

SAE D03

SITUATION PROFESSIONNELLE

Vous êtes technicien dans une équipe en charge des réseaux d'une grande entreprise. L'entreprise gère une AS BGP privée pour chacune de ses filiales et vous êtes responsable d'une de ces AS.

TABLE DES MATIÈRES

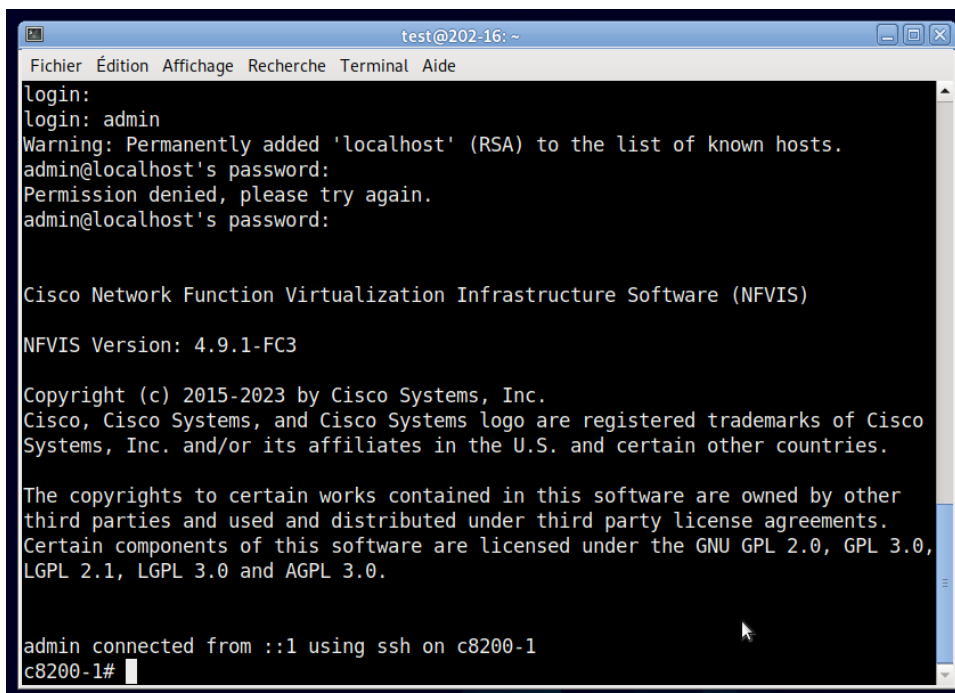
1. Mise en place de la SAE
2. Mise en place des configurations
3. Preuves de bon fonctionnement
4. Utilisation des différents protocoles d'administrations

MISE EN PLACE DE LA SAE

1. Initialisation
2. Mise à jour
3. Limitation et solution
4. Mise en place des machines virtuelles

INITIALISATION

Comme tout routeur Cisco l'initialisation débute avec la connexion de notre ordinateur au routeur par port série, pour ce faire, nous avons utilisé le logiciel `minicom`, on branche le câble série, on paramètre minicom avec le bon port série (dans notre cas `/dev/ttyS0` qui est trouvable avec la commande `sudo dmesg | grep tty`) et le bon débit (dans notre cas `9600 8N1`). Une fois connecté, on arrive à cela :



```
test@202-16: ~  
Fichier  Édition  Affichage  Recherche  Terminal  Aide  
login:  
login: admin  
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.  
admin@localhost's password:  
Permission denied, please try again.  
admin@localhost's password:  
  
Cisco Network Function Virtualization Infrastructure Software (NFWIS)  
  
NFWIS Version: 4.9.1-FC3  
  
Copyright (c) 2015-2023 by Cisco Systems, Inc.  
Cisco, Cisco Systems, and Cisco Systems logo are registered trademarks of Cisco  
Systems, Inc. and/or its affiliates in the U.S. and certain other countries.  
  
The copyrights to certain works contained in this software are owned by other  
third parties and used and distributed under third party license agreements.  
Certain components of this software are licensed under the GNU GPL 2.0, GPL 3.0,  
LGPL 2.1, LGPL 3.0 and AGPL 3.0.  
  
admin connected from ::1 using ssh on c8200-1  
c8200-1#
```

Étant connecté sur l'interface série, on peut mettre une IP, et un mot de passe au C8200. Cette IP mise, on peut maintenant réaliser les configurations sur l'interface WEB, de même en SSH.

