14/11/2025

TP 03 Module Devnet

Automatisation Réseau

Ronan FOURNEUVE
CPE LYON

Table des matières

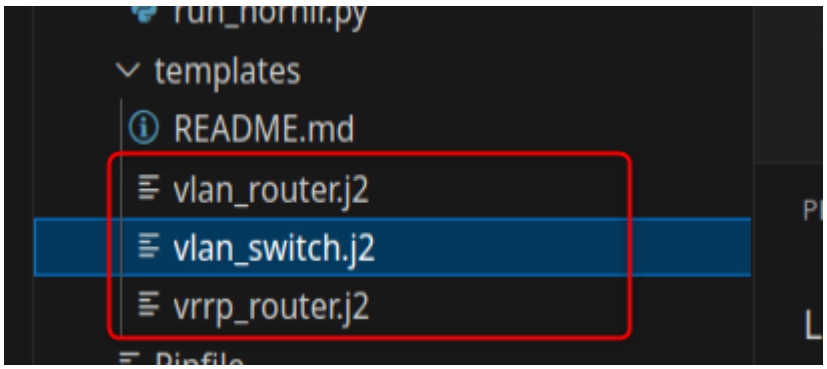
Partie 1 Générer et déployer des configurations réseau avec Python et Nornir

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)

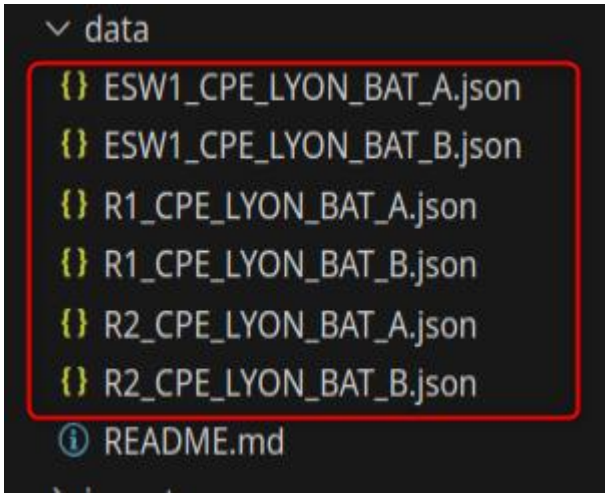
Question 1.8.1 : Développez l'ensemble des templates Jinja2 qui vous permettront de générer la configuration nécessaire pour le bon fonctionnement du bâtiment A. (Pour rappel, le vlan d'admin (99) est déjà configuré sur les équipements.) Les templates JInja2 doivent être sauvegardés dans le dossier templates.

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*.



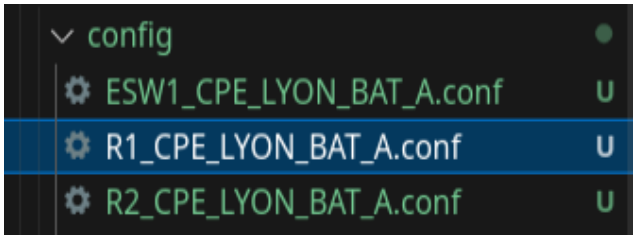
Question 1.8.3 : Développez les fonctions vous permettant de générer automatiquement les configurations à partir du template Jinja2 et de la structure de données que vous avez définis à la question 1 et 2. Écrivez votre code dans le fichier scripts/create_config.py.

```python
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_config)

    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).

```python
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
    }
```

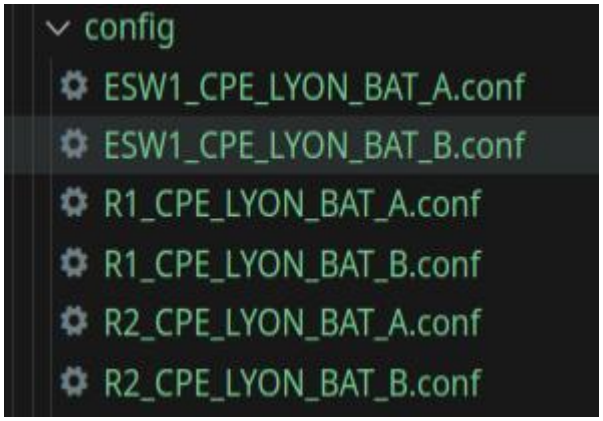
```python
#question 3:
config = create_config_cpe_lyon_batA()

#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'))
```



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

```python
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
    }
```

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



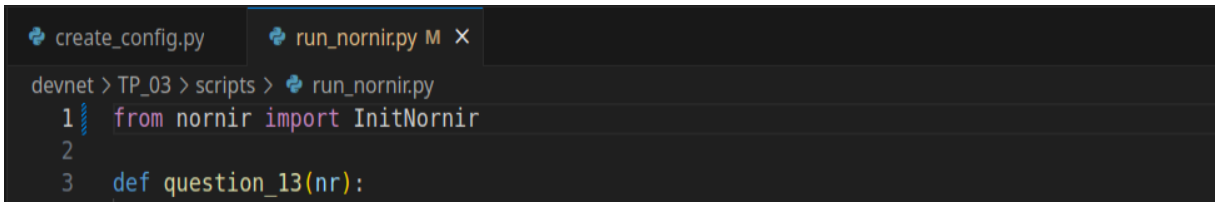
1.9 -Automatisation réseau avec Nornir

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

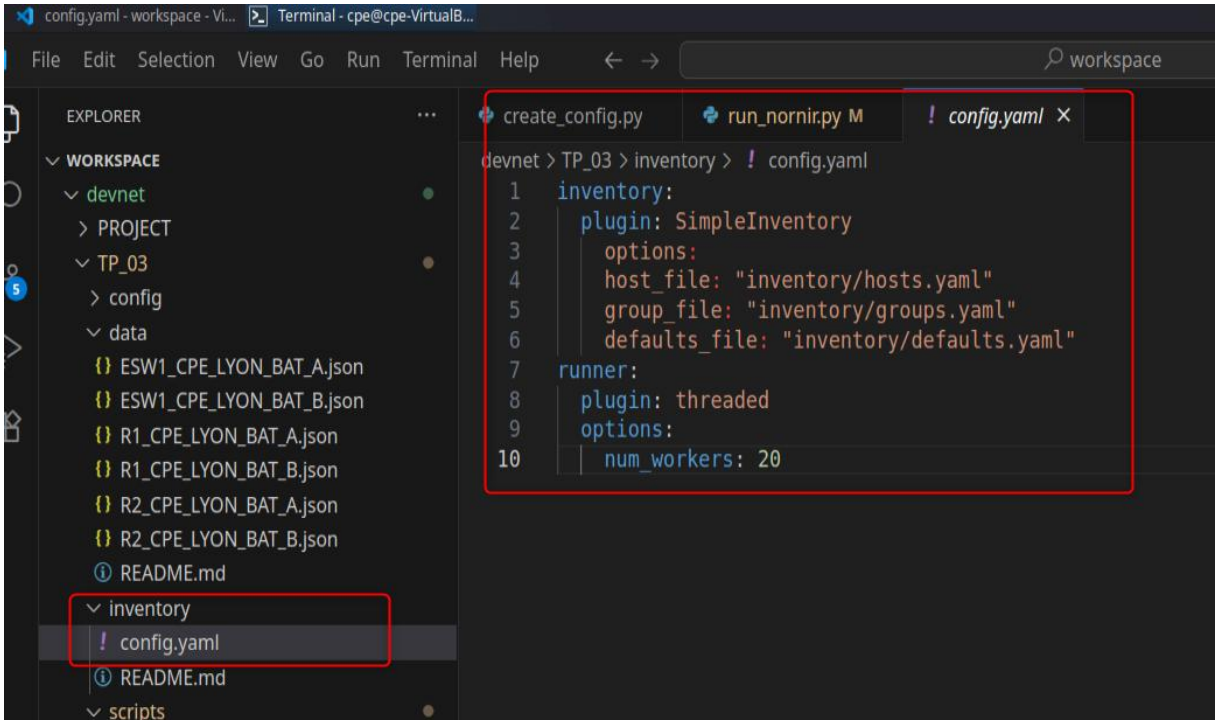
Nous installons le module Python Normir dans l'environnement virtuel géré par Pipenv

