```

get_arp_table*****
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
get_arp_table*****
* ESW1-CPE-BAT-A *** changed : False ****
vvvv get_arp_table ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_get ** changed : False ----- INFO
{ 'arp_table': [ { 'age': 0.0,
                  'interface': 'Vlan99',
                  'ip': '172.16.100.2',
                  'mac': '00:00:5E:00:01:63'},
                  { 'age': -1.0,
                  'interface': 'Vlan99',
                  'ip': '172.16.100.123',
                  'mac': 'C2:02:0F:B5:00:00'}]}
~~~~ END get_arp_table ~~~~~
* ESW1-CPE-BAT-B *** changed : False ****
vvvv get_arp_table ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO

```

La fonction utilise la tâche napalm\_get avec le getter "arp\_table" pour récupérer et afficher la table ARP de tous les équipements filtrés comme router\_switch dans l'inventaire Nornir

**Question 1.9.34 :** Développez deux tasks permettant de créer une interface de loopback sur les routeurs R1 et R2 du bâtiment A . (Une task par routeur) à l'aide de la task napalm\_configure

```
def configure_loopback_r1(task):
    task.run(
        task=napalm_configure,
        configuration=""""
interface Lo1
    ip address 1.1.1.1 255.255.255.255
    description Loopback pour R1-CPE-BAT-A
"""
    )

def configure_loopback_r2(task):
    task.run(
        task=napalm_configure,
        configuration=""""
interface Lo1
    ip address 2.2.2.2 255.255.255.255
    description Loopback pour R2-CPE-BAT-A
"""
    )
```

```

def question_34(nr):
    R1 = nr.filter(name='R1-CPE-BAT-A')
    result_R1 = R1.run(task=configure_loopback_r1)
    R2 = nr.filter(name='R2-CPE-BAT-A')
    result_R2 = R2.run(task=configure_loopback_r2)
    print_result(result_R1)
    print_result(result_R2)

```

```

(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
configure_loopback_r1*****
* R1-CPE-BAT-A ** changed : True ****
vvvv configure_loopback_r1 ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_configure ** changed : True ----- INFO
+interface Lo1
+ ip address 1.1.1.1 255.255.255.255
+ description Loopback pour R1-CPE-BAT-A
~~~~ END configure_loopback_r1 ~~~~~
configure_loopback_r2*****
* R2-CPE-BAT-A ** changed : True ****
vvvv configure_loopback_r2 ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_configure ** changed : True ----- INFO
+interface Lo1
+ ip address 2.2.2.2 255.255.255.255
+ description Loopback pour R2-CPE-BAT-A
~~~~ END configure_loopback_r2 ~~~~~

```

La fonction `configure_loopback_r1` et `configure_loopback_r2` configurent chacune une interface Loopback sur le routeur correspondant avec une adresse IP et une description spécifique.

**Question 1.9.35 :** Développez une fonction contenant une task permettant de sauvegarder la running-config sur les équipements (routeurs / switchs) depuis `napalm_cli`

```

127 def save_running_config(task):
128     # Exécute la commande de sauvegarde de la configuration
129     task.run(
130         task=napalm_cli,
131         commands=["wr"]
132     )
133
134 def question_35(nr):
135     result = nr.run(task=save_running_config)
136     print_result(result)
137
138 def question_36(nr):
139     pass

```

TERMINAL OUTPUT:

```

---- napalm_cli ** changed : False ----- INFO
{'wr': 'Building configuration...\\n[OK]'} ①
~~~~ END save_running_config ~~~~~
* R2-CPE-BAT-A ** changed : False ****
vvvv save_running_config ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_cli ** changed : False ----- INFO
{'wr': 'Building configuration...\\n[OK]'} ②
~~~~ END save_running_config ~~~~~
* R2-CPE-BAT-B ** changed : False ****
vvvv save_running_config ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvv INFO
---- napalm_cli ** changed : False ----- INFO
{'wr': 'Building configuration...\\n[OK]'} ③
~~~~ END save_running_config ~~~~~
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ 

```

La fonction `save_running_config` définit une task qui exécute la commande permettant de sauvegarder la configuration en cours (`running-config`) sur les équipements de type routeur ou switch via `napalm_cli`.