MISE À JOUR

Selon la documentation Cisco, il existe des restrictions :

Restrictions for Cisco NFWIS ISO File Upgrade

Cisco NFWIS supports .iso upgrade only from version N to versions N+1 and N+2. NFWIS does not support .iso upgrade from version N to version N+3 and above.

On sait donc qu'avec notre version de base en 4.5.1 on pourra aller qu'à N+2, selon le tableau de la même documentation, on doit passer par la 4.6.1 puis la 4.7.1 et pour finir la 4.9.1, pour réaliser ces mises à jour, on va dans l'onglet **Operation** ensuite dans **Upgrade**. Ici, on ajoute l'image de mise à jour, ensuite, une fois envoyée, on peut effectuer les mises à jour unes par unes. Elles durent en moyenne entre 20 et 30 minutes.



LIMITATION ET SOLUTION

Un problème qui arrive assez rapidement avec le C8200 d'origine est son espace de stockage, il est assez limité, tout particulièrement quand on fait tourner plusieurs machines virtuelles.

Pour régler ce problème, on peut supprimer des images gardées en cache dans la mémoire du C8200. Pour ce faire, il existe deux commandes, la première est la suivante, qui permet de lister tous les fichiers du C8200, pour voir lesquels on peut supprimer.

```
show system file-list
```

Ce qui donne :