```
pipenv install nornir
```

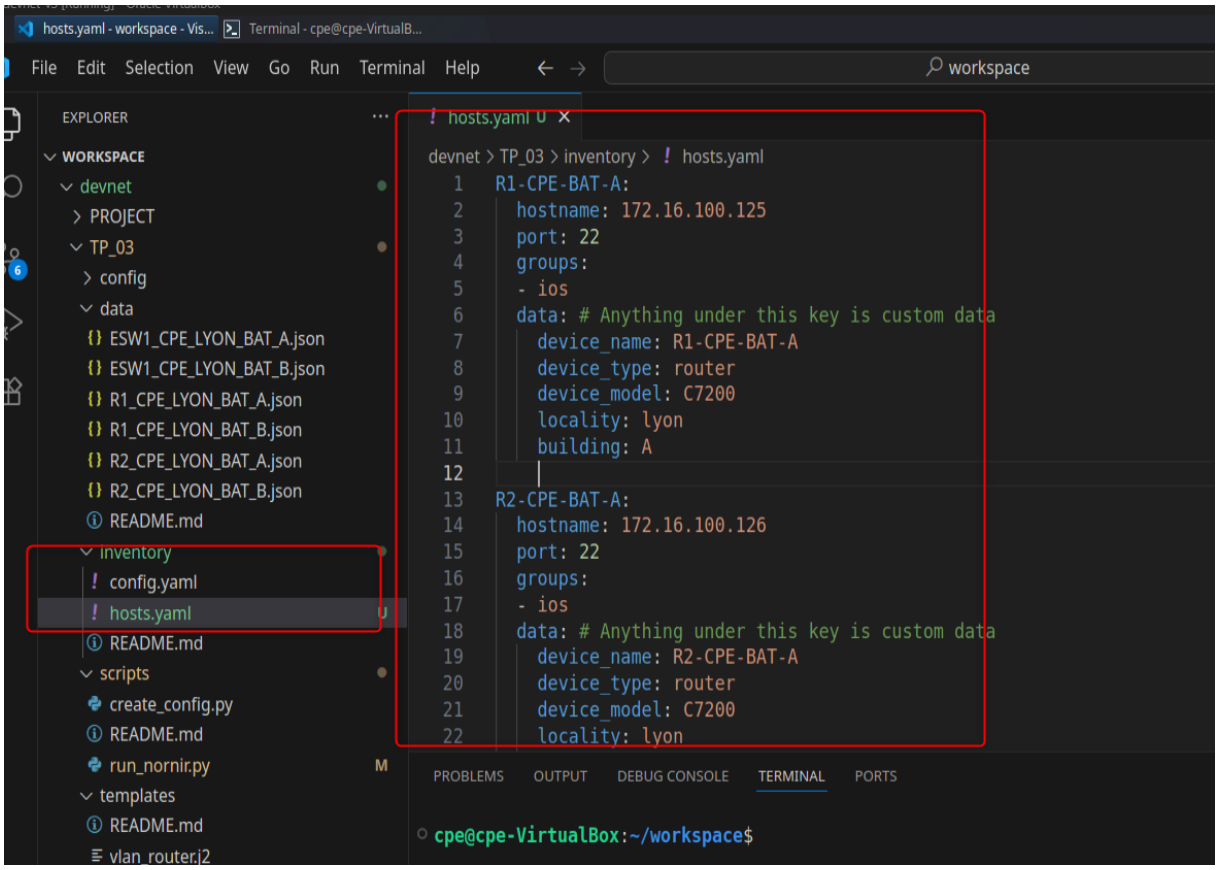
Question 1.9.7 : Créez un fichier nommé run_nornir.py dans le dossier scripts de votre workspace (TP03) et importez-le package nornir en en-tête du fichier



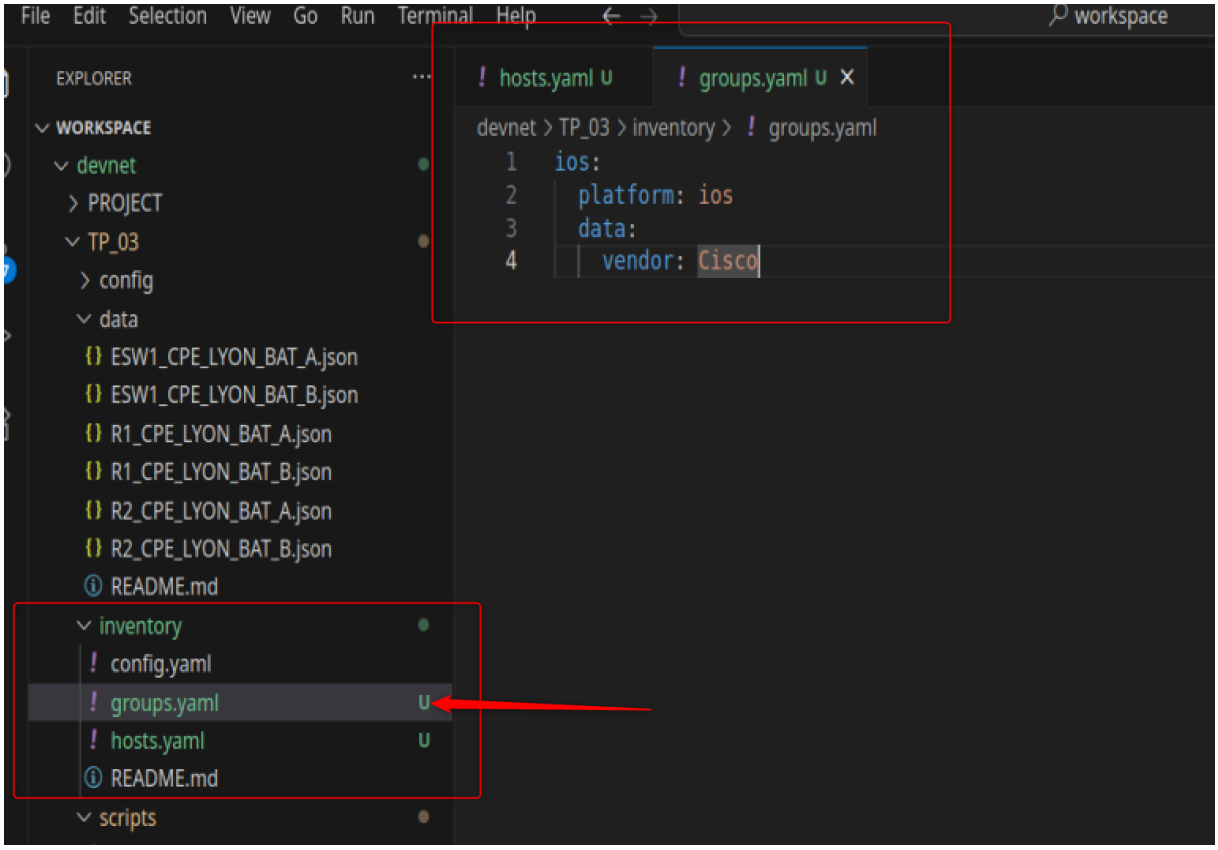
Question 1.9.8 : Créez un fichier config.yaml dans le dossier inventory de votre workspace et ajoutez la config suivante



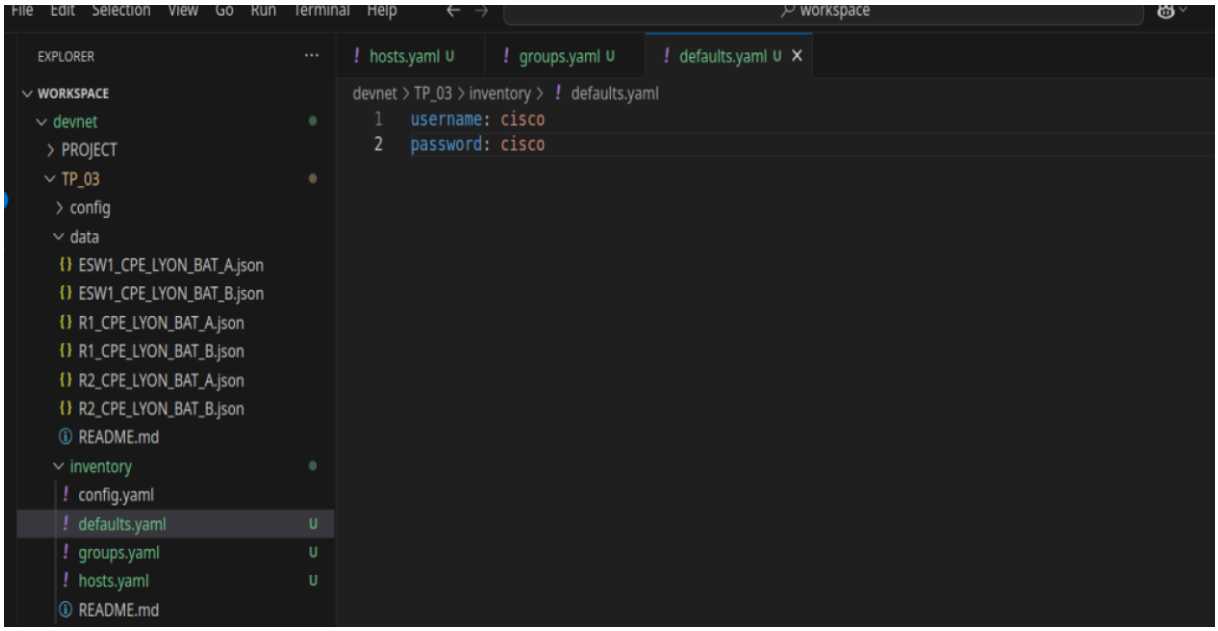
Question 1.9.9 : Créez le fichier hosts.yaml dans le dossier inventory et ajoutez la structure de données suivante :



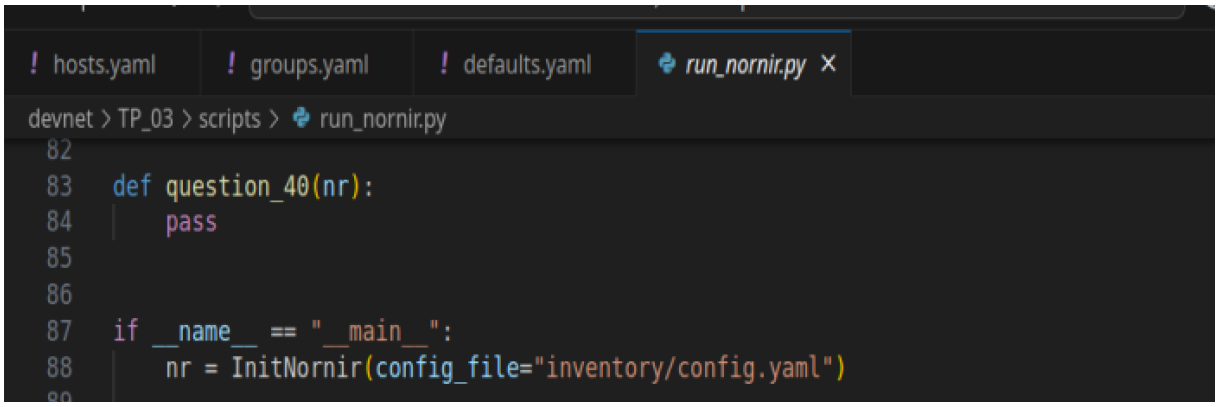
Question 1.9.10 : Créez le fichier groups.yaml dans le dossier inventory et ajoutez la structure de données suivante.



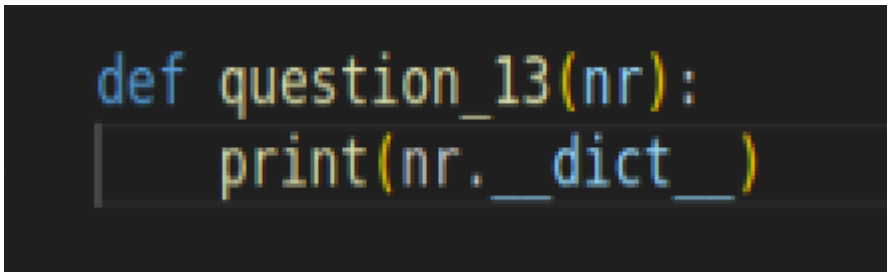
