TP12 - Découverte d'un contrôleur de réseau SDN sous Packet Tracer

I/ Comparer la gestion d'un réseau à partir de l'interface en ligne de commande (CLI) et l'utilisation d'un contrôleur de réseau défini par logiciel (SDN)

I/1 Explorer la topologie du réseau

Depuis le PC d'administration (Admin), accéder en toute sécurité au commutateur SWR3

Question : Indique la commande tapée

`ssh -l cisco 10.0.1.4` puis entrer le mot de passe `cisco123!`

I/6 Utiliser un contrôleur SDN pour configurer les paramètres réseau

Question : Comment fonctionnent les commandes ci-dessous et que vont-elles apprendre sur les matériels d'interconnexion

Ces commandes fonctionne sur le CLI d'un routeur ou commutateur de couche 3

— `show run | begin ip domain` Permet de voir la config du domaine DNS

— `show run | begin ip name-` Permet de voir les configurations de nom d'hôte (hostname)

— `show ntp associations` Permet de voir les associations NTP

— `show run | include logging` Permet de voir si la journalisation (Syslog) est configurée

Question : Quelle commande avez-vous tapé ? Quel est le résultat?

```
R1#show ntp associations

address         ref clock       st   when     poll    reach  delay          offset
disp
 ~192.168.101.100127.127.1.1     1    6        16      17     0.00           -755999.00
0.12
 * sys.peer, # selected, + candidate, - outlyer, x falseticker, ~ configured
```

ET

```
R1#show run | begin ip domain
ip domain-name example.com
ip name-server 192.168.101.100
!
!
spanning-tree mode pvst
!
!
!
!
```

```
!
!
interface GigabitEthernet0/0/0
 ip address 192.168.101.1 255.255.255.0
 duplex auto
 speed auto
!
interface GigabitEthernet0/0/1
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface GigabitEthernet0/0/2
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface Serial0/1/0
 ip address 192.168.1.2 255.255.255.0
!
interface Serial0/1/1
 no ip address
 clock rate 2000000
 shutdown
!
interface Vlan1
 no ip address
 shutdown
!
router ospf 1
 log-adjacency-changes
 network 0.0.0.0 255.255.255.255 area 0
!
ip classless
ip route 0.0.0.0 0.0.0.0 Serial0/1/0
!
ip flow-export version 9
!
!
!
no cdp run
!
!
!
!
!
logging 192.168.101.100
line con 0
!
line aux 0
!
line vty 0 4
```

```
 login local
!
!
ntp server 192.168.101.100
!
end
```

Question : Quels sont les inconvénients d'un paramétrage classique en ligne de commande CLI ?

Le paramétrage en CLI est manuel, donc lent, surtout quand il faut répéter les mêmes configurations sur plusieurs équipements. Il augmente le risque d'erreurs humaines (fautes de frappe, oublis) et ne permet pas une gestion centralisée ou automatisée du réseau.

Question : Quels sont les avantages de l'utilisation du contrôleur SDN pour le paramétrage des matériels?

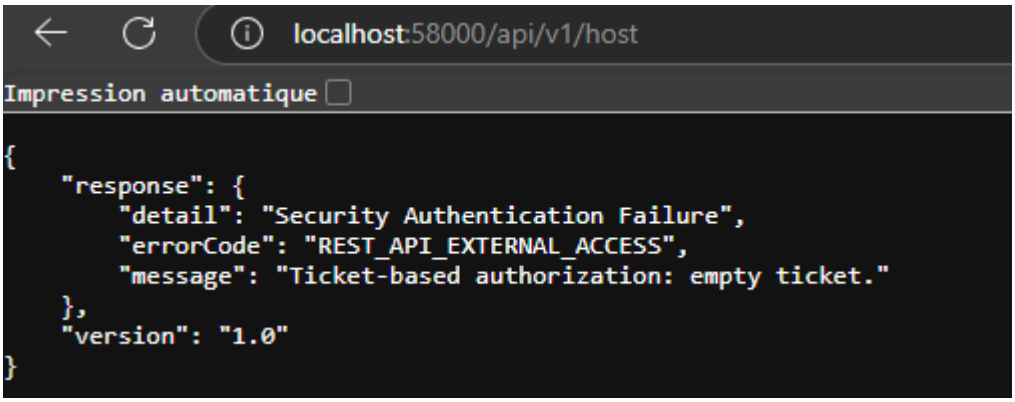
Le contrôleur SDN permet de configurer tous les équipements réseau depuis une interface centralisée. Cela rend les déploiements plus rapides, cohérents et automatisés. On peut appliquer une stratégie réseau globale en un clic, sans avoir à intervenir manuellement sur chaque appareil.

Question : Quel(s) type(s) d'erreur(s) ce déploiement évite-il ?