**Question 1.9.36 :** Développez une fonction permettant d'afficher l'état des interfaces de chaque routeur à l'aide de la task netmiko\_send\_command.

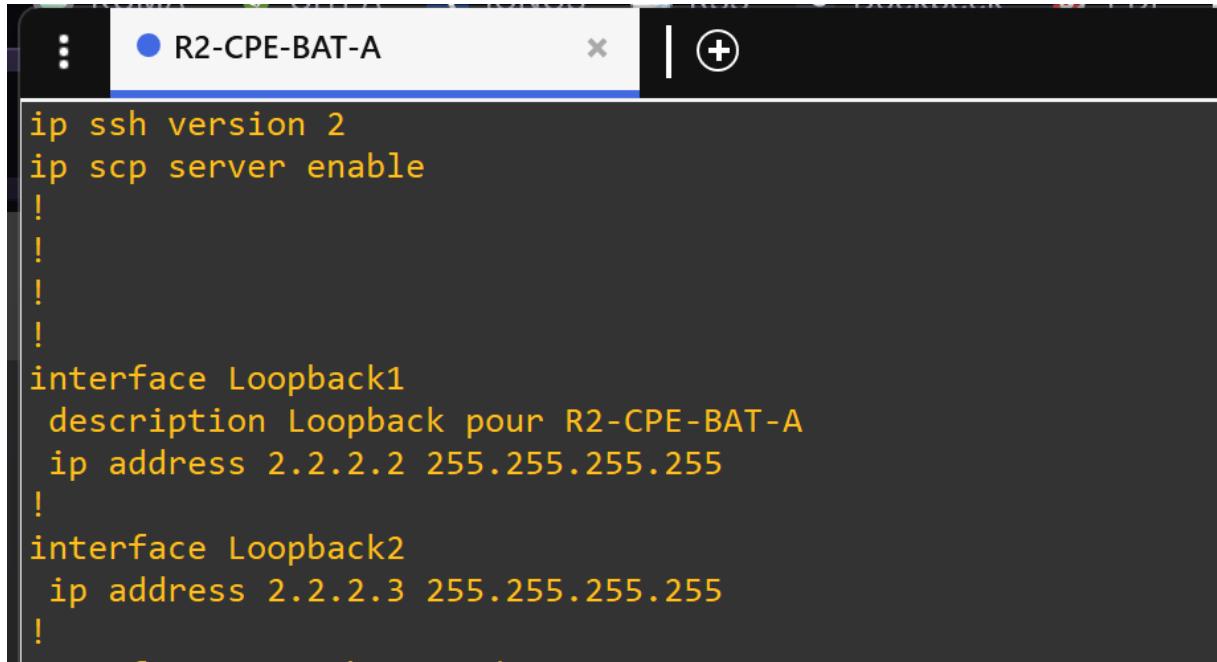
La fonction exécute la commande show ip interface brief uniquement sur les équipements de type routeur et affiche l'état de leurs interfaces.

