# TP 03

## Automatisation Réseau

## Module DEVNET

## Amar KANTAS

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

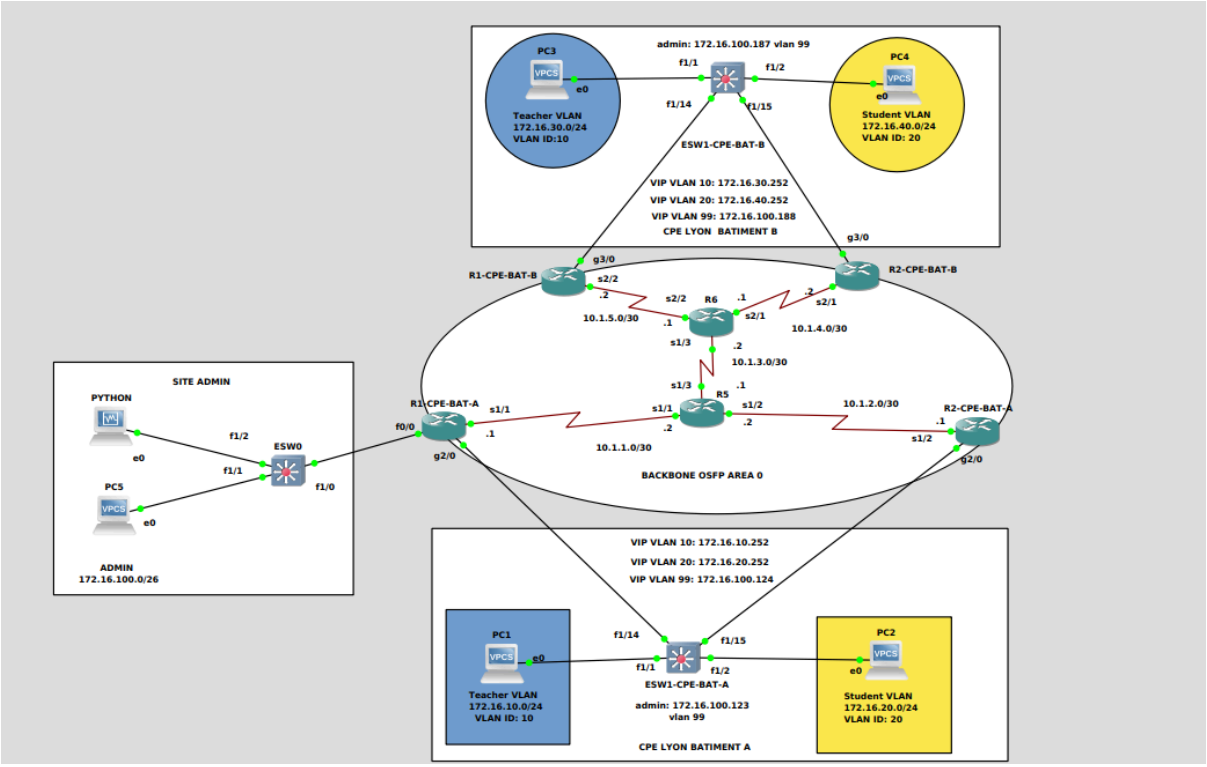
## 1-Automatiser le déploiement de configuration réseau sur des équipements Cisco

### 1.1- Objectif

Ce TP a pour objectif de développer en ensemble de scripts Python dans le but d'automatiser la création de configuration réseau et le déploiement de la configuration sur des routeurs Cisco 7200 et des switchs de niv2/ niv3 3725 Cisco.

Pour le déploiement automatique des configurations réseau sur les équipements nous utiliserons la librairie Nornir.