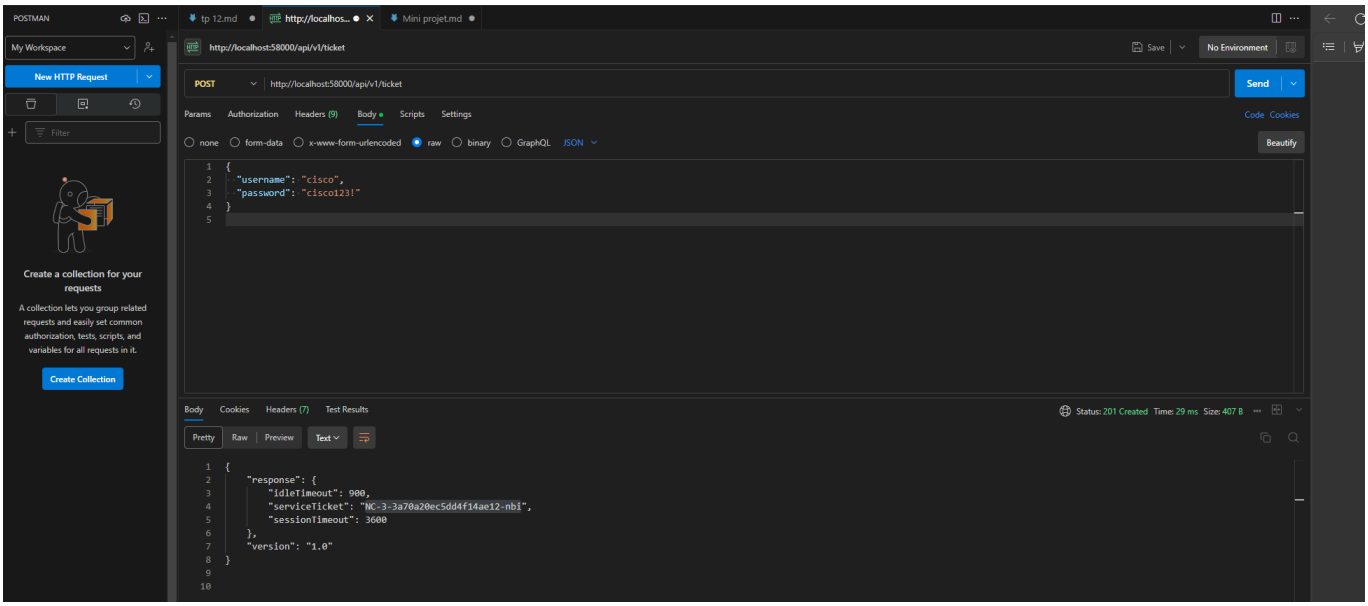
Il évite les erreurs de saisie (mauvaises commandes), les oublis de configuration sur certains équipements, et les incohérences entre les appareils. Il garantit que tous les équipements reçoivent exactement les mêmes paramètres.

