

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

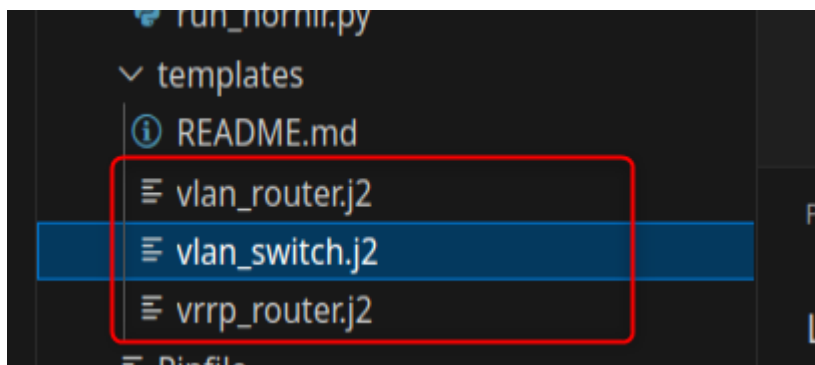
Partie 1 Générer et déployer des configurations réseau avec Python et Nornir

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)

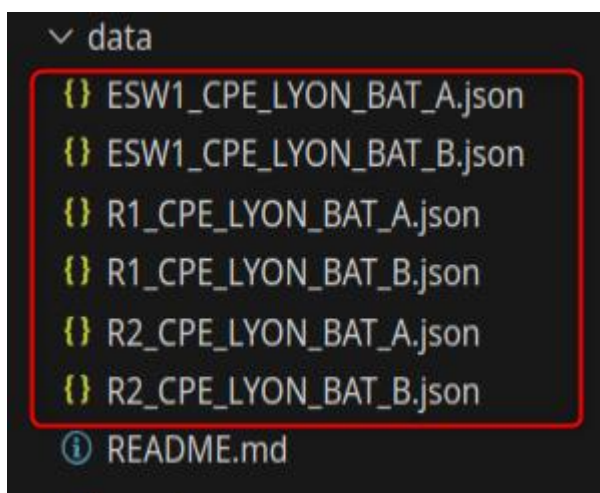
Question 1.8.1 : Développez l'ensemble des templates Jinja2 qui vous permettront de générer la configuration nécessaire pour le bon fonctionnement du bâtiment A. (Pour rappel, le vlan d'admin (99) est déjà configuré sur les équipements.) Les templates Jinja2 doivent être sauvegardés dans le dossier templates.

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_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).

```
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	U
⚙ R1_CPE_LYON_BAT_A.conf	U
⚙ R2_CPE_LYON_BAT_A.conf	U

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

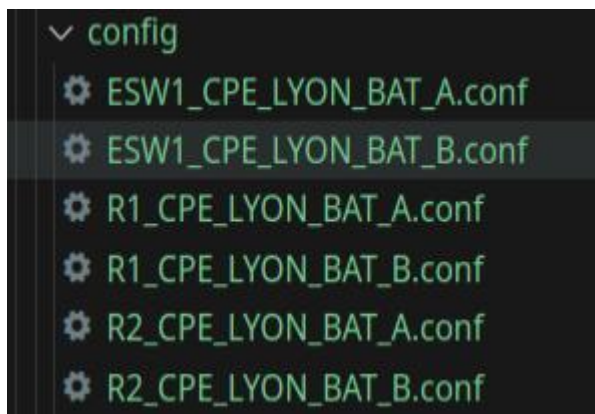
```
def create_config_cpe_lyon_batB():
    ESW1_CPE_LYON_BAT_B_data= load_json_data_from_file(file_path='data/ESW1_CPE_LYON_BAT_B.json')
    ESW1_CPE_LYON_BAT_B_config = render_network_config(template_name='vlan_switch.j2', data=ESW1_CPE_LYON_BAT_B_data)

    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_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):

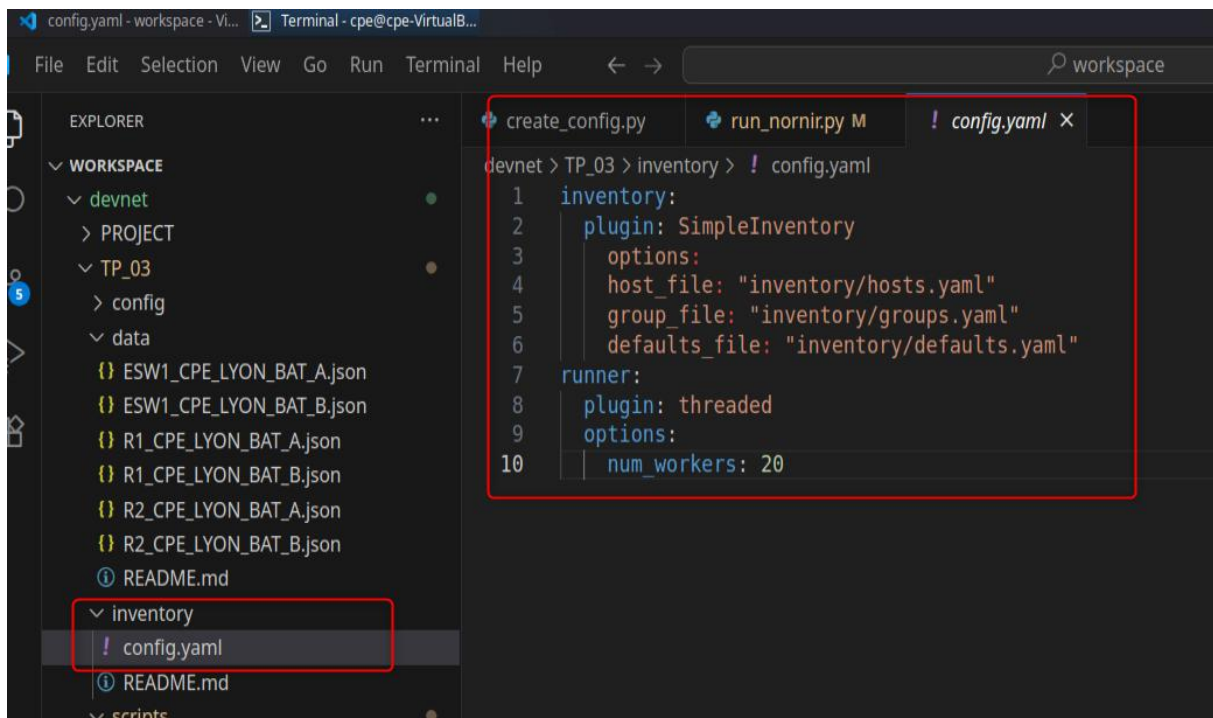
Nous installons le module Python Nornir dans l'environnement virtuel géré par Pipenv

```
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):
```