Question 1.9.11 : Créez le fichier defaults.yaml dans le dossier inventory et ajoutez la structure de données suivante.

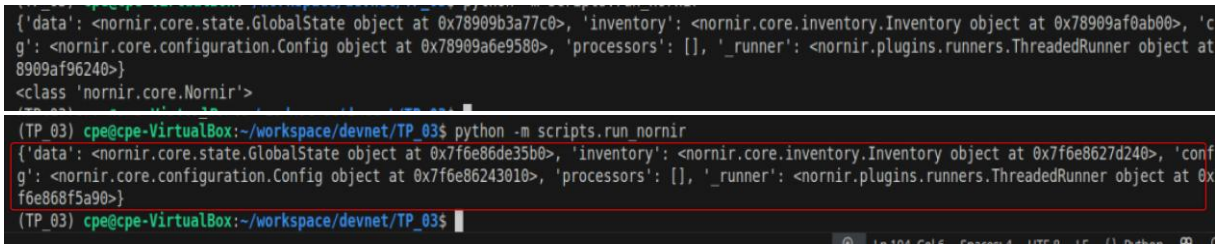


Question 1.9.12 : Vous pouvez à présent initialiser Nornir dans le __main__ du script run_nornir.py de la manière suivante



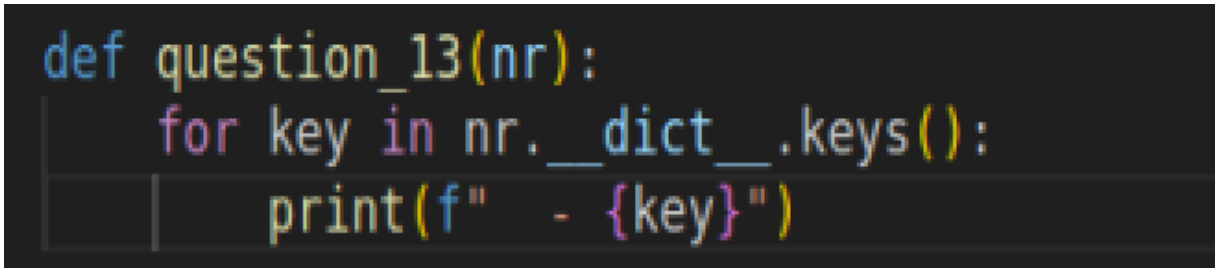
Question 1.9.13 : Quels sont les attributs de l'objet nr ? Pour connaître les attributs de l'objet nr utilisez l'attribut __dict__ (utilisable sur tout objet python). Quel est le format de données de sortie ? Quelles sont les attributs de l'objet nr ? A votre avis lequel nous permettrait d'aller lire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





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