# 1.2 - Architecture réseau



## 1.3- Réseau d’administration

Nous disposons du réseau d’admin suivant : 172.16.100.0 /24. Ce réseau est découpé en 4 sous réseau /26 qui sont les suivants

**Réseau SITE admin:**
Network: 172.16.100.0/26
Broadcast: 172.16.100.63
HostMin: 172.16.100.1
HostMax: 172.16.100.62
Hosts/Net: 62

**Réseau admin CPE-LYON Batiment A:**
Network: 172.16.100.64/26
Broadcast: 172.16.100.127
HostMin: 172.16.100.65
HostMax: 172.16.100.126
Hosts/Net: 62

**Réseau admin CPE-LYON Batiment B:**
Network: 172.16.100.128/26
Broadcast: 172.16.100.191
HostMin: 172.16.100.129
HostMax: 172.16.100.190
Hosts/Net: 62

**Réseau admin non utilisé:**
Network: 172.16.100.192/26
Broadcast: 172.16.100.255
HostMin: 172.16.100.193
HostMax: 172.16.100.254
Hosts/Net: 62

Seules les adresses ip du réseau d’admin seront utilisées pour les connexions SSH depuis la machine virtuelle PYTHON.

Chaque routeur et switch dispose d’une adresse ip dans le réseau d’admin. De manière générale les routeurs ont la dernière adresse ip de la plage et les switch l’avant-dernière.

Les switch disposent d’une interface de management pour l’accès à distance (via l’ip admin) qui est configurée dans le vlan 99 (admin) de chaque bâtiment.

## 1.4 Liste des équipements

### 1.4.1 - liste des équipements réseau et de leur adresse ip d'admin

| Hostname | Ip address | SITE |
|---|---|---|
| PC5 | 172.16.100.1 | ADMIN |
| PYTHON | 172.16.100.2 | ADMIN |
| R1-CPE-BAT-A | f0/0 : 172.16.100.62<br>g2/0.99 : 172.16.100.125 | CPE-LYON Batiment A |
| R2-CPE-BAT-A | g2/0.99 : 172.16.100.126 | CPE-LYON Batiment A |
| ESW1-CPE-BAT-A | 172.16.100.123 (vlan 99) | CPE-LYON Batiment A |
| R1-CPE-BAT-B | g3/0.99 : 172.16.100.189 | CPE-LYON Batiment B |
| R2-CPE-BAT-B | g2/0.99 : 172.16.100.190 | CPE-LYON Batiment B |
| ESW1-CPE-BAT-B | 172.16.100.187 (vlan 99) | CPE-LYON Batiment B |

### 1.4.2 - liste des équipements clients et de leur adresse ip

| Hostname | Ip address | SITE |
|---|---|---|
| PC1 | 172.16.10.1 (vlan 10 - teacher) | CPE-LYON Batiment A |
| PC2 | 172.16.20.1 (vlan 20 - student ) | CPE-LYON Batiment A |
| PC3 | 172.16.30.1 (vlan 10 - teacher) | CPE-LYON Batiment B |
| PC4 | 172.16.40.1 (vlan 20 - student ) | CPE-LYON Batiment B |

## 1.5- Prenez connaissance de l’architecture et de la configuration mise en place

**Eléments déjà configurés dans l’architecture :**

- L’ensemble des adresses ip d’admin et des PC client est configuré
- Les liaisons point à point entre les routeurs du backbone OSPF sont également configurées
- Le routage OSPF est configuré pour le subnet ADMIN seulement, permettant à la machine Python d’accéder à l’ensemble des équipements des deux bâtiments via leur adresse ip d’admin
- La haute disponibilité VRRP (HA) des bâtiments A et B est configurée sur le subnet d’admin seulement