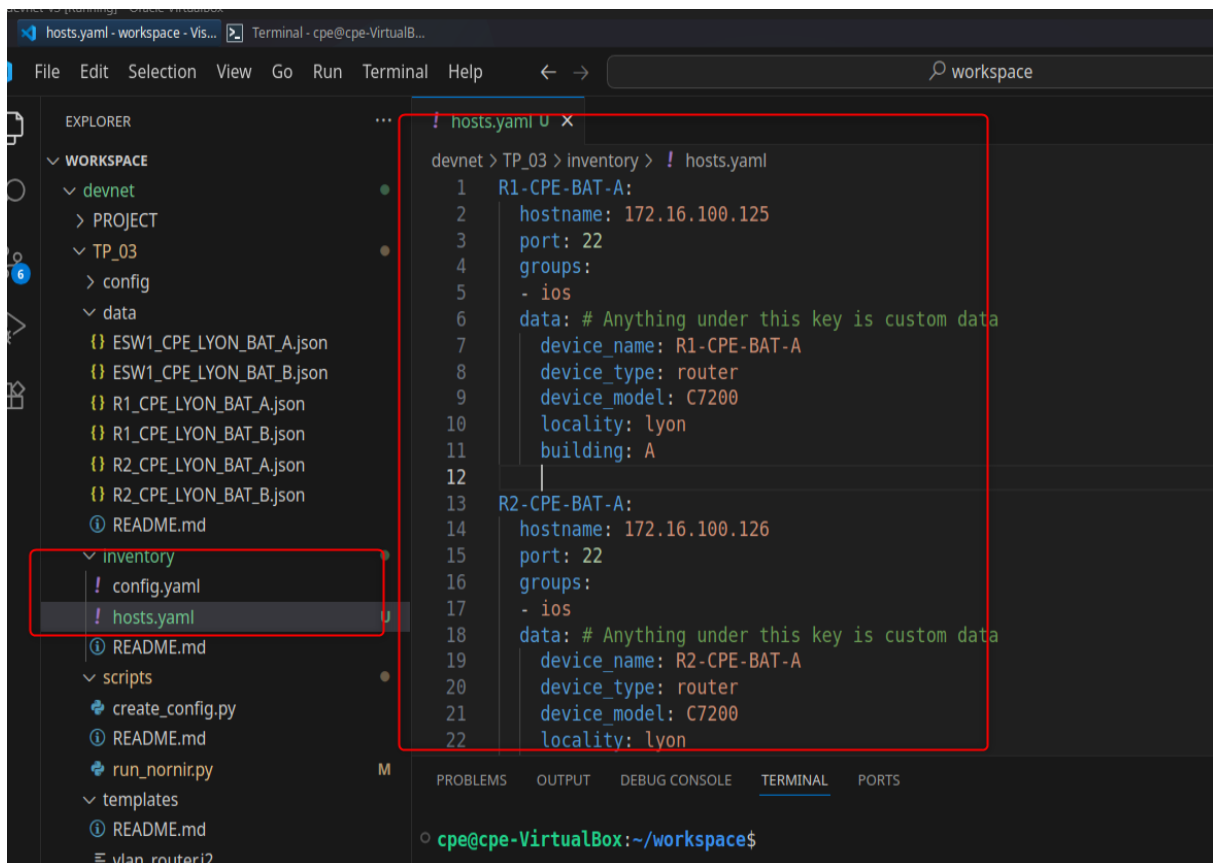
Question 1.9.8 : Créez un fichier config.yaml dans le dossier inventory de votre workspace et ajoutez la config suivante



The screenshot shows the Visual Studio Code interface. On the left, the Explorer panel displays the workspace structure. The 'inventory' folder is expanded, and 'config.yaml' is highlighted. On the right, the editor shows the content of 'config.yaml' with a red box highlighting the entire file content.

```
devnet > TP_03 > inventory > ! config.yaml
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
```