```
<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.

```python
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())?

```python
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.

```python
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
ecursively', '_has_parent_group_by_name', '_has_parent_group_by_object', 'close_connection', 'close_connections', 'connection_options', 'connect
ns', 'data', 'defaults', 'dict', 'extended_data', 'extended_groups', 'get', 'get_connection', 'get_connection_parameters', 'groups', 'has_parent
roup', 'hostname', 'items', 'keys', 'name', 'open_connection', 'password', 'platform', 'port', 'schema', 'username', 'values']
```

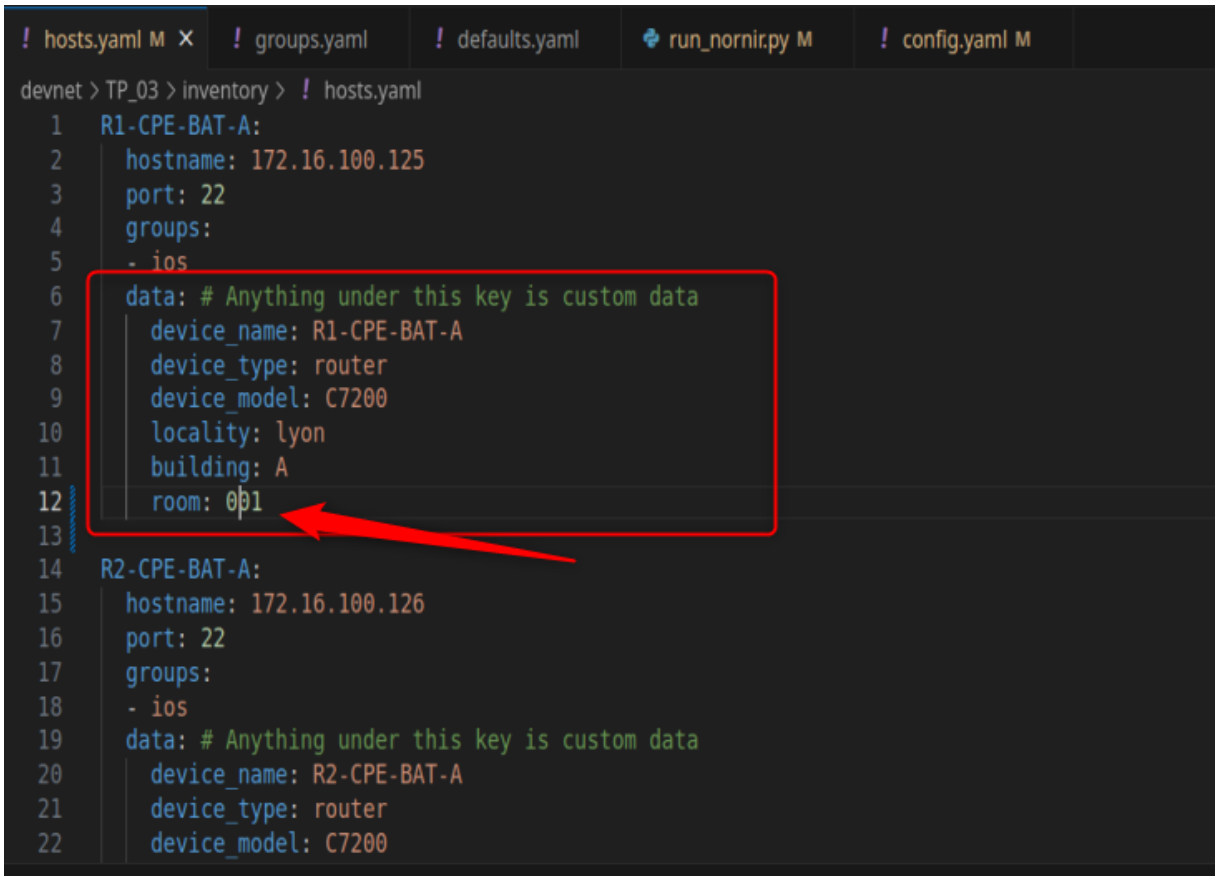
Dans un inventaire Nornir, les credentials et l'adresse IP proviennent du fichier hosts.yaml

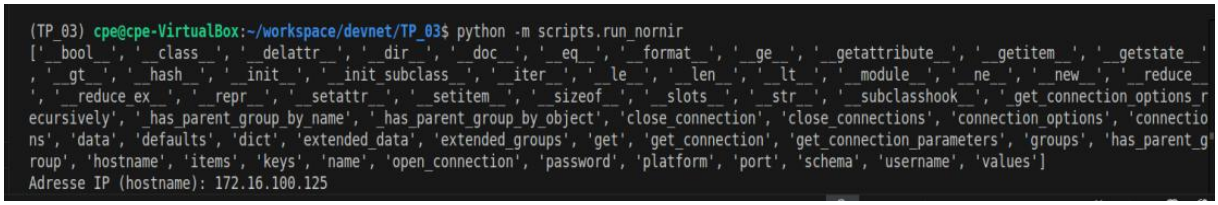
dir() est une fonction intégrée de Python qui permet d'afficher la liste complète des attributs et méthodes accessibles pour un objet donné.

```python
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.





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

```python
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}
```

```python
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

```python
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]
```

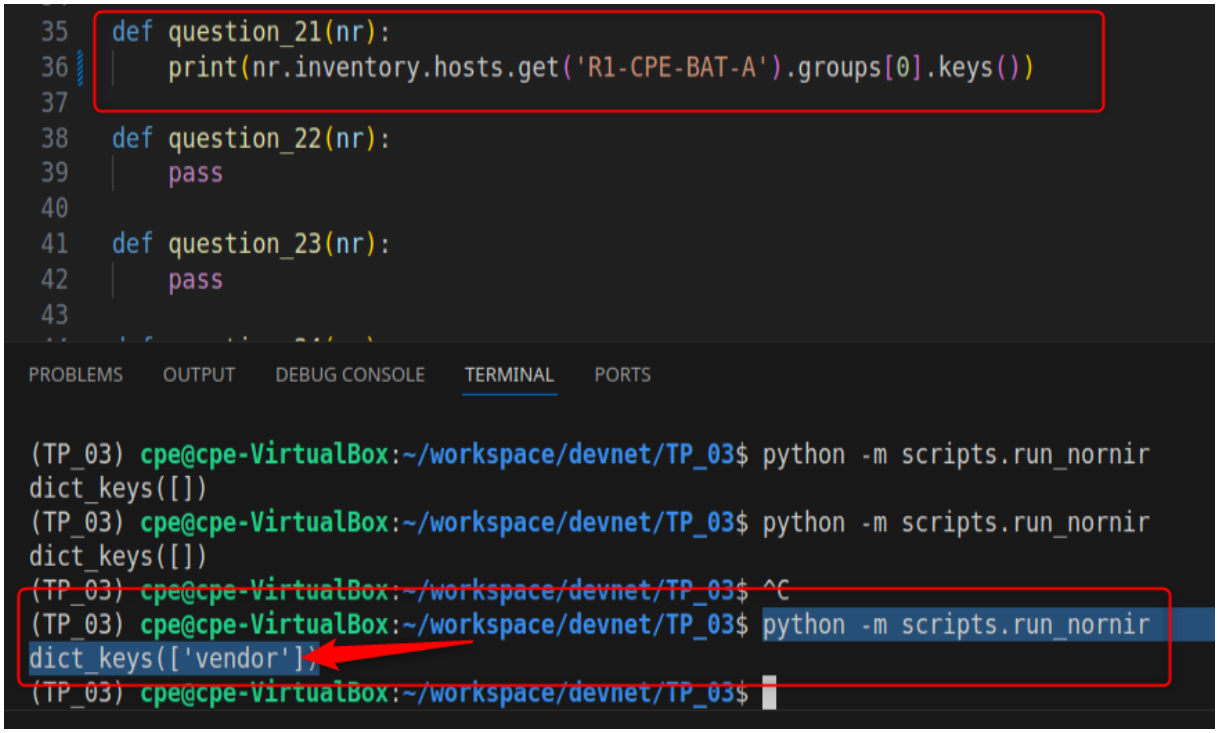
nr.inventory.hosts.get('R1-CPE-BAT-A') permet de récupérer l'objet Host correspondant au nom R1-CPE-BAT-A dans l'inventaire.

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

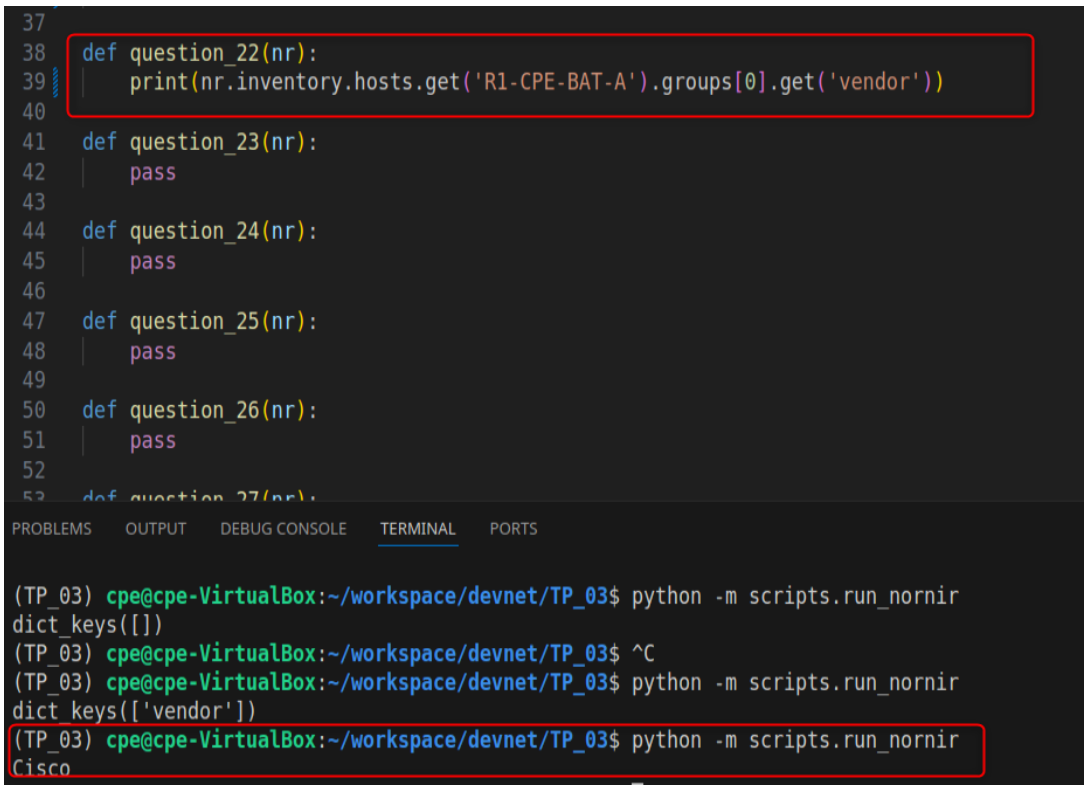
Une fois que l'on dispose de cet objet Host, on peut accéder à son attribut .groups, qui contient la liste des groupes auxquels cet hôte appartient.