SI NO	NAME	PATH	SIZE	TYPE	DATE MODIFIED
1	Cisco NfVIS-4.9.1-FC3.iso	/data/intdatastore/uploads	1.8G	VM Package	2023-01-04 16:07:05
2	frr-8.2.2.qcow2	/data/intdatastore/uploads	54M	VM Package	2023-01-05 10:47:10
3	c8000v-universalk9_16G_serial.17.06.01a.tar.gz	/data/intdatastore/uploads	1.5G	VM Package	2023-01-05 07:57:43
4	nfvisvmpackagingtool.tar	/data/intdatastore/uploads/vmpackagingutility	370K	VM Packaging Tool	2023-01-04 16:14:48
5	nfvis shell.log	/data/intdatastore/logs	0	Log File	2023-01-10 07:55:42
6	nfvis scp.log	/data/intdatastore/logs	0	Log File	2023-01-10 07:55:42
7	escmanager_tagged.log-20230110.gz	/data/intdatastore/logs/2023-01	151K	Log File	2023-01-10 09:00:05
8	access.log-20230109.gz	/data/intdatastore/logs/2023-01	33K	Log File	2023-01-09 07:42:33
9	messages-20230109.gz	/data/intdatastore/logs/2023-01	187K	Log File	2023-01-09 17:00:01
10	confd_audit.log-20230106.gz	/data/intdatastore/logs/2023-01	1.7M	Log File	2023-01-06 17:00:03
11	escdatabase.log-20230106.gz	/data/intdatastore/logs/2023-01	280K	Log File	2023-01-06 16:15:03
12	debug_yangesc.log-20230105.gz	/data/intdatastore/logs/2023-01	2.1M	Log File	2023-01-05 17:30:01
13	php.log-20230109.gz	/data/intdatastore/logs/2023-01	213K	Log File	2023-01-09 20:00:02
14	escmanager_tagged.log-20230109.gz	/data/intdatastore/logs/2023-01	179K	Log File	2023-01-09 15:45:02
15	access.log-20230104.gz	/data/intdatastore/logs/2023-01	35K	Log File	2023-01-04 14:00:07
16	escmanager_tagged.log-20230106.gz	/data/intdatastore/logs/2023-01	193K	Log File	2023-01-06 13:30:03
17	confd_audit.log-20230104.gz	/data/intdatastore/logs/2023-01	1.8M	Log File	2023-01-04 17:30:02
18	messages-20230104.gz	/data/intdatastore/logs/2023-01	142K	Log File	2023-01-04 15:30:02
19	lastlog-20230109.gz	/data/intdatastore/logs/2023-01	5.1K	Log File	2023-01-09 17:00:03
20	nfvis_config.log-20230109.gz	/data/intdatastore/logs/2023-01	145K	Log File	2023-01-09 16:50:28
21	server.log-20230110.gz	/data/intdatastore/logs/2023-01	25K	Log File	2023-01-10 08:59:49
22	lastlog-20230106.gz	/data/intdatastore/logs/2023-01	2.6K	Log File	2023-01-06 07:12:19
23	debug_yangesc.log-20230104.gz	/data/intdatastore/logs/2023-01	47K	Log File	2023-01-04 17:00:01
24	debug_yangesc.log-20230109.gz	/data/intdatastore/logs/2023-01	8.9M	Log File	2023-01-09 23:45:01
25	confd_audit.log-20230105.gz	/data/intdatastore/logs/2023-01	1.6M	Log File	2023-01-05 17:30:01
26	escdatabase.log-20230105.gz	/data/intdatastore/logs/2023-01	122K	Log File	2023-01-05 13:00:02
27	escdatabase.log-20230109.gz	/data/intdatastore/logs/2023-01	501K	Log File	2023-01-09 17:00:03
28	escdatabase.log-20230110.gz	/data/intdatastore/logs/2023-01	193K	Log File	2023-01-10 09:00:05
29	escmanager.log-20230110.gz	/data/intdatastore/logs/2023-01	94K	Log File	2023-01-10 08:45:03
30	access.log-20230110.gz	/data/intdatastore/logs/2023-01	34K	Log File	2023-01-10 08:45:02
31	debug_yangesc.log-20230110.gz	/data/intdatastore/logs/2023-01	3.7M	Log File	2023-01-10 09:15:12
32	lastlog-20230110.gz	/data/intdatastore/logs/2023-01	2.6K	Log File	2023-01-10 07:58:40
33	confd_audit.log-20230109.gz	/data/intdatastore/logs/2023-01	2.6M	Log File	2023-01-09 23:30:02
34	lastlog-20230104.gz	/data/intdatastore/logs/2023-01	21K	Log File	2023-01-04 17:00:02

Une fois les fichiers que l'on souhaite sélectionner, on peut les supprimer avec la commande suivante :

```
system file-delete file name [chemien complet]
#exemple :
system file-delete file name /data/intdatastore/uploads/centosvm.ova
```

MISE EN PLACE DES MACHINES VIRTUELLES

Pour mettre en place les machines virtuelles, il faut se rendre dans Configuration > Virtual Machine > Images > Image Repository. Une fois sur cette interface, on peut importer nos machines virtuelles.

Une fois les images, ova, tar.gz,... on peut se rendre dans l'onglet Configuration > Deploy. Sur cette interface, on va déployer nos deux routeurs et notre machine virtuelle, une fois déployé, il faut créer les réseaux pour les connecter, pour cela, on va dans Configuration > Virtual Machine > Networking > Bridge où on va créer des bridges que nous n'allons pas connecter à une interface. Dans notre cas, il s'appelle BR1 :

Bridges

Bridge Information and Configuration

Total Record: 8

🔍 search in all record

#	Bridge Name	IP Address	Netmask	Vlan	Interface	Action
1	wan-br				GEO-0	✎ ⚙ 🗑
2	wan2-br				GEO-1	✎ ⚙ 🗑
3	lan-br	10.202.17.100	255.255.0.0		GEO-2	✎ ⚙ 🗑
4	cellular-br				int-CELL-1-0	✎ ⚙ 🗑
5	BR1					✎ ⚙ 🗑