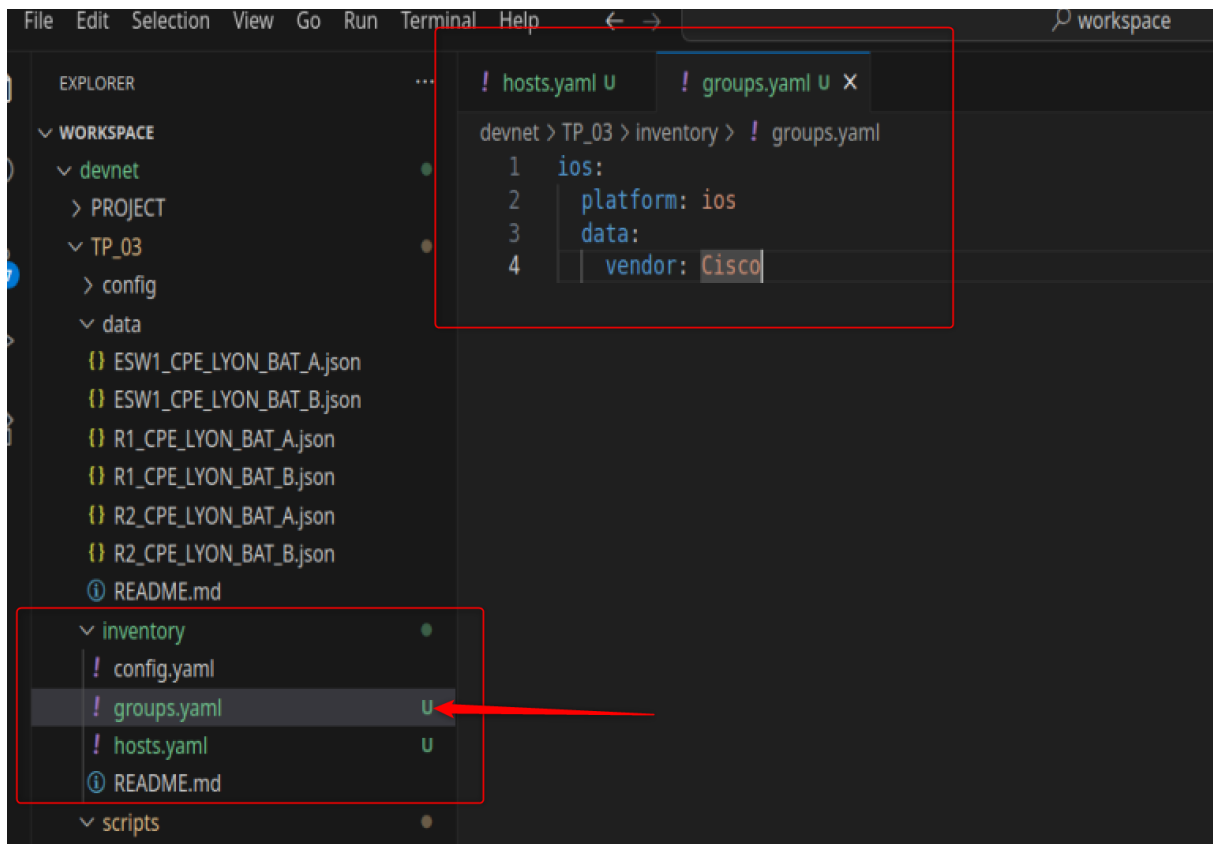
Question 1.9.9 : Créez le fichier hosts.yaml dans le dossier inventory et ajoutez la structure de données suivante :



The screenshot shows the Visual Studio Code interface. On the left, the Explorer panel displays the workspace structure. The 'inventory' folder is expanded, and 'hosts.yaml' is highlighted. On the right, the editor shows the content of 'hosts.yaml' with a red box highlighting the entire file content.

```
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
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
```

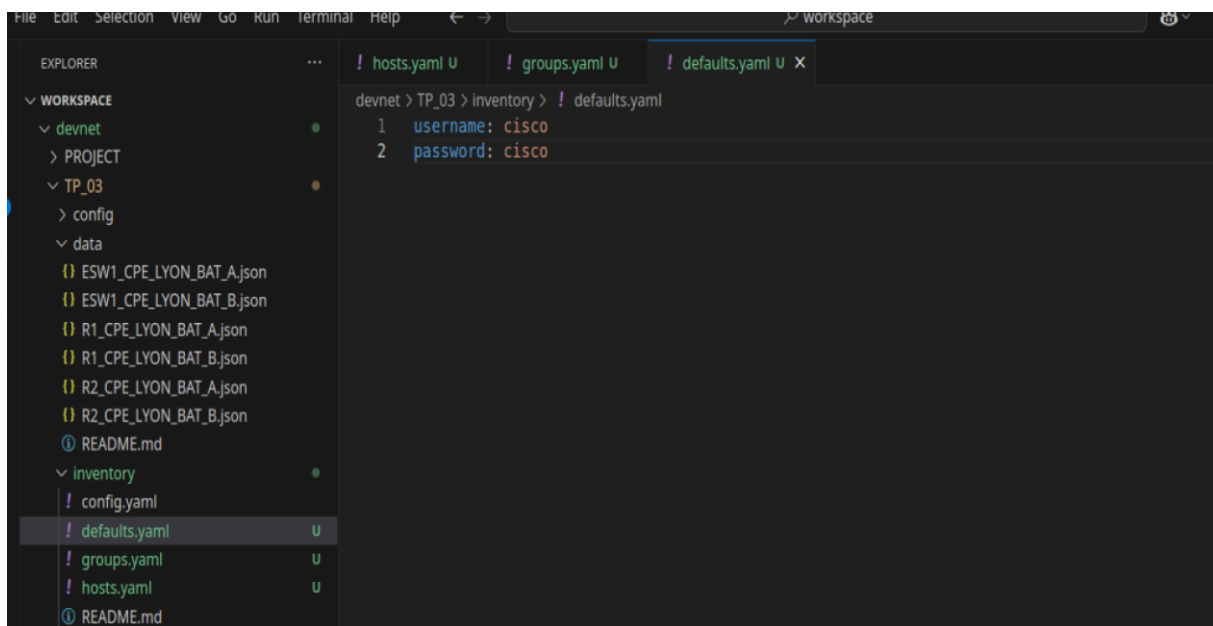
Question 1.9.10 : Créez le fichier groups.yaml dans le dossier inventory et ajoutez la structure de données suivante.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure. The 'inventory' folder is expanded, showing 'config.yaml', 'groups.yaml', 'hosts.yaml', and 'README.md'. The 'groups.yaml' file is selected and highlighted with a red box, and a red arrow points to it. On the right, the editor shows the content of 'groups.yaml' in the 'devnet > TP_03 > inventory' context. The file contains the following YAML structure:

```
1 ios:
2   platform: ios
3   data:
4     vendor: Cisco
```