**Travail demandé ((Il s’agit d’un résumé du TP, les questions sont à traiter à la section suivante):**

- Développez l’ensemble des scripts nécessaires pour générer la configuration réseau (Jinja2) du routage inter-vlan des Bâtiments A et B:
    - Développez l’ensemble des scripts nécessaires pour déployer automatiquement la configuration sur les équipements cibles (Routeurs et Switchs)
    - Testez le résultat de l’implémentation et faites vérifier par votre responsable pédagogique
- Développez l’ensemble des scripts nécessaire pour générer la configuration réseau (JInja2) du backbone OSPF (area 0) pour les subnets des vlans 10 (teacher) et 20 (student )
    - Développez l’ensemble des scripts nécessaire pour déployer automatiquement la configuration sur les équipements cibles
    - Testez le résultat de l’implémentation et faites vérifier par votre responsable pédagogique
- Développez l’ensemble des scripts nécessaire pour générer la configuration réseau (JInja2) du HA (VRRP) de chaque bâtiment pour les vlans 10 (teacher) et 20 (student )
    - Développez l’ensemble des scripts nécessaire pour déployer automatiquement la configuration sur les équipements cibles
    - Testez le résultat de l’implémentation et faites vérifier par votre responsable pédagogique
- Développez l’ensemble des scripts nécessaire pour générer des fichiers de backup de chaque équipement réseau de l’architecture

- Testez le résultat de l'implémentation et faites vérifier par votre responsable pédagogique

## 1.6 - Connectez-vous sur la machine virtuelle d'automatisation

Utilisez le login / password suivant : cpe / 123

Vérifiez que la machine virtuelle peut joindre l'ensemble des équipements du réseau d'ADMIN. Si l'accès au réseau d'admin ne fonctionne pas, référez-vous à la procédure décrite dans le document "Préparation de l'environnement GNS3".

La machine virtuelle doit pouvoir joindre les équipements de l'admin suivants :

| Hostname | Ip address | SITE |
|---|---|---|
| R1-CPE-BAT-A | f0/0 : 172.16.100.62<br>g2/0.99 : 172.16.100.125 | CPE-LYON Batiment A |
| R2-CPE-BAT-A | g2/0.99 : 172.16.100.126 | CPE-LYON Batiment A |
| ESW1-CPE-BAT-A | 172.16.100.123 (vlan 99) | CPE-LYON Batiment A |
| R1-CPE-BAT-B | g3/0.99 : 172.16.100.189 | CPE-LYON Batiment B |
| R2-CPE-BAT-B | g2/0.99 : 172.16.100.190 | CPE-LYON Batiment B |
| ESW1-CPE-BAT-B | 172.16.100.187 (vlan 99) | CPE-LYON Batiment B |
| R1-CPE-BAT-A | f0/0 : 172.16.100.62<br>g2/0.99 : 172.16.100.125 | CPE-LYON Batiment A |
| R2-CPE-BAT-A | g2/0.99 : 172.16.100.126 | CPE-LYON Batiment A |

## 1.7 - Environnement de développement

Prenez connaissance de l'environnement de développement en ouvrant l'éditeur de code VsCode installé dans la machine virtuelle. Ajoutez le dossier précédemment créé lors de la phase de démarrage du tp-01 à votre éditeur si ce n'est pas déjà fait.

Depuis votre terminal vscode, placez-vous à l'intérieur du dossier **TP-03** et installez les dépendances pour utiliser les fonctions du package Jinja2 dans votre environnement virtuel:

**pipenv install Jinja2**

Pensez ensuite à activer votre venv avec la commande suivante: **pipenv shell**

Vous devriez avoir un fichier Pipefile dans votre dossier TP-03. Si ce n'est pas le cas, veuillez solliciter votre responsable pédagogique.

Importez la librairie Jinja depuis le fichier scripts/create_config.py et définissez le dossier "templates" par défaut de Jinja dans une variable env:

```
from jinja2 import Template, Environment,  FileSystemLoader

env = Environment(loader=FileSystemLoader("templates"))
```

Exécutez le fichier create_config à l'aide de la commande : python -m scripts.create_config. Si vous avez une erreur de package qui s'affiche, sollicitez le responsable pédagogique.