Le bridge créé, on peut se rendre dans Configuration > Virtual Machine > Networking > Networks pour créer nos deux réseaux, qui seront réalisés au bridge précédemment créé, afin de pouvoir connecter nos machines virtuelles. Dans notre cas NET-1 et NET-2 :

Networks

Networks Information and Configuration

DPDK

Disabled

Enabled

Total Record: 5

🔍 search in all record

#	Network	Mode	Vlan	Vlan-Range	Native Vlan	Bridge	Interface	Action
1	wan-net	trunk				wan-br	GEO-0	✎ 🗑
2	wan2-net	trunk				wan2-br	GEO-1	✎ 🗑
3	lan-net	trunk				lan-br	GEO-2	✎ 🗑
4	NET1	trunk				BR1		✎ 🗑
5	NET-2	trunk				BR1		✎ 🗑

Les machines en places, reliées et allumées, nous sommes prêts à les configurer.

MISE EN PLACE DES CONFIGURATIONS DES MACHINE VIRTUELLES

Une fois les machines en places, on a trois méthodes pour les configurer, en SSH directement si elles ont une IP, en console depuis l'interface WEB ou en console depuis l'interface CLI (en SSH) du C8200.

Pour le Cisco, je me suis connecté en premier lieu au C8200 en SSH avec la commande `ssh -o HostKeyAlgorithms=+ssh-rsa admin@10.202.17.100` puis une fois dessus, j'ai pu lister les machines virtuelles qui tournaient dessus avec la commande `show system deployments`.

```
mathieu@mathieu-pc ~ ssh -o HostKeyAlgorithms=+ssh-rsa admin@10.202.17.100
admin@10.202.17.100 s password:
```

Cisco Network Function Virtualization Infrastructure Software (NFVIS)

NFVIS Version: 4.9.1-FC3

Copyright (c) 2015-2023 by Cisco Systems, Inc.

Cisco, Cisco Systems, and Cisco Systems logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

The copyrights to certain works contained in this software are owned by other third parties and used and distributed under third party license agreements.

Certain components of this software are licensed under the GNU GPL 2.0, GPL 3.0, LGPL 2.1, LGPL 3.0 and AGPL 3.0.

admin connected from 10.202.0.170 using ssh on c8200-1

c8200-1# show system deployments

NAME	ID	STATE	TYPE
------	----	-------	------

c8000v1	1	running	vm
---------	---	---------	----

ROUTER53	3	running	vm
----------	---	---------	----

DEBIAN	5	running	vm
--------	---	---------	----

Une fois la machine repérée, j'ai pu m'y connecter avec la commande `vmConsole [nom de la vm]`.

```
c8200-1# vmConsole c8000v1
```

```
Connected to domain c8000v1
```

```
Escape character is ^]
```

```
c8000v-math>
```

Une fois dans le Cisco c8000v, j'ai configuré les différentes IP de mes réseaux, mon BGP, mes voisins etc... Voici un extrait de ma configuration du Cisco c8000v dans lequel on peut voir les éléments principaux de la configuration finale.

```
interface Loopback1
description loopback
ip address 1.1.1.1 255.255.255.255
shutdown
!
interface Loopback2
description La loopback deux retour
ip address 5.5.5.5 255.255.255.255
shutdown
!
interface GigabitEthernet1
vrf forwarding Mgmt-intf
ip address 10.20.0.2 255.255.255.0
negotiation auto
no mop enabled
no mop sysid
!
interface GigabitEthernet2
ip address 10.202.17.200 255.255.0.0
negotiation auto
no mop enabled
no mop sysid
!
interface GigabitEthernet4
ip address 192.168.2.1 255.255.255.0
negotiation auto
no mop enabled
no mop sysid
!
router bgp 65200
bgp log-neighbor-changes
neighbor 10.202.18.101 remote-as 65100
neighbor 192.168.2.2 remote-as 65200
!
address-family ipv4
network 10.202.0.0
network 192.168.2.0
redistribute connected
redistribute static
neighbor 10.202.18.101 activate
neighbor 192.168.2.2 activate
exit-address-family
!
ip forward-protocol nd
ip http server
ip http authentication local
ip http secure-server
!
ip route 0.0.0.0 0.0.0.0 10.202.255.254
```

```
ip route vrf Mgmt-intf 0.0.0.0 0.0.0.0 10.20.0.1
ip ssh rsa keypair-name ssh-key
ip ssh version 2
!
snmp-server community public RO
!
netconf-yang
restconf
end
```