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



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure. The 'inventory' folder is expanded, showing 'config.yaml', 'defaults.yaml', 'groups.yaml', 'hosts.yaml', and 'README.md'. The 'defaults.yaml' file is selected and highlighted with a red box. On the right, the editor shows the content of 'defaults.yaml' in the 'devnet > TP_03 > inventory' context. The file contains the following YAML structure:

```
1 username: cisco
2 password: cisco
```

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 0x78909a6e9500>, 'processors': [], '_runner': <nornir.plugins.runners.ThreadedRunner object at 0x78909af96240>}
<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 0x7f6e868f5a90>}
(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. Parmi 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_', '_getattr_', '_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_', '_connectns_', '_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_']
```

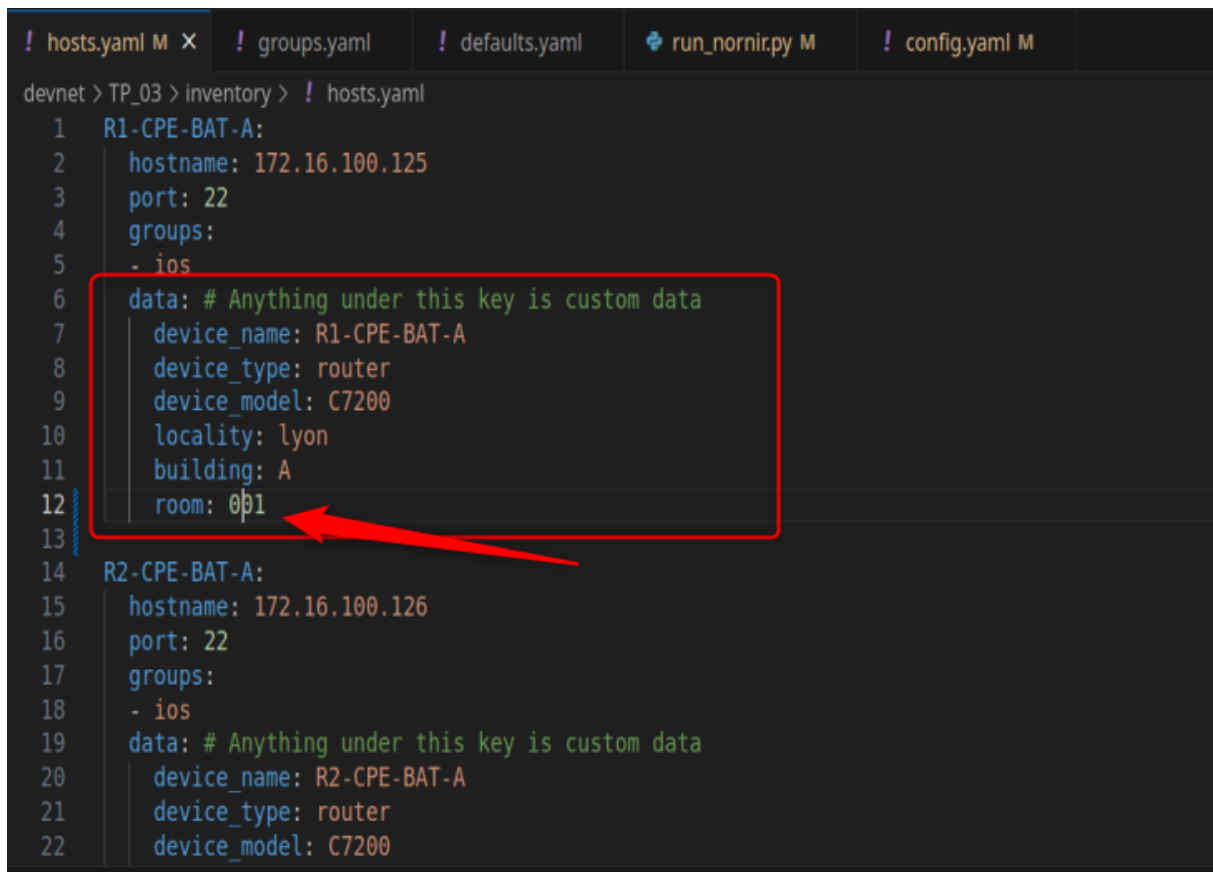
Dans un inventaire Nornir, les credentials et l'adresse IP proviennent du fichier `hosts.yaml`

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

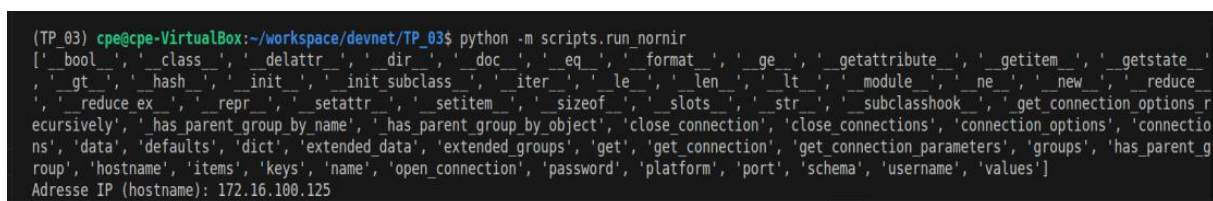
```
def question_16(nr):
    print(dir(nr.inventory.hosts["R1-CPE-BAT-A"]))
    first_host = list(nr.inventory.hosts.values())[0]
    print(f"Adresse IP (hostname): {first_host.hostname}")
    print(f"Username: {first_host.username}")
    print(f>Password: {first_host.password}")
```

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 exécutez 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_', '_getattr_', '_getitem_', '_getstate_',
'_gt_', '_hash_', '_init_', '_init_subclass_', '_iter_', '_le_', '_len_', '_lt_', '_module_', '_ne_', '_new_', '_reduce_',
'_repr_', '_setattr_', '_setitem_', '_sizeof_', '_slots_', '_str_', '_subclasshook_', '_get_connection_options_r',
'ecursively', '_has_parent_group_by_name', '_has_parent_group_by_object', '_close_connection', '_close_connections', '_connection_options', '_connectio',
ns', '_data', '_defaults', '_dict', '_extended_data', '_extended_groups', '_get', '_get_connection', '_get_connection_parameters', '_groups', '_has_parent_g',
roup', '_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" créée 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('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`.