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

A ce stade vous avez pris connaissance de l'architecture et des éléments de configuration en place sur les équipements réseau de l'architecture. Vous avez également vérifié le bon fonctionnement de votre environnement de développement.

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.

   L'objectif est que les machines du bâtiment A puissent communiquer entre elles. Nommez les fichiers de templates de la manière suivante:
   - vlan_router.j2
   - vlan_switch.j2
   - vrrp_router.j2

   Note: Appuyez-vous sur ce qui a été fait en TP-01/02 et en cours

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.

   Nommez les fichiers de data de la manière suivante:
   - R1_CPE_LYON_BAT_A (.json ou .yaml)
   - R2_CPE_LYON_BAT_A (.json ou .yaml)
   - ESW1_CPE_LYON_BAT_A (.json ou .yaml)

Voici les adresses ip à renseigner dans vos fichiers de données :

| Hostname | Ip address |
|---|---|
| R1_CPE_LYON_BAT_A | g2/0.10 :<br>● IP: 172.16.10.253 - /24<br>● VIP: 172.16.10.252 (backup priority 100)<br>g2/0.20 :<br>● IP: 172.16.20.253 - /24 |

| | |
|---|---|
| | ● VIP: 172.16.20.252 (backup priority 100) |
| R2_CPE_LYON_BAT_A | g2/0.10 :<br>● IP: 172.16.10.254 - /24<br>● VIP: 172.16.10.252 (master priority 110)<br>g2/0.20 :<br>● IP: 172.16.20.254 - /24<br>● VIP: 172.16.20.252 (master priority 110) |

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.

   Le code permettant de créer la config à partir du template jinja2 et du jeux de données (JSON / YAML) doit être placé dans une fonction nommée : create_config_cpe_lyon_batA() qui retournera la config générée pour chaque équipement du bâtiment (routeur r01, routeur r01 et switch esw1).

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

   Nommez les fichiers de config de la manière suivant:
   - R1_CPE_LYON_BAT_A.conf
   - R2_CPE_LYON_BAT_A.conf
   - ESW1_CPE_LYON_BAT_A.conf

   Le code permettant de créer un fichier de conf et de sauvegarder le résultat de la fonction de la question 3 doit être placé dans une fonction nommée : save_built_config() qui prend en paramètre la config générée et le nom du fichier.

Faites vérifier le résultat par votre responsable pédagogique

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

| Hostname | Ip address |
|---|---|
| R1_CPE_LYON_BAT_B | g2/0.10 :<br>● IP: 172.16.30.253 - /24<br>● VIP: 172.16.30.252 (backup priority 100)<br><br>g2/0.20 :<br>● IP: 172.16.40.253 - /24<br>● VIP: 172.16.40.252 (backup priority 100) |
| R2_CPE_LYON_BAT_B | g2/0.10 :<br>● IP: 172.16.30.254 - /24<br>● VIP: 172.16.30.252 (master priority 110)<br><br>g2/0.20 :<br>● IP: 172.16.40.254 - /24<br>● VIP: 172.16.40.252 (master priority 110) |

Faites vérifier le résultat par votre responsable pédagogique

A ce stade le __main__ de votre script python ne doit contenir que les appels aux fonctions suivantes:

- create_config_cpe_lyon_batA()
- create_config_cpe_lyon_batB()
- save_built_config()

Aucune configuration ne doit être implémentée sur les équipements pour le moment. Nous reviendrons sur ces éléments de configurations un peu plus tard dans le TP.

## 1.9 -Automatisation réseau avec Nornir

Nornir est un framework d'automatisation réseau qui se différencie des autres librairies parcourues tout au long du TP02 (Netmiko, Napalm) car il fournit une abstraction pour l'inventaire ( les hosts et les groupes d'hosts), l'exécution des tâches simultanées  (nous n'avons pas besoin d'écrire nos propres threads) et se veut être flexible  / extensible par l'intégration et l'écriture de plug-ins .