Configuration complète dans confc8000v.txt

Pour le FRR, j'ai réalisé la configuration depuis la console WEB, pour cela, je me suis rendu dans Configuration > Virtual Machine > Manage et je clique sur **Terminal**. Une fois sur la console, je peux, comme pour le Cisco, configurer mes différentes interfaces, réseaux, configurations BGP, etc... Voici ce que donne ma configuration finale :

```
Current configuration:
!
frr version 8.2.2
frr defaults traditional
hostname frr
service integrated-vtysh-config
!
interface eth0
    ip address 192.168.2.2/24
exit
!
interface eth1
    ip address 192.168.20.1/24
exit
!
interface lo
    ip address 2.2.2.2/32
    shutdown
exit
!
router bgp 65200
    neighbor 192.168.2.1 remote-as 65200
    !
    address-family ipv4 unicast
        network 192.168.2.0/24
        network 192.168.20.0/24
    exit-address-family
exit
!
end
```

Configuration complète dans confFRR.png

En ce qui concerne la machine virtuelle Debian, rien de très compliqué, il suffit de configurer l'interface réseau.

```

Debian GNU/Linux 10 debian10 tty1

-----
                        LINUXVMIMAGES.COM
-----

Host Name: debian10
Time: Thu Jan 12 2023 09:05:44
System IP address: 127.0.1.1
User Name: debian
Password:  debian (sudo su -)

debian10 login: debian
Password:
Last login: Tue Jan 10 03:47:03 CST 2023 on tty1
Linux debian10 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

-----
                        LINUXVMIMAGES.COM
-----

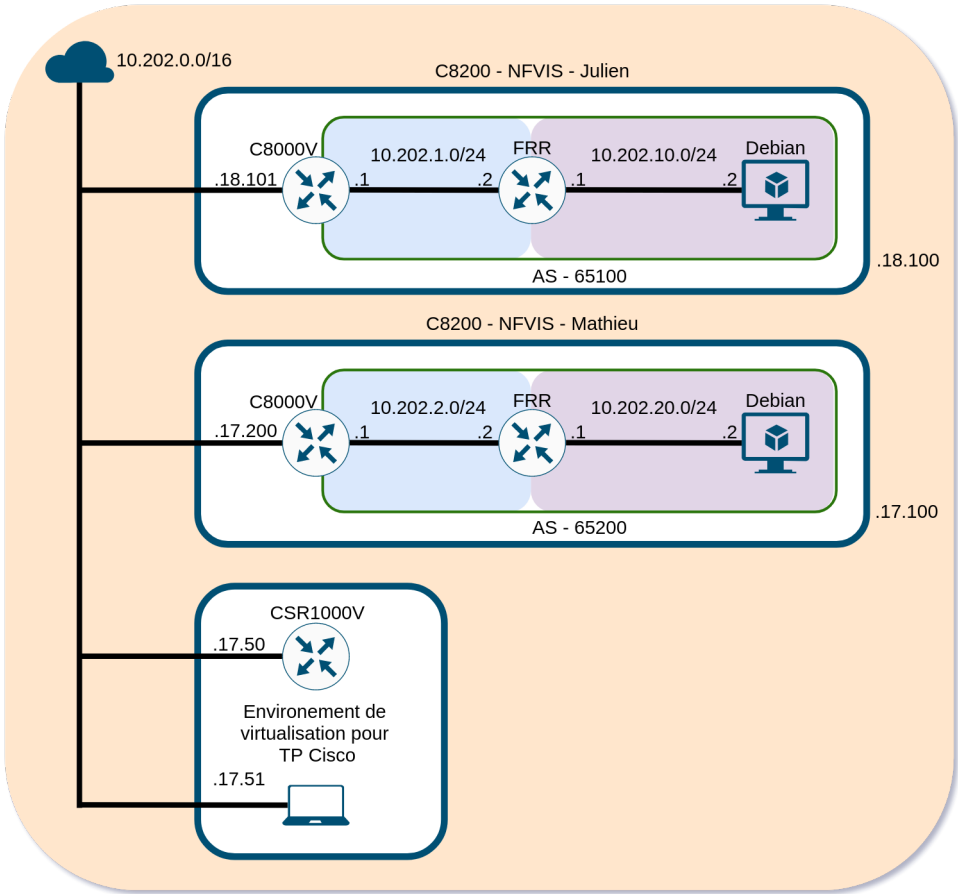
User Name: debian
Password:  debian (sudo su -)

debian@debian10:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 52:54:00:42:a8:34 brd ff:ff:ff:ff:ff:ff
debian@debian10:~$ sudo ip addr add 192.168.20.2/24 dev ens4
debian@debian10:~$ sudo ip route add default via 192.168.20.1
Error: NextHop has invalid gateway.
debian@debian10:~$ sudo ip link set ens4 up
debian@debian10:~$ sudo ip route add default via 192.168.20.1
debian@debian10:~$ ip -br a
lo                UNKNOWN      127.0.0.1/8 ::1/128
ens4              UP          192.168.20.2/24 fe80::5054:ff:fe42:a834/64
debian@debian10:~$

```


BON FONCTIONNEMENT