```
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 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):
54     pass
55
56 def question_28(nr):
57     pass
58
59 def question_29(nr):
60     pass
61
62 def question_30(nr):
63     pass
64
65 def question_31(nr):
66     pass
67
68 def question_32(nr):
69     pass
70
71 def question_33(nr):
72     pass
73
74 def question_34(nr):
75     pass
76
77 def question_35(nr):
78     pass
79
80 def question_36(nr):
81     pass
82
83 def question_37(nr):
84     pass
85
86 def question_38(nr):
87     pass
88
89 def question_39(nr):
90     pass
91
92 def question_40(nr):
93     pass
94
95 def question_41(nr):
96     pass
97
98 def question_42(nr):
99     pass
100
101 def question_43(nr):
102     pass
103
104 def question_44(nr):
105     pass
106
107 def question_45(nr):
108     pass
109
110 def question_46(nr):
111     pass
112
113 def question_47(nr):
114     pass
115
116 def question_48(nr):
117     pass
118
119 def question_49(nr):
120     pass
121
122 def question_50(nr):
123     pass
124
125 def question_51(nr):
126     pass
127
128 def question_52(nr):
129     pass
130
131 def question_53(nr):
132     pass
133
134 def question_54(nr):
135     pass
136
137 def question_55(nr):
138     pass
139
140 def question_56(nr):
141     pass
142
143 def question_57(nr):
144     pass
145
146 def question_58(nr):
147     pass
148
149 def question_59(nr):
150     pass
151
152 def question_60(nr):
153     pass
154
155 def question_61(nr):
156     pass
157
158 def question_62(nr):
159     pass
160
161 def question_63(nr):
162     pass
163
164 def question_64(nr):
165     pass
166
167 def question_65(nr):
168     pass
169
170 def question_66(nr):
171     pass
172
173 def question_67(nr):
174     pass
175
176 def question_68(nr):
177     pass
178
179 def question_69(nr):
180     pass
181
182 def question_70(nr):
183     pass
184
185 def question_71(nr):
186     pass
187
188 def question_72(nr):
189     pass
190
191 def question_73(nr):
192     pass
193
194 def question_74(nr):
195     pass
196
197 def question_75(nr):
198     pass
199
200 def question_76(nr):
201     pass
202
203 def question_77(nr):
204     pass
205
206 def question_78(nr):
207     pass
208
209 def question_79(nr):
210     pass
211
212 def question_80(nr):
213     pass
214
215 def question_81(nr):
216     pass
217
218 def question_82(nr):
219     pass
220
221 def question_83(nr):
222     pass
223
224 def question_84(nr):
225     pass
226
227 def question_85(nr):
228     pass
229
230 def question_86(nr):
231     pass
232
233 def question_87(nr):
234     pass
235
236 def question_88(nr):
237     pass
238
239 def question_89(nr):
240     pass
241
242 def question_90(nr):
243     pass
244
245 def question_91(nr):
246     pass
247
248 def question_92(nr):
249     pass
250
251 def question_93(nr):
252     pass
253
254 def question_94(nr):
255     pass
256
257 def question_95(nr):
258     pass
259
260 def question_96(nr):
261     pass
262
263 def question_97(nr):
264     pass
265
266 def question_98(nr):
267     pass
268
269 def question_99(nr):
270     pass
271
272 def question_100(nr):
273     pass
274
275 def question_101(nr):
276     pass
277
278 def question_102(nr):
279     pass
280
281 def question_103(nr):
282     pass
283
284 def question_104(nr):
285     pass
286
287 def question_105(nr):
288     pass
289
290 def question_106(nr):
291     pass
292
293 def question_107(nr):
294     pass
295
296 def question_108(nr):
297     pass
298
299 def question_109(nr):
300     pass
301
302 def question_110(nr):
303     pass
304
305 def question_111(nr):
306     pass
307
308 def question_112(nr):
309     pass
310
311 def question_113(nr):
312     pass
313
314 def question_114(nr):
315     pass
316
317 def question_115(nr):
318     pass
319
320 def question_116(nr):
321     pass
322
323 def question_117(nr):
324     pass
325
326 def question_118(nr):
327     pass
328
329 def question_119(nr):
330     pass
331
332 def question_120(nr):
333     pass
334
335 def question_121(nr):
336     pass
337
338 def question_122(nr):
339     pass
340
341 def question_123(nr):
342     pass
343
344 def question_124(nr):
345     pass
346
347 def question_125(nr):
348     pass
349
350 def question_126(nr):
351     pass
352
353 def question_127(nr):
354     pass
355
356 def question_128(nr):
357     pass
358
359 def question_129(nr):
360     pass
361
362 def question_130(nr):
363     pass
364
365 def question_131(nr):
366     pass
367
368 def question_132(nr):
369     pass
370
371 def question_133(nr):
372     pass
373
374 def question_134(nr):