**Question 1.9.37 :** Développez une fonction permettant de créer une interface de loopback 2 sur les routeurs R1 et R2 du bâtiment A . (Une task par routeur) en utilisant la task netmiko send config

```
def configure_loopback2_r1(task):
    config_commands = [
        "interface Lo2",
        "ip address 1.1.1.2 255.255.255.255",
        "description Loopback2 pour R1-CPE-BAT-A"
    ]
    task.run(task=netmiko_send_config, config_commands=config_commands)

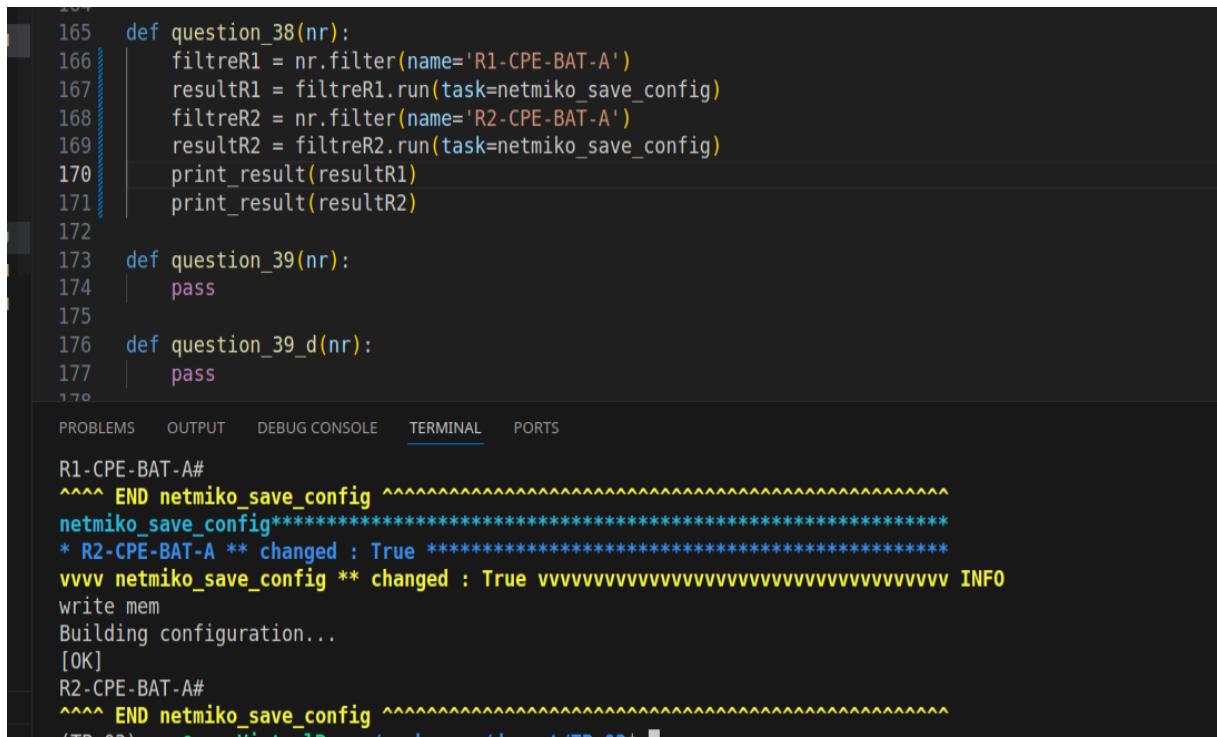
def configure_loopback2_r2(task):
    config_commands = [
        "interface Lo2",
        "ip address 2.2.2.3 255.255.255.255",
        "description Loopback2 pour R2-CPE-BAT-A"
    ]
    task.run(task=netmiko_send_config, config_commands=config_commands)
```

La fonction question\_37 configure une interface Loopback 2 avec une adresse IP et une description spécifique sur les routeurs R1 et R2 du bâtiment A en exécutant des tasks Netmiko séparées pour chaque routeur.



```
ip ssh version 2
ip scp server enable
!
!
!
!
interface Loopback1
description Loopback pour R2-CPE-BAT-A
ip address 2.2.2.2 255.255.255.255
!
interface Loopback2
ip address 2.2.2.3 255.255.255.255
!
```

**Question 1.9.38 :** Développez une fonction contenant une task permettant de sauvegarder la running-config de la question précédente à l'aide de la task netmiko\_save\_config



```
165 def question_38(nr):
166     filtreR1 = nr.filter(name='R1-CPE-BAT-A')
167     resultR1 = filtreR1.run(task=netmiko_save_config)
168     filtreR2 = nr.filter(name='R2-CPE-BAT-A')
169     resultR2 = filtreR2.run(task=netmiko_save_config)
170     print_result(resultR1)
171     print_result(resultR2)
172
173 def question_39(nr):
174     pass
175
176 def question_39_d(nr):
177     pass
178
```