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('RP1-CPE-BAT-A').groups[0].keys() renvoie dict_keys(['vendor']). Cela signifie que le groupe associé à l'hôte RP1-CPE-BAT-A dans le fichier groups.yaml contient une seule clé définie : vendor.

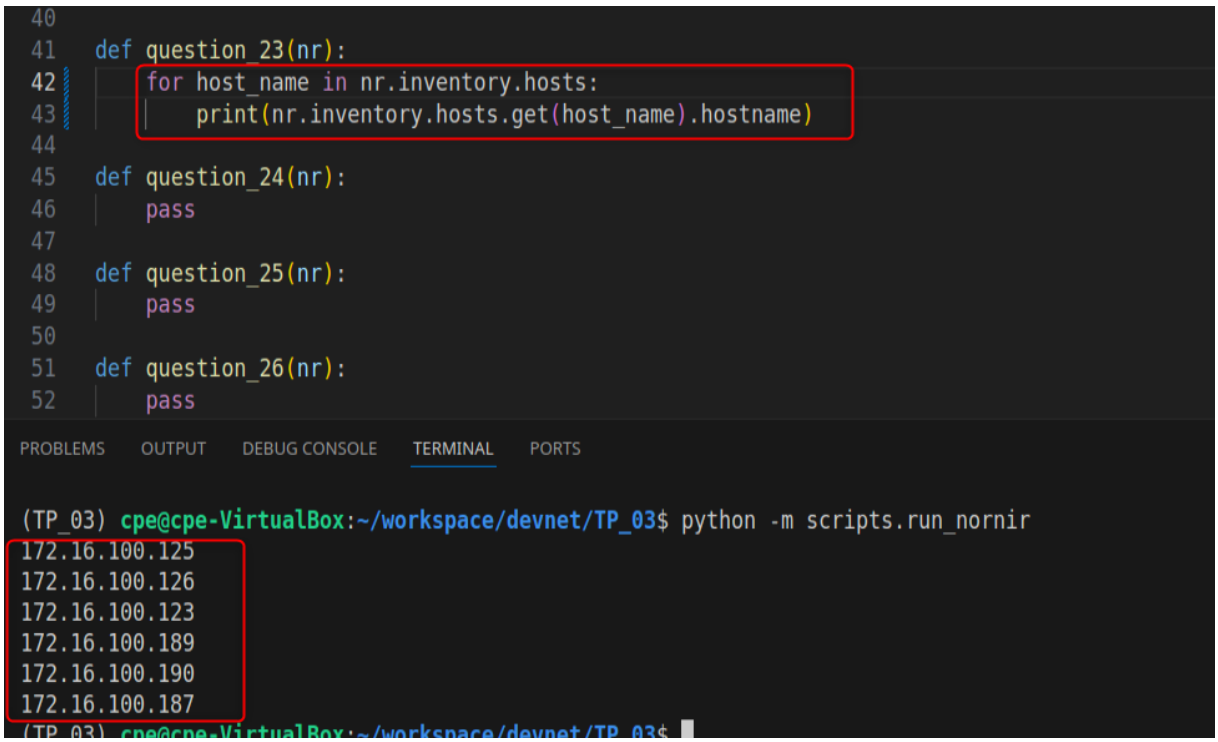


<u>Question 1.9.22 :</u> Affichez le vendor de l'host R1-CPE-BAT-A en partant du code de la question 21



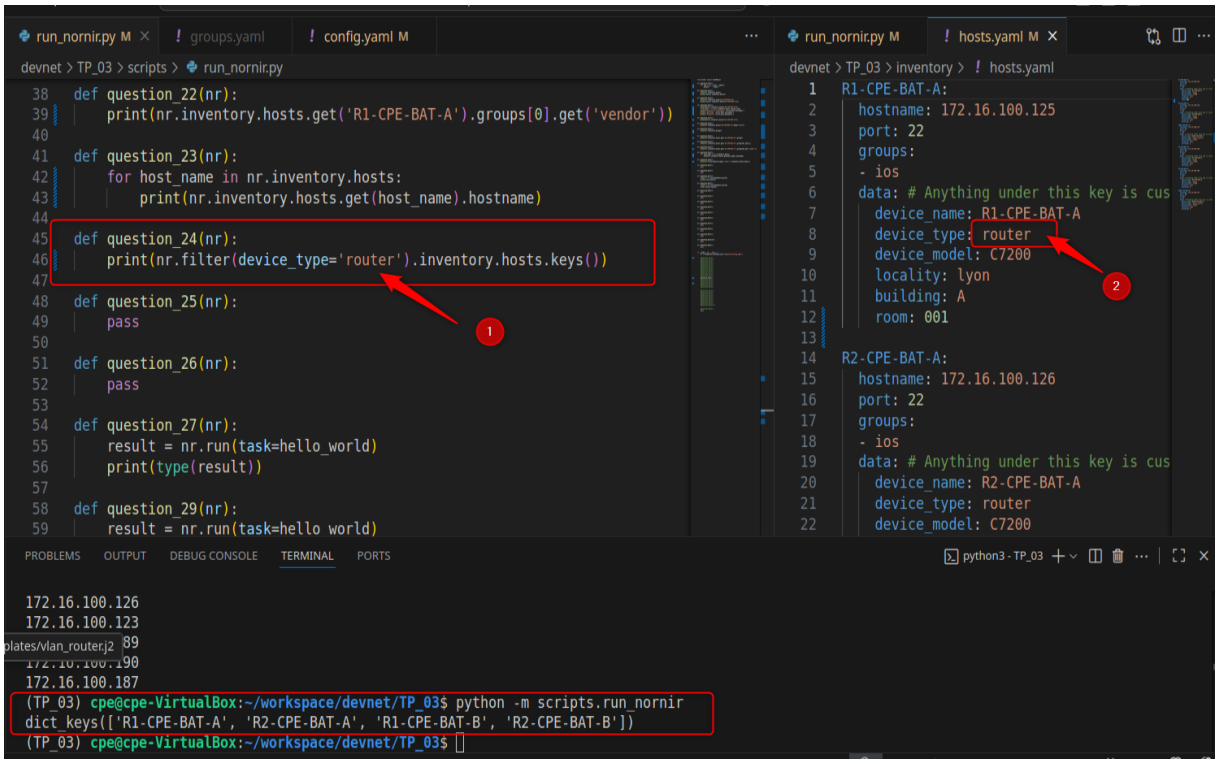
La ligne nr.inventory.hosts.get('R1-CPE-BAT-A').groups[0].get('vendor') permet de récupérer l'objet hôte R1-CPE-BAT-A, d'accéder à son premier groupe associé, puis d'extraire et d'afficher la valeur du champ vendor (ici, "Cisco").

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.