375     pass
376
377 def question_135(nr):
378     pass
379
380 def question_136(nr):
381     pass
382
383 def question_137(nr):
384     pass
385
386 def question_138(nr):
387     pass
388
389 def question_139(nr):
390     pass
391
392 def question_140(nr):
393     pass
394
395 def question_141(nr):
396     pass
397
398 def question_142(nr):
399     pass
400
401 def question_143(nr):
402     pass
403
404 def question_144(nr):
405     pass
406
407 def question_145(nr):
408     pass
409
410 def question_146(nr):
411     pass
412
413 def question_147(nr):
414     pass
415
416 def question_148(nr):
417     pass
418
419 def question_149(nr):
420     pass
421
422 def question_150(nr):
423     pass
424
425 def question_151(nr):
426     pass
427
428 def question_152(nr):
429     pass
430
431 def question_153(nr):
432     pass
433
434 def question_154(nr):
435     pass
436
437 def question_155(nr):
438     pass
439
440 def question_156(nr):
441     pass
442
443 def question_157(nr):
444     pass
445
446 def question_158(nr):
447     pass
448
449 def question_159(nr):
450     pass
451
452 def question_160(nr):
453     pass
454
455 def question_161(nr):
456     pass
457
458 def question_162(nr):
459     pass
460
461 def question_163(nr):
462     pass
463
464 def question_164(nr):
465     pass
466
467 def question_165(nr):
468     pass
469
470 def question_166(nr):
471     pass
472
473 def question_167(nr):
474     pass
475
476 def question_168(nr):
477     pass
478
479 def question_169(nr):
480     pass
481
482 def question_170(nr):
483     pass
484
485 def question_171(nr):
486     pass
487
488 def question_172(nr):
489     pass
490
491 def question_173(nr):
492     pass
493
494 def question_174(nr):
495     pass
496
497 def question_175(nr):
498     pass
499
500 def question_176(nr):
501     pass
502
503 def question_177(nr):
504     pass
505
506 def question_178(nr):
507     pass
508
509 def question_179(nr):
510     pass
511
512 def question_180(nr):
513     pass
514
515 def question_181(nr):
516     pass
517
518 def question_182(nr):
519     pass
520
521 def question_183(nr):
522     pass
523
524 def question_184(nr):
525     pass
526
527 def question_185(nr):
528     pass
529
530 def question_186(nr):
531     pass
532
533 def question_187(nr):
534     pass
535
536 def question_188(nr):
537     pass
538
539 def question_189(nr):
540     pass
541
542 def question_190(nr):
543     pass
544
545 def question_191(nr):
546     pass
547
548 def question_192(nr):
549     pass
550
551 def question_193(nr):
552     pass
553
554 def question_194(nr):
555     pass
556
557 def question_195(nr):
558     pass
559
560 def question_196(nr):
561     pass
562
563 def question_197(nr):
564     pass
565
566 def question_198(nr):
567     pass
568
569 def question_199(nr):
570     pass
571
572 def question_200(nr):
573     pass
574
575 def question_201(nr):
576     pass
577
578 def question_202(nr):
579     pass
580
581 def question_203(nr):
582     pass
583
584 def question_204(nr):
585     pass
586
587 def question_205(nr):
588     pass
589
590 def question_206(nr):
591     pass
592
593 def question_207(nr):
594     pass
595
596 def question_208(nr):
597     pass
598
599 def question_209(nr):
600     pass
601
602 def question_210(nr):
603     pass
604
605 def question_211(nr):
606     pass
607
608 def question_212(nr):
609     pass
610
611 def question_213(nr):
612     pass
613
614 def question_214(nr):
615     pass
616
617 def question_215(nr):
618     pass
619
620 def question_216(nr):
621     pass
622
623 def question_217(nr):
624     pass
625
626 def question_218(nr):
627     pass
628
629 def question_219(nr):
630     pass
631
632 def question_220(nr):
633     pass
634
635 def question_221(nr):
636     pass
637
638 def question_222(nr):
639     pass
640
641 def question_223(nr):
642     pass
643
644 def question_224(nr):
645     pass
646
647 def question_225(nr):
648     pass
649
650 def question_226(nr):
651     pass
652
653 def question_227(nr):
654     pass
655
656 def question_228(nr):
657     pass
658
659 def question_229(nr):
660     pass
661
662 def question_230(nr):
663     pass
664
665 def question_231(nr):
666     pass
667
668 def question_232(nr):
669     pass
670
671 def question_233(nr):
672     pass
673
674 def question_234(nr):
675     pass
676
677 def question_235(nr):
678     pass
679
680 def question_236(nr):
681     pass
682
683 def question_237(nr):
684     pass
685
686 def question_238(nr):
687     pass
688
689 def question_239(nr):
690     pass
691
692 def question_240(nr):
693     pass
694