A ce stade, notre réseau ressemble à cela (avec l'environnement de TP Cisco pris en compte) :



Niveau NFVIS :



Du côté des routeurs les échanges des routes BGP, on était effectué.

Pour le Cisco :

```

c8000v-math#sh bgp
% Command accepted but obsolete, unreleased or unsupported; see documentation.

BGP table version is 14, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
               x best-external, a additional-path, c RIB-compressed,
               t secondary path, L long-lived-stale,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network        Next Hop           Metric LocPrf Weight Path
*    10.202.0.0/16   10.202.18.101             0           0 65100 ?
*>    0.0.0.0         0.0.0.0                   0           32768 ?
*>   192.168.1.0     10.202.18.101             0           0 65100 i
* i  192.168.2.0     192.168.2.2               0          100      0 i
*>    0.0.0.0         0.0.0.0                   0           32768 i
*>   192.168.10.0    10.202.18.101             0           0 65100 i
*> i 192.168.20.0    192.168.2.2               0          100      0 i

c8000v-math#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
       n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       H - NHRP, G - NHRP registered, g - NHRP registration summary
       o - ODR, P - periodic downloaded static route, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR
       & - replicated local route overrides by connected

Gateway of last resort is 10.202.255.254 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 10.202.255.254
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C      10.202.0.0/16 is directly connected, GigabitEthernet2
L      10.202.17.200/32 is directly connected, GigabitEthernet2
B      192.168.1.0/24 [20/0] via 10.202.18.101, 2d07h
      192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C      192.168.2.0/24 is directly connected, GigabitEthernet4
L      192.168.2.1/32 is directly connected, GigabitEthernet4
B      192.168.10.0/24 [20/0] via 10.202.18.101, 2d06h
B      192.168.20.0/24 [200/0] via 192.168.2.2, 2d05h
c8000v-math#