6) Installez nornir dans votre workspace (TP03):  **pipenv install nornir**

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 :

   ```
   from nornir import InitNornir
   ```

Nornir peut être initialisée de trois manières différentes :

1. Depuis un fichier de config YAML
2. Depuis le code python (manuellement)
3. Depuis un fichier de config YAML et le code python

Nous allons utiliser la première méthode , car c'est ce qui fait sa force par rapport aux autres librairies et c'est la méthode la plus propre pour initialiser l'environnement Nornir.

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

```yaml
inventory:
  plugin: SimpleInventory
  options:
    host_file: "inventory/hosts.yaml"
    group_file: "inventory/groups.yaml"
    defaults_file: "inventory/defaults.yaml"

runner:
  plugin: threaded
  options:
    num_workers: 20
```

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

```yaml
R1-CPE-BAT-A:
  hostname: 172.16.100.125
  port: 22
  groups:
    - ios
  data: # Anything under this key is custom data
    device_name: R1-CPE-BAT-A
    device_type: router
    device_model: C7200
    locality: lyon
    building: A

R2-CPE-BAT-A:
  hostname: 172.16.100.126
  port: 22
  groups:
    - ios
  data: # Anything under this key is custom data
    device_name: R2-CPE-BAT-A
    device_type: router
    device_model: C7200
    locality: lyon
    building: A

ESW1-CPE-BAT-A:
  hostname: 172.16.100.123
  port: 22
  groups:
    - ios
  data: # Anything under this key is custom data
    device_name: ESW1-CPE-BAT-A
    device_type: router_switch
    device_model: C3725
    locality: lyon
    building: A

R1-CPE-BAT-B:
  hostname: 172.16.100.189
```

```
  port: 22
  groups:
    - ios
  data: # Anything under this key is custom data
    device_name: R1-CPE-BAT-B
    device_type: router
    device_model: C7200
    locality: lyon
    building: B

R2-CPE-BAT-B:
  hostname: 172.16.100.190
  port: 22
  groups:
    - ios
  data: # Anything under this key is custom data
    device_name: R2-CPE-BAT-B
    device_type: router
    device_model: C7200
    locality: lyon
    building: B
ESW1-CPE-BAT-B:
  hostname: 172.16.100.187
  port: 22
  groups:
    - ios
  data: # Anything under this key is custom data
    device_name: ESW1-CPE-BAT-B
    device_type: router_switch
    device_model: C3725
    locality: lyon
    building: B
```

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

```
ios:
  platform: ios
  data:
    vendor: Cisco
```

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

```
username: cisco
password: cisco
```

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

```
nr = InitNornir(config_file="inventory/config.yaml")
```

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) ?

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 ?

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() )?

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?

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

18) Affichez la valeur de l'attribut "room" crée à la question 17 sur le terminal

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:

```
print(nr.inventory.groups)
```

20) Il est possible d'afficher les groupes rattaché à un host à l'aide de l'attribut groups de l'objet:

```
nr.inventory.hosts.get('R1-CPE-BAT-A').groups
```

21) Par exemple, si on veut afficher les keys définis dans le fichier groups.yaml du host RP1-CPE-BAT-A, il suffit d'exécuter :

```
nr.inventory.hosts.get('R1-CPE-BAT-A').groups[0].keys()
```

22) Affichez le vendor du host R1-CPE-BAT-A en partant du code de la question 21

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. Aidez-vous des questions précédentes.

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.

```
nr.filter(key_example=".....").inventory.hosts.keys()
```

N'hésitez pas à vous aider de la doc : Nornir

25) Affichez à présent la liste des hosts de type router_switch.

Nous avons vu comment initialiser Nornir et comment parcourir notre inventory (hosts, groups, defaults) . L'inventory est une partie importante de Nornir car c'est ce qui fait l'une de ses forces par rapport aux autres librairies que nous avons vu.

Nous allons à présent aborder la notion de Tasks qui sont tout simplement des instructions à exécuter sur un groupe d'hosts (inventory).