695 def question_241(nr):
696     pass
697
698 def question_242(nr):
699     pass
700
701 def question_243(nr):
702     pass
703
704 def question_244(nr):
705     pass
706
707 def question_245(nr):
708     pass
709
710 def question_246(nr):
711     pass
712
713 def question_247(nr):
714     pass
715
716 def question_248(nr):
717     pass
718
719 def question_249(nr):
720     pass
721
722 def question_250(nr):
723     pass
724
725 def question_251(nr):
726     pass
727
728 def question_252(nr):
729     pass
730
731 def question_253(nr):
732     pass
733
734 def question_254(nr):
735     pass
736
737 def question_255(nr):
738     pass
739
740 def question_256(nr):
741     pass
742
743 def question_257(nr):
744     pass
745
746 def question_258(nr):
747     pass
748
749 def question_259(nr):
750     pass
751
752 def question_260(nr):
753     pass
754
755 def question_261(nr):
756     pass
757
758 def question_262(nr):
759     pass
760
761 def question_263(nr):
762     pass
763
764 def question_264(nr):
765     pass
766
767 def question_265(nr):
768     pass
769
770 def question_266(nr):
771     pass
772
773 def question_267(nr):
774     pass
775
776 def question_268(nr):
777     pass
778
779 def question_269(nr):
780     pass
781
782 def question_270(nr):
783     pass
784
785 def question_271(nr):
786     pass
787
788 def question_272(nr):
789     pass
790
791 def question_273(nr):
792     pass
793
794 def question_274(nr):
795     pass
796
797 def question_275(nr):
798     pass
799
800 def question_276(nr):
801     pass
802
803 def question_277(nr):
804     pass
805
806 def question_278(nr):
807     pass
808
809 def question_279(nr):
810     pass
811
812 def question_280(nr):
813     pass
814
815 def question_281(nr):
816     pass
817
818 def question_282(nr):
819     pass
820
821 def question_283(nr):
822     pass
823
824 def question_284(nr):
825     pass
826
827 def question_285(nr):
828     pass
829
830 def question_286(nr):
831     pass
832
833 def question_287(nr):
834     pass
835
836 def question_288(nr):
837     pass
838
839 def question_289(nr):
840     pass
841
842 def question_290(nr):
843     pass
844
845 def question_291(nr):
846     pass
847
848 def question_292(nr):
849     pass
850
851 def question_293(nr):
852     pass
853
854 def question_294(nr):
855     pass
856
857 def question_295(nr):
858     pass
859
860 def question_296(nr):
861     pass
862
863 def question_297(nr):
864     pass
865
866 def question_298(nr):
867     pass
868
869 def question_299(nr):
870     pass
871
872 def question_300(nr):
873     pass
874
875 def question_301(nr):
876     pass
877
878 def question_302(nr):
879     pass
880
881 def question_303(nr):
882     pass
883
884 def question_304(nr):
885     pass
886
887 def question_305(nr):
888     pass
889
890 def question_306(nr):
891     pass
892
893 def question_307(nr):
894     pass
895
896 def question_308(nr):
897     pass
898
899 def question_309(nr):
900     pass
901
902 def question_310(nr):
903     pass
904
905 def question_311(nr):
906     pass
907
908 def question_312(nr):
909     pass
910
911 def question_313(nr):
912     pass
913
914 def question_314(nr):
915     pass
916
917 def question_315(nr):
918     pass
919
920 def question_316(nr):
921     pass
922
923 def question_317(nr):
924     pass
925
926 def question_318(nr):
927     pass
928
929 def question_319(nr):
930     pass
931
932 def question_320(nr):
933     pass
934
935 def question_321(nr):
936     pass
937
938 def question_322(nr):
939     pass
940
941 def question_323(nr):
942     pass
943
944 def question_324(nr):
945     pass
946
947 def question_325(nr):
948     pass
949
950 def question_326(nr):
951     pass
952
953 def question_327(nr):
954     pass
955
956 def question_328(nr):
957     pass
958
959 def question_329(nr):
960     pass
961
962 def question_330(nr):
963     pass
964
965 def question_331(nr):
966     pass
967
968 def question_332(nr):
969     pass
970
971 def question_333(nr):
972     pass
973
974 def question_334(nr):
975     pass
976
977 def question_335(nr):
978     pass
979
980 def question_336(nr):
981     pass
982
983 def question_337(nr):
984     pass
985
986 def question_338(nr):
987     pass
988
989 def question_339(nr):
990     pass
991
992 def question_340(nr):
993     pass
994
995 def question_341(nr):
996     pass
997
998 def question_342(nr):
999     pass
1000 def question_343(nr):
1001     pass
1002 def question_344(nr):
1003     pass
1004 def question_345(nr):
1005     pass
1006 def question_346(nr):
1007     pass
1008 def question_347(nr):
1009     pass
1009
```