R1-CPE-BAT-A#  
~~~~ END netmiko\_save\_config ~~~~~  
netmiko\_save\_config\*\*\*\*\*  
\* R2-CPE-BAT-A \*\* changed : True \*\*\*\*\*  
vvv netmiko\_save\_config \*\* changed : True vvvvvvvvvvvvvvvvvvvvvv INFO  
write mem  
Building configuration...  
[OK]  
R2-CPE-BAT-A#  
~~~~ END netmiko\_save\_config ~~~~~

La fonction sauvegarde la configuration en cours des routeurs R1-CPE-BAT-A et R2-CPE-BAT-A dans la startup config et affiche le résultat.

**Question 1.9.39 :** Reprenez la première partie du TP afin de déployer les configurations générées sur les équipements du bâtiment A et B (vlan, routage inter-vlan, vrrp pour les vlans 10 et 20. Vous avez le choix d'utiliser nornir\_netmiko ou nornir\_napalm (ou les deux) pour le déploiement de la configuration. Pensez à sauvegarder automatiquement (depuis nornir) vos configurations après le déploiement. Aidez-vous de la doc de nornir\_napalm et nornir\_netmiko pour le déploiement de config via un fichier de conf.

```
def deploy_config_from_file(task: Task, config_file: str) -> Result:
    """Déploie la configuration depuis un fichier sur un équipement via Nornir/Netmiko."""
    with open(config_file, "r") as f:
        commands = f.read().splitlines()
    result = task.run(task=netmiko_send_config, config_commands=commands)
    print_result(result)
    result = task.run(task=netmiko_save_config)
    print_result(result)
    return result
```

```
def deploy_vrrp(task):
    vrrp_filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + "_VRRP.conf"
    task.run(
        task=deploy_config_from_file,
        config_file=f"config/{vrrp_filename}",
        name=f"Déploiement VRRP {task.host.name}"
    )

def deploy_to_hosts(nr, host_patterns):
    """Déploie les configurations pour tous les hôtes et ajoute le fichier VRRP pour les routeurs"""
    for pattern in host_patterns:
        # Filtrer uniquement par nom pour tous
        filtered_hosts = nr.filter(name=pattern)
        if not filtered_hosts.inventory.hosts:
            print(f"Aucun hôte correspondant à '{pattern}'")
            continue

        # Déploiement des configs principales
        def deploy_main(task):
            filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + ".conf"
            task.run(
                task=deploy_config_from_file,
                config_file=f"config/{filename}",
                name=f"Déploiement config {task.host.name}"
            )

        filtered_hosts.run(task=deploy_main)
```

```

def question_39(nr):
    host_patterns = [
        'R1-CPE-BAT-A',
        'R2-CPE-BAT-A',
        'R1-CPE-BAT-B',
        'R2-CPE-BAT-B',
        'ESW1-CPE-BAT-A',
        'ESW1-CPE-BAT-BS'
    ]
    deploy_to_hosts(nr, host_patterns)

```

Ce code déploie les fichiers de configuration sur plusieurs équipements réseau dont les noms correspondent aux motifs spécifiés (R1-CPE-BAT-A, R2-CPE-BAT-A, etc.) en utilisant Nornir et Netmiko. Pour chaque équipement trouvé, il lit le fichier de configuration correspondant, applique les commandes et sauvegarde la configuration.