II/1 Vérifier la connectivité externe avec Packet Tracer

Ouvrir le navigateur et accéder à http 😕/localhost :58000/api/v1/host



Créer une nouvelle demande POST

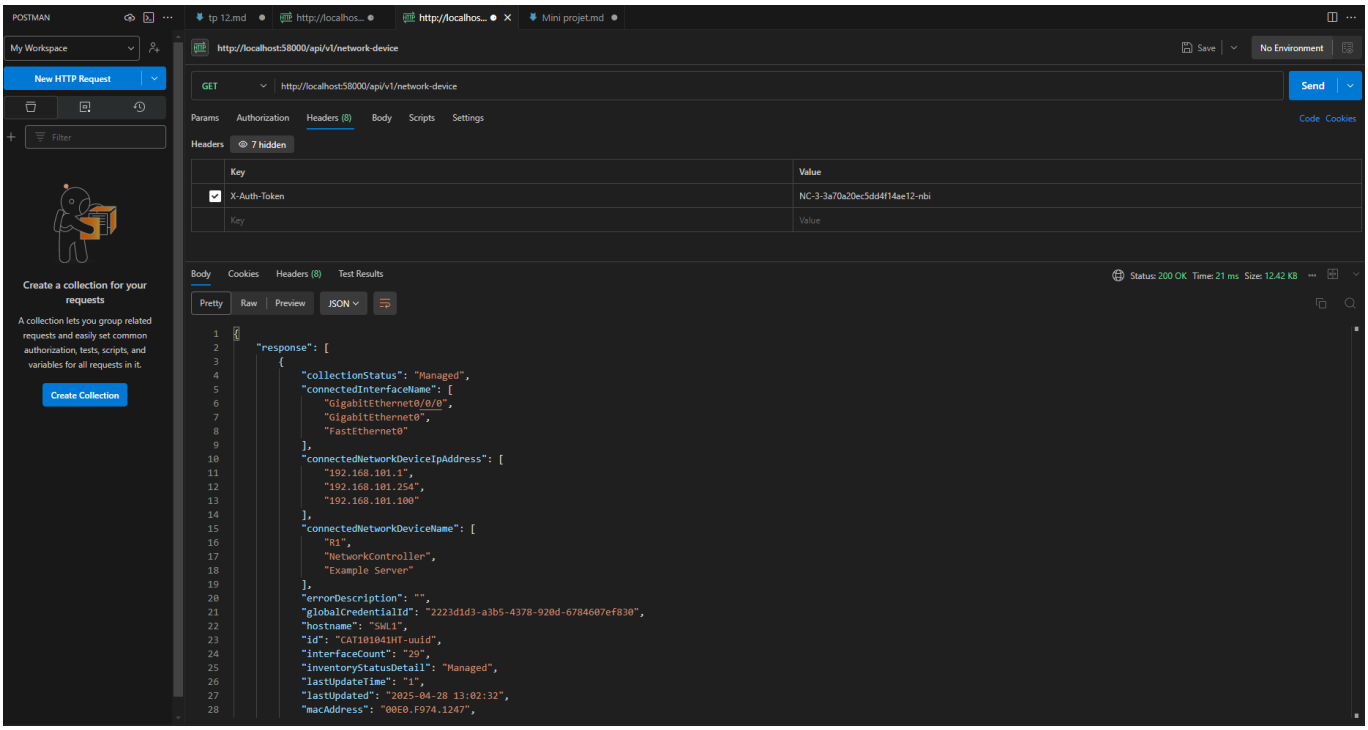


Question : Sachant que l'obtention d'un jeton permet l'accès par API au contrôleur et peut être conditionné à un mot de passe et limité dans le temps, quels en sont les avantages :

L'utilisation de tickets permet un accès plus sécurisé au contrôleur SDN, avec une expiration automatique possible et une limitation dans le temps, ce qui renforce le contrôle des accès API.

II/3 Envoyer des demandes REST avec Postman

Créer une nouvelle requête GET pour tous les périphériques réseau du réseau



III/ Requêtes REST en Python

Se placer sur 01_get-ticket.py et lancer le programme (penser à récupérer le numéro de ticket)

```python
import json
import requests
api_url = "http://localhost:58000/api/v1/ticket"

headers = {
    "content-type": "application/json"
}

body_json = {
    "username": "cisco",
    "password": "cisco123!"
}

resp = requests.post(api_url, json.dumps(body_json), headers=headers, verify=False)

print("Ticket request status: ", resp.status_code)
response_json = resp.json()

serviceTicket = response_json["response"]["serviceTicket"]
print("The service ticket number is: ", serviceTicket)

```

```
PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE

Successfully installed certifi-2025.4.26 charset-normalizer-3.4.1 idna-3.10 requests-2.32.3 urllib3-2.4.0

[notice] A new release of pip is available: 24.2 -> 25.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\ADAOUAIRI> & C:/Python312/python.exe c:/Users/ADAOUAIRI/Downloads/01_get-ticket_base.py
Ticket request status:  201
The service ticket number is:  NC-4-a8af5bb5747d4eafbf13-nbi
```

Se placer sur 02_get-network-device.py, le compléter et le lancer

Question : Quel est le résultat obtenu ?

```python
import json
import requests
api_url = "http://localhost:58000/api/v1/network-device"

headers={"X-Auth-Token": "NC-4-a8af5bb5747d4eafbf13-nbi"}

resp = requests.get(api_url, headers=headers, verify=False)

print("Request status: ", resp.status_code)

response_json = resp.json()
networkDevices = response_json["response"]

for networkDevice in networkDevices:
    print(networkDevice["hostname"], "\t", networkDevice["platformId"], "\t", networkDevice["managementIpAddress"])
```

Le résultat est :

```
Request status:  200
SWL1     3650    192.168.101.2
R1       ISR4300         192.168.1.2
R3       ISR4300         192.168.2.1
R2       ISR4300         192.168.2.2
SWR1     3650    10.0.1.2
SWR2     3650    10.0.1.3
SWL2     3650    192.168.102.2
```

```
SWR4      3650     10.0.1.5
SWR3      3650     10.0.1.4
```

Se placer sur 03_get-host.py, le compléter et le lancer

Question : Quel est le résultat obtenu ?

Le script compléter :

```python
import json
import requests
api_url = "http://localhost:58000/api/v1/host"

headers={"X-Auth-Token": "NC-4-a8af5bb5747d4eafbf13-nbi"}

resp = requests.get(api_url, headers=headers, verify=False)

print("Request status: ", resp.status_code)

response_json = resp.json()
hosts = response_json["response"]

for host in hosts:
    print(host["hostName"], "\t", host["hostIp"], "\t", host["hostMac"], "\t", host["connectedInterfaceName"])

```

Le résultat obtenu est :

```
Request status:  200
Example Server   192.168.101.100         000A.4113.C0B0
GigabitEthernet1/0/3
PC4      192.168.102.3   0001.435B.5044          GigabitEthernet1/0/24
PC2      10.0.2.129      0060.700B.2BC5          GigabitEthernet1/0/23
PC3      10.0.2.130      0050.0F6E.234D          GigabitEthernet1/0/24
Admin    10.0.1.130      000B.BE16.D6BA          GigabitEthernet1/0/21
PC1      10.0.1.129      0001.9747.D29B          GigabitEthernet1/0/22
```

Question : Est-il possible de récupérer le résultat des deux programmes précédents et de le rendre exploitable? Donnez un exemple d'exploitation de ces résultats?

Oui il est possible de récupérer les résultats des deux programmes `(02_get-network-device.py et 03_get-host.py)` et de les exploiter ensemble.