```
(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$
```

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

```
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):
54     pass
55
56 def question_28(nr):
57     pass
58
59 def question_29(nr):
60     pass
61
62 def question_30(nr):
63     pass
64
65 def question_31(nr):
66     pass
67
68 def question_32(nr):
69     pass
70
71 def question_33(nr):
72     pass
73
74 def question_34(nr):
75     pass
76
77 def question_35(nr):
78     pass
79
80 def question_36(nr):
81     pass
82
83 def question_37(nr):
84     pass
85
86 def question_38(nr):
87     pass
88
89 def question_39(nr):
90     pass
91
92 def question_40(nr):
93     pass
94
95 def question_41(nr):
96     pass
97
98 def question_42(nr):
99     pass
100
101 def question_43(nr):
102     pass
103
104 def question_44(nr):
105     pass
106
107 def question_45(nr):
108     pass
109
110 def question_46(nr):
111     pass
112
113 def question_47(nr):
114     pass
115
116 def question_48(nr):
117     pass
118
119 def question_49(nr):
120     pass
121
122 def question_50(nr):
123     pass
124
125 def question_51(nr):
126     pass
127
128 def question_52(nr):
129     pass
129
```

```
(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.

```
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


```
def question_24(nr):
    print(nr.filter(device_type='router').inventory.hosts.keys())

def question_25(nr):
    pass

def question_26(nr):
    pass

def question_27(nr):
    result = nr.run(task=hello_world)
    print(type(result))

def question_29(nr):
    result = nr.run(task=hello_world)
```

```
172.16.100.126
172.16.100.123
plates/vlan_routerj2 89
172.16.100.190
172.16.100.187
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict 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$
```

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

```
def question_24(nr):
    print(nr.filter(device_type='router').inventory.hosts.keys())

def question_25(nr):
    print(nr.filter(device_type='router_switch').inventory.hosts.keys())

def question_26(nr):
    pass

def question_27(nr):
    result = nr.run(task=hello_world)
    print(type(result))

def question_29(nr):
    result = nr.run(task=hello_world)
    print_result(result)

def question_30(nr):
    pass

def question_32(nr):
```

```
172.16.100.187
(TP_03) cpe@cpe-VirtualBox:~/workspace/devnet/TP_03$ python -m scripts.run_nornir
dict 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
dict 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
52 def hello_world(task: Task) -> Result:
53     return Result(
54         host=task.host,
55         result=f"{task.host.name} says hello world!"
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

```
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()

```
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-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"]}
(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.

```
1 from nornir import InitNornir
2 from nornir.core.task import Task, Result
3
```

[illegible]

Question 1.9.30 : Faites en sorte que la task `hello_world` s'exécute uniquement sur les équipements de type `router switch`.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS


```
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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv 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 / switches) depuis napalm cli

[illegible]

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
```

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 vvvvvvvvvvvvvvvvvvvvvvvvvv 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_to_hosts(nr, host_patterns):
    """Déploie les configurations pour une liste de motifs d'hôtes."""
    for pattern in host_patterns:
        filtered_hosts = nr.filter(name=pattern)
        if not filtered_hosts.inventory.hosts:
            print(f"Aucun hôte correspondant à '{pattern}'")
            continue

        for host_name, host_obj in filtered_hosts.inventory.hosts.items():
            filename = host_obj.name.replace("-", "_").replace("CPE", "CPE_LYON")
            filtered_hosts.run(
                task=deploy_config_from_file,
                config_file=f"config/{filename}.conf",
                name=f"Déploiement config {host_obj.name}"
            )
```

```
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.

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

```
R2-CPE-BAT-A#
R2-CPE-BAT-A#show vrrp
GigabitEthernet2/0.99 - Group 99
  State is Master
  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 110
  Master Router is 172.16.100.126 (local), priority is 110
  Master Advertisement interval is 1.000 sec
  Master Down interval is 3.570 sec
```

```
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)
```

Question 1.9.39.c : 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 brief
Interface      Grp Pri Time  Own Pre State  Master addr  Group addr
Gi2/0.99       99 100 3609    Y Backup 172.16.100.126 172.16.100.124
Gi2/0.10       10 110 3570    Y Backup 172.16.10.254 172.16.10.252
Gi2/0.20       20 110 3570    Y Backup 172.16.20.254 172.16.20.252
```

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

```
R2_LYON_BAT_B_config_vrrp = render_network_config(template_name='vrrp_router.j2', data=R2_LYON_BAT_B_data)
R1_LYON_BAT_B_config_vrrp = render_network_config(template_name='vrrp_router.j2', data=R1_LYON_BAT_B_data)

80 | save_built_config('config/LYON_CPE_LYON_BAT_B_VRRP.conf', config.get('csw1'))
81 | save_built_config('config/R1_CPE_LYON_BAT_B_VRRP.conf', config.get('r1_vrrp'))
82 | save_built_config('config/R2_CPE_LYON_BAT_B_VRRP.conf', config.get('r2_vrrp'))
83 |
```

```

# Déploiement des fichiers VRRP uniquement pour les routers
routers = nr.filter(device_type='router')
for host_name, host_obj in routers.inventory.hosts.items():
    vrrp_filename = host_obj.name.replace("-", "_").replace("CPE", "CPE_LYON") + "_VRRP.conf"
    routers.run(
        task=deploy_config_from_file,
        config_file=f"config/{vrrp_filename}",
        name=f"Déploiement VRRP {host_obj.name}"
    )

```

Question 1.9.39.d : 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).