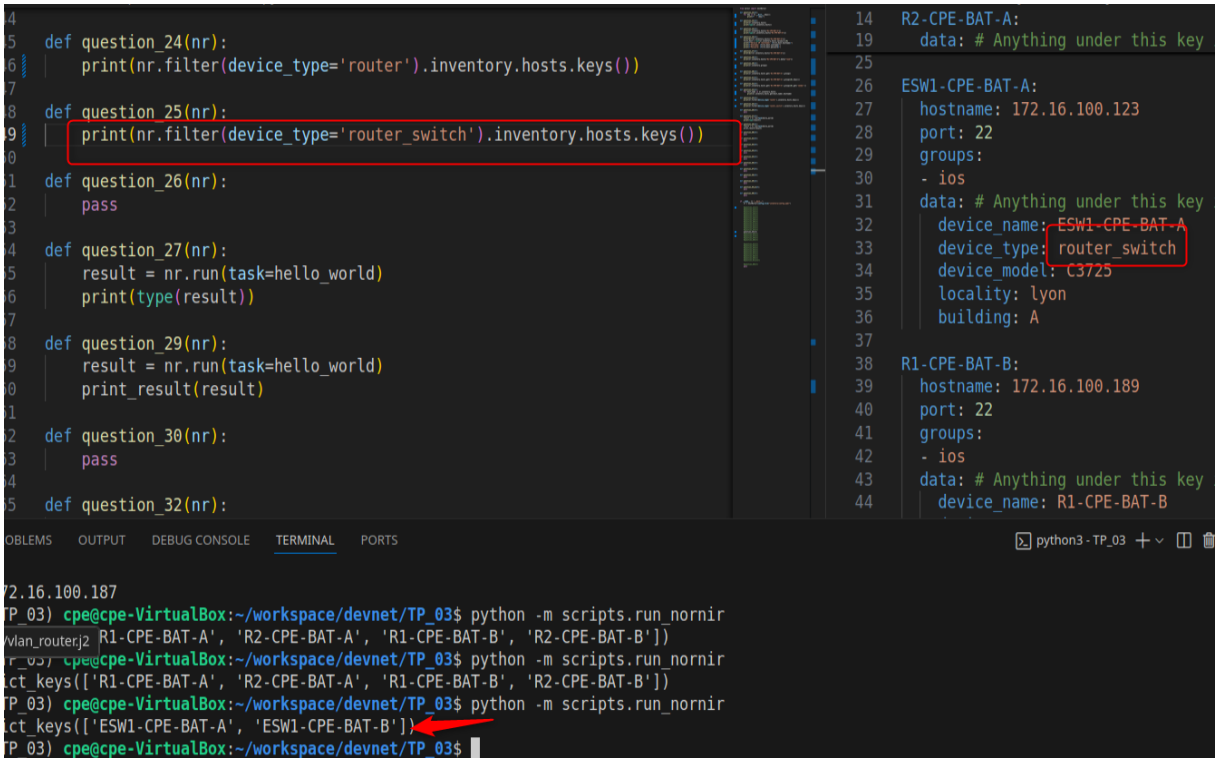
La fonction parcourt chaque hôte défini dans nr.inventory.hosts et affiche son hostname (adresse IP) en récupérant directement l'attribut depuis l'objet associé à chaque nom d'hôte.

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



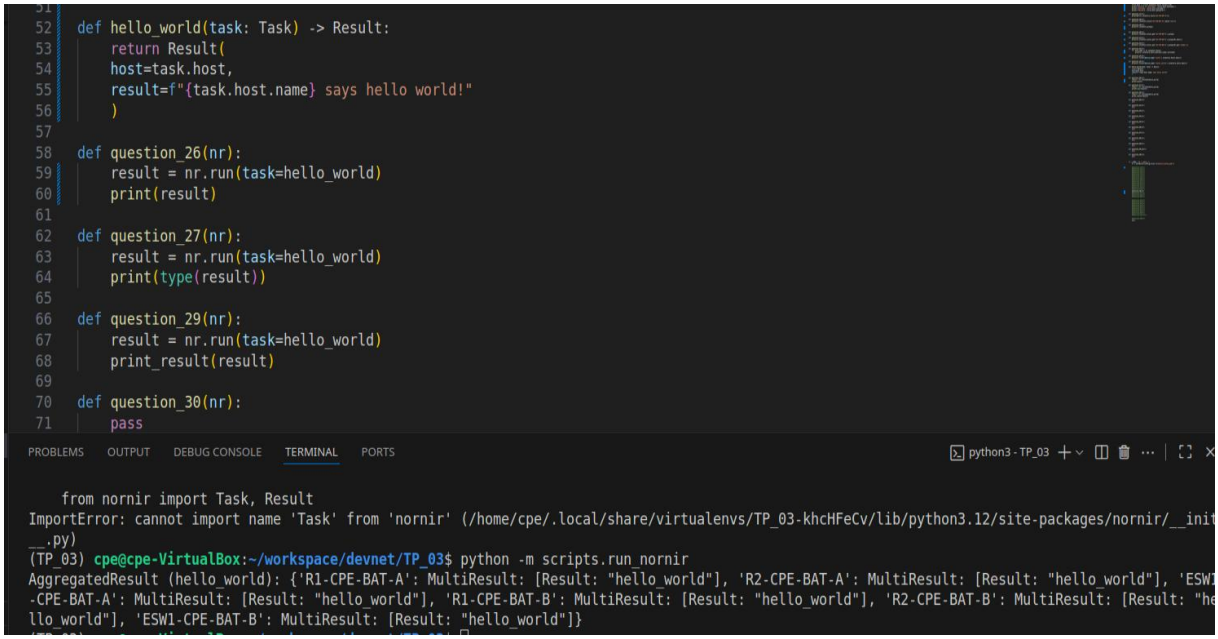
La fonction utilise la méthode .filter(device_type='router') pour sélectionner uniquement les hôtes dont le type est défini comme "router", puis affiche la liste des noms de ces hôtes avec .inventory.hosts.keys().

Question 1.9.25 : Affichez à présent la liste des hosts de type router_switch



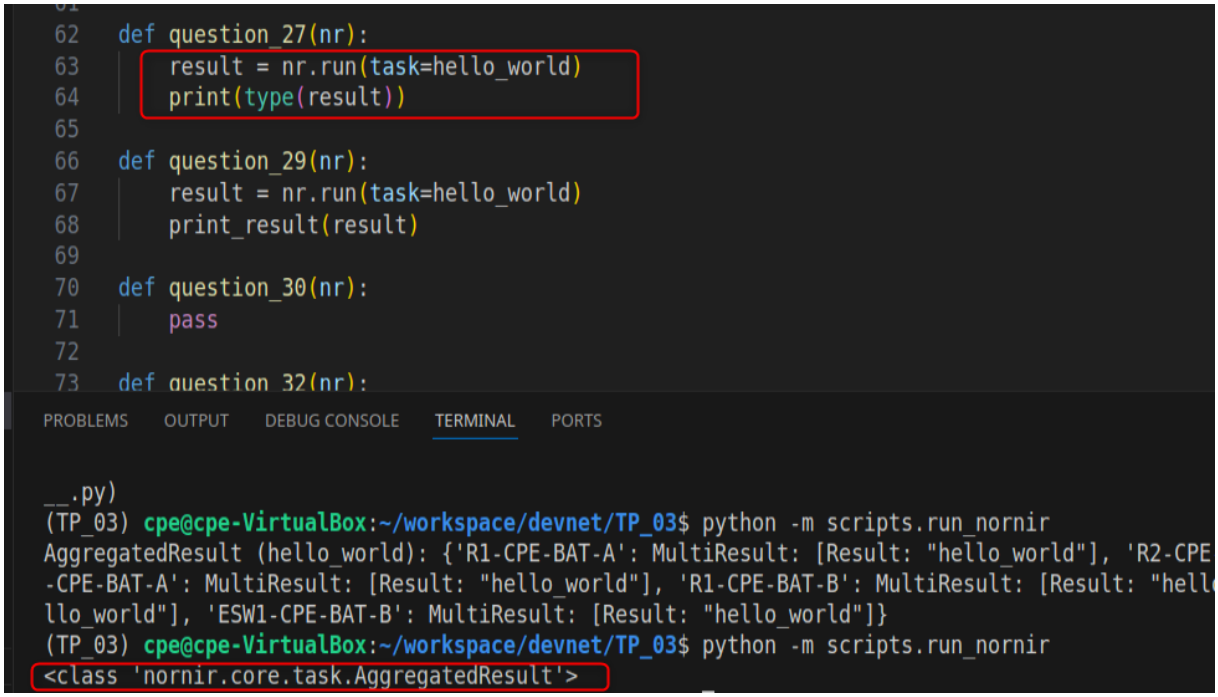
Je filtre maintenant le device_type sur "router_switch" pour afficher uniquement les switchs présents dans l'inventaire.