Le ping fonctionne correctement entre les machines du bâtiment A et du bâtiment B

```

Bad command:  CIS . OSC . For help.

PC1> ping 172.16.20.1
84 bytes from 172.16.20.1 icmp_seq=1 ttl=63 time=19.975 ms
84 bytes from 172.16.20.1 icmp_seq=2 ttl=63 time=17.354 ms
84 bytes from 172.16.20.1 icmp_seq=3 ttl=63 time=14.289 ms
84 bytes from 172.16.20.1 icmp_seq=4 ttl=63 time=20.970 ms
84 bytes from 172.16.20.1 icmp_seq=5 ttl=63 time=15.281 ms

PC1>

PC4> ping 172.16.30.1
84 bytes from 172.16.30.1 icmp_seq=1 ttl=63 time=14.893 ms
84 bytes from 172.16.30.1 icmp_seq=2 ttl=63 time=18.324 ms
84 bytes from 172.16.30.1 icmp_seq=3 ttl=63 time=14.846 ms
84 bytes from 172.16.30.1 icmp_seq=4 ttl=63 time=20.380 ms
84 bytes from 172.16.30.1 icmp_seq=5 ttl=63 time=17.644 ms

```

**Question 1.9.39.b :** Vérifier l'état du HA entre les routeurs de chaque bâtiment : show vrrp brief. Le routeur R1 de chaque bâtiment doit être le backup et R2 doit être le master

```
R1-CPE-BAT-A#
R1-CPE-BAT-A#show vrrp
GigabitEthernet2/0.99 - Group 99
  State is Backup
  Virtual IP address is 172.16.100.124
  Virtual MAC address is 0000.5e00.0163
  Advertisement interval is 1.000 sec
  Preemption enabled
  Priority is 100
  Master Router is 172.16.100.126, priority is 110
  Master Advertisement interval is 1.000 sec
  Master Down interval is 3.609 sec (expires in 2.925 sec)
```

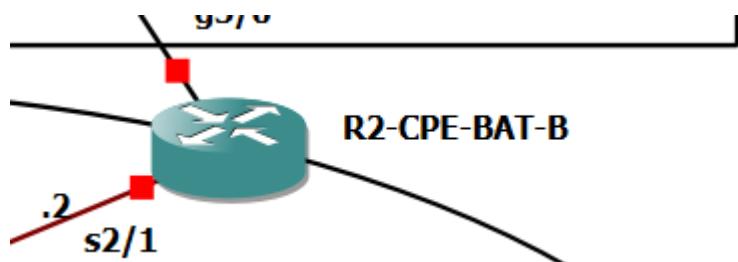
```
Nov 19 18:48:07.520: %S13->CONFIG_1: Configured from console by console
R4#show vrrp
GigabitEthernet3/0.99 - Group 99
  State is Master
  Virtual IP address is 172.16.100.188
  Virtual MAC address is 0000.5e00.0163
  Advertisement interval is 1.000 sec
  Preemption enabled
  Priority is 110
  Master Router is 172.16.100.190 (local), priority is 110
  Master Advertisement interval is 1.000 sec
  Master Down interval is 3.570 sec

GigabitEthernet3/0.10 - Group 10
  State is Master
  Virtual IP address is 172.16.30.252
  Virtual MAC address is 0000.5e00.010a
  Advertisement interval is 1.000 sec
  Preemption enabled
  Priority is 110
  Master Router is 172.16.30.254 (local), priority is 110
  Master Advertisement interval is 1.000 sec
  Master Down interval is 3.570 sec

GigabitEthernet3/0.20 - Group 20
```

### Question 1.9.39.c : Testez manuellement votre HA vrrp sur chaque bâtiment

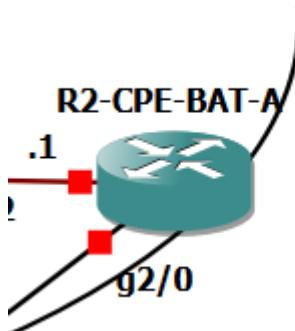
Si j'éteins le routeur R2-CPE-BAT-B



Lorsque je ping le depuis PC4 PC3 le ping fonctionne :

```
PC4> ping 172.16.30.1
84 bytes from 172.16.30.1 icmp_seq=1 ttl=63 time=29.856 ms
84 bytes from 172.16.30.1 icmp_seq=2 ttl=63 time=16.386 ms
84 bytes from 172.16.30.1 icmp_seq=3 ttl=63 time=15.784 ms
84 bytes from 172.16.30.1 icmp_seq=4 ttl=63 time=13.222 ms
```