Il est nécessaire d'avoir bien compris la partie du TP qui précède cette section pour aborder les tasks dans de bonnes conditions.

Si vous avez des questions ou un doute ou bien encore des zones d'ombres sur les notions précédentes n'hésitez pas à solliciter votre responsable pédagogique avant d'aborder la suite.

26) Importez les classes Task et Result et définissez une première task nommée hello_world

```
from nornir.core.task import Task, Result

def hello_world(task: Task) -> Result:
    return Result(
        host=task.host,
        result=f"{task.host.name} says hello world!"
    )
```

Pour exécuter la task hello_world il vous suffit de faire appel à la méthode run de l'objet nornir:

```
result = nr.run(task=hello_world)
```

Affichez le résultat : print(result)

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

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.

Vous devez installer le package nornir_utils : **pipenv install nornir_utils** et importer print_result depuis ce package:

```
from nornir_utils.plugins.functions import print_result
```

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 ?

30) Faites en sorte que la task hello_world s'exécute uniquement sur les équipements de type router_switch.

Comme décrit dans le cours et dans le TP, Nornir est un framework flexible et extensible. Il a la possibilité d'utiliser des plugins pour faire appels à certaines méthodes de la librairie Netmiko et Napalm (TP-02) pour interagir avec les équipements.

31) Installez les packages nornir_napalm et nornir_netmiko à votre environnement de développement (TP03):

**pipenv install nornir_napalm**
**pipenv install nornir_netmiko**

Importez les méthodes de chaque package:

```
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
```

32) Développez une task permettant d'afficher l'état des interfaces de chaque **routeur** à l'aide de la fonction napalm_cli.

```
result =  nr.run(task=....., commands=["....."])
print_result(result)
```

33) Développez une task permettant d'afficher la table arp de chaque **switch** référencé dans l'inventory. Utilisez la fonction napalm_get pour cela.

```
result =  nr.run(task=....., getters=["....."])
print_result(result)
```

Liste des getters napalm : Napalm doc

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

```
interface loopback 1  R1:  1.1.1.1 255.255.255.255
interface loopback 1  R2:  2.2.2.2 255.255.255.255
```

```
result =  nr.run(task=....., configuration=".....")
print_result(result)

result =  nr.run(task=....., configuration=".....")
print_result(result)
```

35) Développez une fonction contenant une task permettant de sauvegarder la running-config sur les équipements (routeurs / switchs) depuis napalm_cli.

36) Développez une fonction permettant d'afficher l'état des interfaces de chaque **routeur** à l'aide de la task netmiko_send_command.

```
result = nr.run(task=....., command_string="....")
```

```
print_result(result)
```

37) Développez une fonction permettant de créer une interface de loopback 2 sur les routeurs R1 et R2 du bâtiment A . (Une task par routeur) en utilisant la task netmiko_send_config

interface loopback 2 R1: 1.1.1.2 255.255.255.255
interface loopback 2 R2: 2.2.2.3 255.255.255.255

```
result = nr.run(task=....., config_commands=["....",”....”])
print_result(result)
```

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

A ce stade du TP vous êtes en mesure d'utiliser Nornir pour parcourir les hosts de l'inventory et appliquer des filtres de recherche afin d'exécuter des tasks en fonction du type de host ou de l'adresse ip etc.

Vous êtes également en capacité d'interagir avec les équipements en lecture et écriture à l'aide des plugins nornir_netmiko et nornir_napalm.

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.

a) Vérifiez que la communication entre le vlan 10 et 20 du bâtiment A est fonctionnelle après déploiement (même chose pour le bâtiment B)

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.

c) Testez manuellement votre HA vrrp sur chaque bâtiment

Faites vérifier par votre responsable pédagogique

d) Testez automatiquement (via nornir) votre HA vrrp sur chaque bâtiment

Faites vérifier par votre responsable pédagogique

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).