Question 1.9.26 : Importez les classes Task et Result et définissez une première task nommée 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()



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 ?

```
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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
ESW1-CPE-BAT-A says hello world!
^^^^ END hello_world ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* ESW1-CPE-BAT-B ** changed : False ******************************************
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
ESW1-CPE-BAT-B says hello world!
^^^^ END hello_world ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* R1-CPE-BAT-A ** changed : False *******************************************
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
R1-CPE-BAT-A says hello world!
^^^^ END hello_world ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* R1-CPE-BAT-B ** changed : False *******************************************
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
R1-CPE-BAT-B says hello world!
^^^^ END hello_world ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* R2-CPE-BAT-A ** changed : False *******************************************
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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.

```
70
71  def question_30(nr):
72      host = nr.filter(device_type='router_switch')
73      print_result(host.run(task=hello_world))
74
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

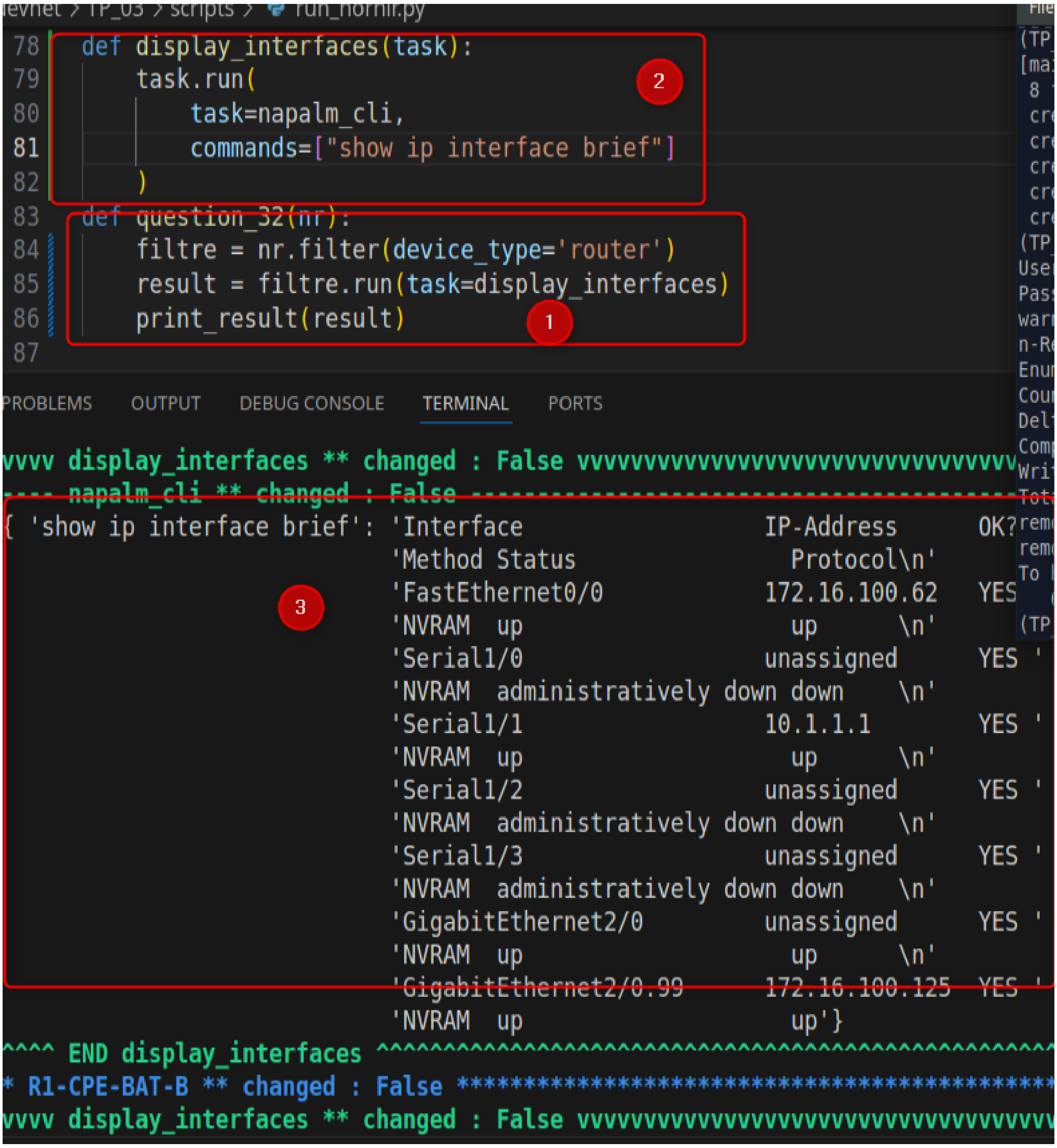
```
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
hello_world*********************************************************************
* ESW1-CPE-BAT-A ** changed : False ***********************************************
vvvv hello_world ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv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

<u>Question 1.9.33 :</u> Développez une task permettant d'afficher la table arp de chaque switch référencé dans l'inventory. Utilisez la fonction napalm_get pour cela.

```python
def get_arp_table(task):
    task.run(
        task=napalm_get,
        getters=["arp_table"]
    )

def question_33(nr):
    switches = nr.filter(device_type="router_switch")
    result = switches.run(task=get_arp_table)
    print_result(result)
```

```
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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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

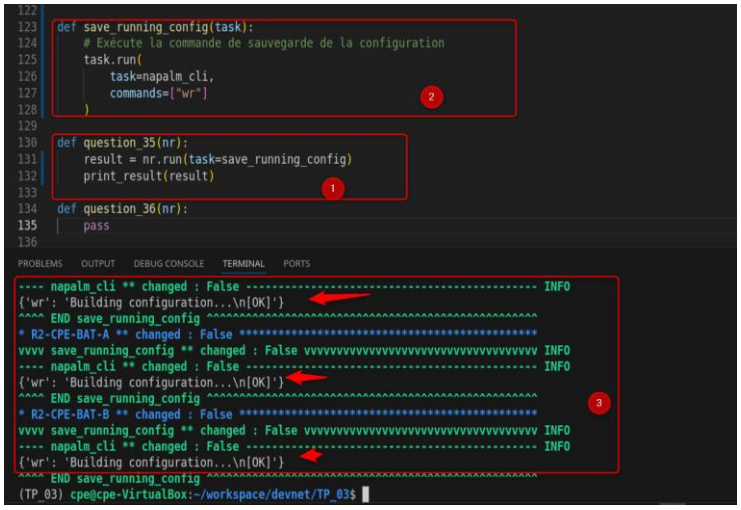
```
def configure_loopback_r1(task):
    task.run(
        task=napalm_configure,
        configuration="""
interface Lo1
 ip address 1.1.1.1 255.255.255.255
 description Loopback pour R1-CPE-BAT-A
"""
    )

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
"""
    )
```

```python
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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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.

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



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.

<u>Question 1.9.36 :</u> Développez une fonction permettant d'afficher l'état des interfaces de chaque routeur à l'aide de la task netmiko_send_command.