Du coté du batiment A j'éteins R2-CPE-BAT-A



```
PC1> ping 172.16.20.1
84 bytes from 172.16.20.1 icmp_seq=1 ttl=63 time=52.823 ms
84 bytes from 172.16.20.1 icmp_seq=2 ttl=63 time=18.733 ms
84 bytes from 172.16.20.1 icmp_seq=3 ttl=63 time=13.897 ms
84 bytes from 172.16.20.1 icmp_seq=4 ttl=63 time=21.503 ms
```

Le ping fonctionne

**Question 1.9.39.d :** Testez automatiquement (via nornir) votre HA vrrp sur chaque bâtiment

```
def deploy_config_from_file(task: Task, config_file: str) -> Result:
    """Déploie la configuration depuis un fichier sur un équipement via Nornir/Netmiko."""
    with open(config_file, "r") as f:
        commands = f.read().splitlines()
    result = task.run(task=netmiko_send_config, config_commands=commands)
    print_result(result)
    result = task.run(task=netmiko_save_config)
    print_result(result)
    return result

def deploy_vrrp(task):
    vrrp_filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + "_VRRP.conf"
    task.run(
        task=deploy_config_from_file,
        config_file=f"config/{vrrp_filename}",
        name=f"Déploiement VRRP {task.host.name}"
    )
```

```

def deploy_to_hosts(nr, host_patterns):
    """Déploie les configurations pour tous les hôtes et ajoute le fichier VRRP pour les routers."""

    for pattern in host_patterns:
        # Filtrer uniquement par nom pour tous
        filtered_hosts = nr.filter(name=pattern)
        if not filtered_hosts.inventory.hosts:
            print(f"Aucun hôte correspondant à '{pattern}'")
            continue

        # Déploiement des configs principales
        def deploy_main(task):
            filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + ".conf"
            task.run(
                task=deploy_config_from_file,
                config_file=f"config/{filename}",
                name=f"Déploiement config {task.host.name}"
            )

        filtered_hosts.run(task=deploy_main)

    # Déploiement VRRP uniquement pour les routers
    routers = nr.filter(device_type="router")
    routers.run(task=deploy_vrrp)

```

```

def question_39(nr):
    host_patterns = [
        'R1-CPE-BAT-A',
        'R2-CPE-BAT-A',
        'R1-CPE-BAT-B',
        'R2-CPE-BAT-B',
        'ESW1-CPE-BAT-A',
        'ESW1-CPE-BAT-BS'
    ]
    deploy_to_hosts(nr, host_patterns)

```

**Question 1.9.40 :** Créez une task à l'aide de nornir\_netmiko ou nornir\_napalm permettant d'annoncer les subnets des vlans 10 et 20 du bâtiment A et B sur le backbone OSPF. Les vlans de chaque bâtiment pourront ainsi communiquer ensemble.. Assurez-vous de générer automatiquement la configuration OSPF à l'aide de Jinja2 (Voir TP02).

J'ai commencé par reprendre le template Jinja du TP02 pour produire automatiquement mes configurations OSPF dans des fichiers distincts.

```

R2_CPE_LYON_BAT_B_VRRP.conf   vlan_router.j2 M   config_ospf.j2 X   create_config.py   run_nornir.py M
devnet > TP-02 > templates > config_ospf.j2
1  router ospf 1
2  | router-id {{ id_router }}
3  {% for network in networks %}
4  | network {{ network }} area 0
5  {% endfor %}
6  end

```

Je rajoute dans mes fonctions de création de configuration la partie OSPF

```

def create_config_cpe_lyon.batA():
    ESW1_CPE_LYON_BAT_A_data= load_json_data_from_file(file_path='data/ESW1_CPE_LYON_BAT_A.json')
    ESW1_CPE_LYON_BAT_A_config = render_network_config(template_name='vlan_switch.j2', data=ESW1_CPE_LYON_BAT_A_data)

    R2_LYON_BAT_A_data = load_json_data_from_file(file_path='data/R2_CPE_LYON_BAT_A.json')
    R2_LYON_BAT_A_config = render_network_config(template_name='vlan_router.j2', data=R2_LYON_BAT_A_data)

    R1_LYON_BAT_A_data = load_json_data_from_file(file_path='data/R1_CPE_LYON_BAT_A.json')
    R1_LYON_BAT_A_config = render_network_config(template_name='vlan_router.j2', data=R1_LYON_BAT_A_data)

    R2_LYON_BAT_A_config_vrrp = render_network_config(template_name='vrrp_router.j2', data=R2_LYON_BAT_A_data)
    R1_LYON_BAT_A_config_vrrp = render_network_config(template_name='vrrp_router.j2', data=R1_LYON_BAT_A_data)

    R2_LYON_BAT_A_config_ospf_data = load_json_data_from_file(file_path='data/R2_CPE_LYON_BAT_A OSPF.json')
    R2_LYON_BAT_A_config_ospf = render_network_config(template_name='config_ospf.j2', data=R2_LYON_BAT_A_config_ospf_data)
    R1_LYON_BAT_A_config_ospf_data = load_json_data_from_file(file_path='data/R1_CPE_LYON_BAT_A OSPF.json')
    R1_LYON_BAT_A_config_ospf = render_network_config(template_name='config_ospf.j2', data=R1_LYON_BAT_A_config_ospf_data)

```

Dans mon script de déploiement je rajoute la partie déploiement de conf OSPF

```

def deploy_ospf(task):
    vrrp_filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + "_OSPF.conf"
    task.run(
        task=deploy_config_from_file,
        config_file=f"config/{vrrp_filename}",
        name=f"Déploiement VRRP {task.host.name}"
    )

```

```

def deploy_to_hosts(nr, host_patterns):
    """Déploie les configurations pour tous les hôtes et ajoute le fichier VRRP pour les routers."""

    for pattern in host_patterns:
        # Filtrer uniquement par nom pour tous
        filtered_hosts = nr.filter(name=pattern)
        if not filtered_hosts.inventory.hosts:
            print(f"Aucun hôte correspondant à '{pattern}'")
            continue

        # Déploiement des configs principales
        def deploy_main(task):
            filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + ".conf"
            task.run(
                task=deploy_config_from_file,
                config_file=f"config/{filename}",
                name=f"Déploiement config {task.host.name}"
            )

        filtered_hosts.run(task=deploy_main)

    # Déploiement VRRP uniquement pour les routers
    routers = nr.filter(device_type="router")
    routers.run(task=deploy_vrrp)
    routers.run(task=deploy_ospf)

```

Sur mon routeur cela à bien récupérer la configuration

```

router ospf 1
router-id 4.4.4.4
log-adjacency-changes
passive-interface Serial1/0
passive-interface GigabitEthernet3/0.99
network 10.1.5.0 0.0.0.3 area 0
network 172.16.30.0 0.0.0.255 area 0
network 172.16.40.0 0.0.0.255 area 0
network 172.16.100.64 0.0.0.63 area 0
network 172.16.100.128 0.0.0.63 area 0
!

```

Pour tester je prends PC1 et j'essaye de ping vers le Réseau de PC4

```

PC1> ping 172.16.40.1
84 bytes from 172.16.40.1 icmp_seq=1 ttl=60 time=83.318 ms
84 bytes from 172.16.40.1 icmp_seq=2 ttl=60 time=78.165 ms
84 bytes from 172.16.40.1 icmp_seq=3 ttl=60 time=67.675 ms
84 bytes from 172.16.40.1 icmp_seq=4 ttl=60 time=82.686 ms

```

PING OK