```
def question_36(nr):
    filtre = nr.filter(device_type='router')
    result = filtre.run(task=netmiko_send_command,command_string="show ip interface brief")
    print_result(result)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

^^^^ END netmiko_send_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* R2-CPE-BAT-A ** changed : False **********************************************
vvvv netmiko_send_command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            unassigned      YES NVRAM  administratively down down
Serial1/0                  unassigned      YES NVRAM  administratively down down
Serial1/1                  unassigned      YES NVRAM  administratively down down
Serial1/2                  10.1.2.1        YES NVRAM  up                    up
Serial1/3                  unassigned      YES NVRAM  administratively down down
GigabitEthernet2/0         unassigned      YES NVRAM  up                    up
GigabitEthernet2/0.99      172.16.100.126  YES NVRAM  up                    up
Loopback1                  2.2.2.2         YES TFTP   up                    up
^^^^ END netmiko_send_command ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
* R2-CPE-BAT-B ** changed : False **********************************************
vvvv netmiko_send_command ** changed : False vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv INFO
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            unassigned      YES NVRAM  administratively down down
Serial1/0                  10.102.1.1      YES NVRAM  up                    down
Serial1/1                  unassigned      YES NVRAM  administratively down down
Serial1/2                  unassigned      YES NVRAM  administratively down down
Serial1/3                  unassigned      YES NVRAM  administratively down down
Serial2/0                  unassigned      YES NVRAM  administratively down down
```

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

<u>Question 1.9.37 :</u> 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.

R2-CPE-BAT-A

```
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
!
```

<u>Question 1.9.38 :</u> 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

```python
def question_38(nr):
    filtreR1 = nr.filter(name='R1-CPE-BAT-A')
    resultR1 = filtreR1.run(task=netmiko_save_config)
    filtreR2 = nr.filter(name='R2-CPE-BAT-A')
    resultR2 = filtreR2.run(task=netmiko_save_config)
    print_result(resultR1)
    print_result(resultR2)

def question_39(nr):
    pass

def question_39_d(nr):
    pass
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
R1-CPE-BAT-A#
^^^^ END netmiko_save_config ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
netmiko_save_config*************************************************************
* R2-CPE-BAT-A ** changed : True ***********************************************
vvvv netmiko_save_config ** changed : True vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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.

```python
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
```

```python
def deploy_vrrp(task):
    vrrp_filename = task.host.name.replace("-", "_").replace("CPE", "CPE_LYON") + "_VRRP.co
    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 ro

    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)
```

```python
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

```
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
```

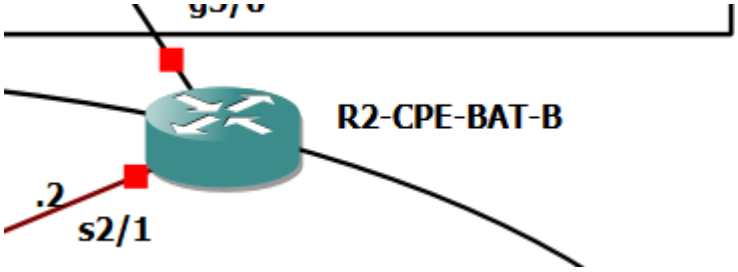
Question 1.9.39.b : Vérifier l'état du HA entre les routeurs de chaque bâtiment : show vrrp brief. Le routeur R1 de chaque bâtiment doit être le backup et R2 doit être le master

```
R1-CPE-BAT-A#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)
```

```
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

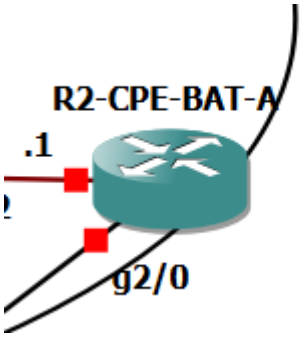
Si j'éteins le routeur R2-CPE-BAT-B



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

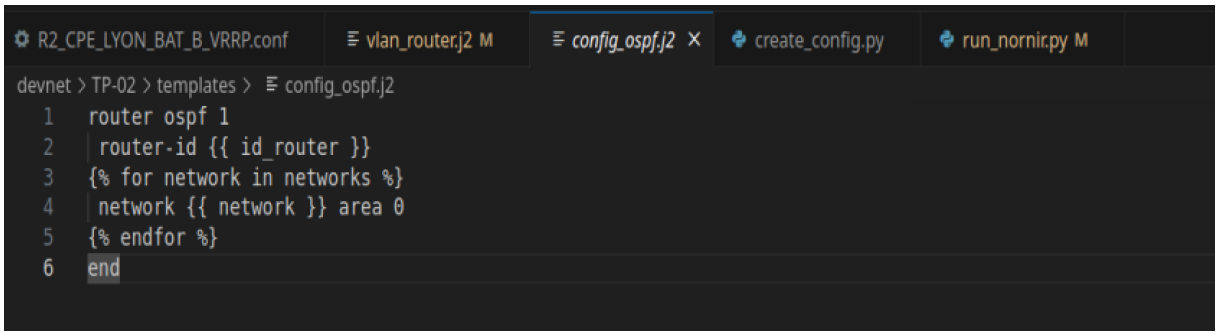
```python
def deploy_config_from_file(task: Task, config_file: str) -> Result:
    """Déploie la configuration depuis un fichier sur un équipement via Nornir/Netmiko."""
    with open(config_file, "r") as f:
        commands = f.read().splitlines()
    result = task.run(task=netmiko_send_config, config_commands=commands)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

```python
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)
```

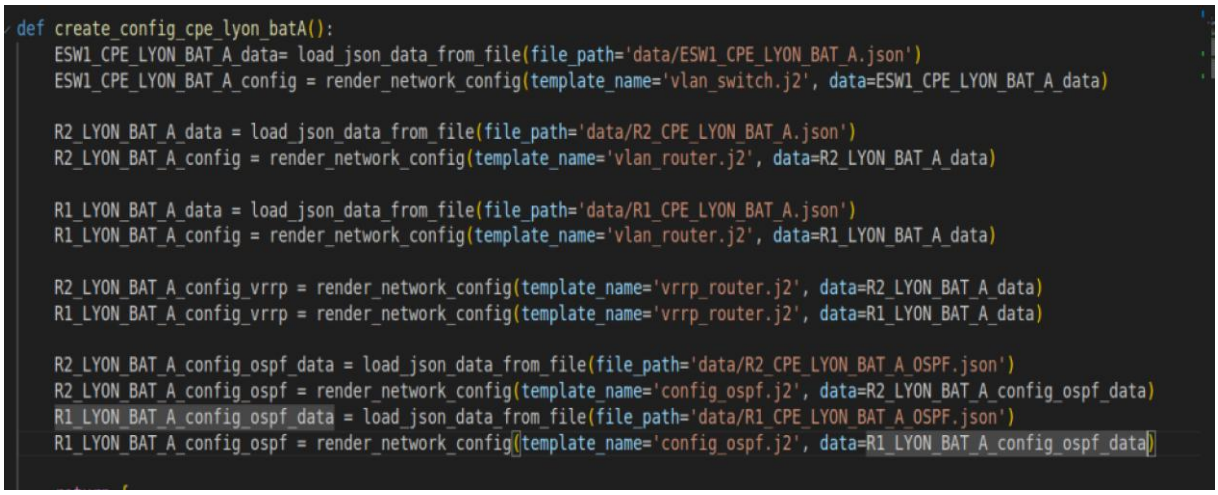
```python
def question_39(nr):
    host_patterns = [
        'R1-CPE-BAT-A',
        'R2-CPE-BAT-A',
        'R1-CPE-BAT-B',
        'R2-CPE-BAT-B',
        'ESW1-CPE-BAT-A',
        'ESW1-CPE-BAT-BS'


    ]
    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).

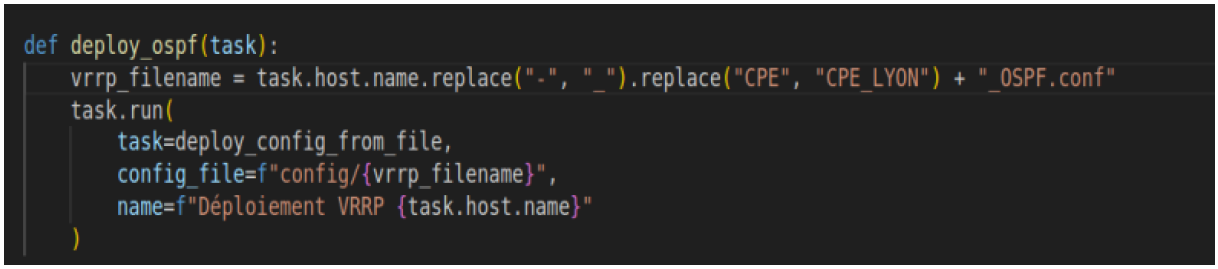
J'ai commencé par reprendre le template Jinja du TP02 pour produire automatiquement mes configurations OSPF dans des fichiers distincts.



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



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



```python
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