```

Pour le FRR :

```
frr# sh bgp all

For address family: IPv4 Unicast
BGP table version is 13, local router ID is 192.168.2.2, vrf id 0
Default local pref 100, local AS 65200
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: 0NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: U valid, I invalid, N Not found

   Network          Next Hop          Metric LocPrf Weight Path
*>i10.202.0.0/16     192.168.2.1          0      100      0 ?
*>i192.168.1.0/24    10.202.18.101        0      100      0 65100 i
* i192.168.2.0/24    192.168.2.1          0      100      0 i
*>                   0.0.0.0              0              32768 i
*>i192.168.10.0/24   10.202.18.101        0      100      0 65100 i
*> 192.168.20.0/24   0.0.0.0              0              32768 i

Displayed 5 routes and 6 total paths
frr# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, U - UNC-Direct, A - Babel, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

B>* 10.202.0.0/16 [200/0] via 192.168.2.1, eth0, weight 1, 2d06h46m
B> 192.168.1.0/24 [200/0] via 10.202.18.101 (recursive), weight 1, 2d06h46m
*                               via 192.168.2.1, eth0, weight 1, 2d06h46m
C>* 192.168.2.0/24 is directly connected, eth0, 2d06h46m
B> 192.168.10.0/24 [200/0] via 10.202.18.101 (recursive), weight 1, 2d06h08m
*                               via 192.168.2.1, eth0, weight 1, 2d06h08m
C>* 192.168.20.0/24 is directly connected, eth1, 2d05h41m
frr#
```

Avec toutes les routes propagées nos machines Debian peuvent communiquer entre elle :

```
debian@debian10:~$ traceroute 192.168.10.2
traceroute to 192.168.10.2 (192.168.10.2), 30 hops max, 60 byte packets
 1  192.168.20.1 (192.168.20.1)  0.294 ms  0.247 ms  0.216 ms
 2  192.168.2.1 (192.168.2.1)  1.218 ms  1.182 ms  1.167 ms
 3  10.202.18.101 (10.202.18.101)  1.197 ms  1.169 ms  1.162 ms
 4  192.168.1.2 (192.168.1.2)  1.284 ms  1.248 ms  1.235 ms
 5  192.168.10.2 (192.168.10.2)  1.450 ms  1.441 ms  1.408 ms
debian@debian10:~$
```

De plus, en preuve de fonctionnement, pour en se connectant au routeur maître, j'ai pu récupérer environ 500 000 routes :

```
Router#
Router#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
        D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
        N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
        E1 - OSPF external type 1, E2 - OSPF external type 2, m - OMP
        n - NAT, Ni - NAT inside, No - NAT outside, Nd - NAT DIA
        i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
        ia - IS-IS inter area, * - candidate default, U - per-user static route
        H - NHRP, G - NHRP registered, g - NHRP registration summary
        o - ODR, P - periodic downloaded static route, l - LISP
        a - application route
        + - replicated route, % - next hop override, p - overrides from PfR
        & - replicated local route overrides by connected

Gateway of last resort is not set

1.0.0.0/8 is variably subnetted, 2486 subnets, 15 masks
B 1.0.0.0/24 [20/1] via 10.202.0.52, 00:03:40
B 1.0.4.0/24 [20/0] via 10.202.0.52, 00:03:40
B 1.0.5.0/24 [20/0] via 10.202.0.52, 00:03:40
B 1.0.6.0/24 [20/0] via 10.202.0.52, 00:03:40
B 1.0.7.0/24 [20/1] via 10.202.0.52, 00:03:41
B 1.0.38.0/24 [20/0] via 10.202.0.52, 00:03:41
B 1.0.64.0/18 [20/1] via 10.202.0.52, 00:03:41
B 1.0.128.0/17 [20/1] via 10.202.0.52, 00:03:41
B 1.0.128.0/18 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.128.0/19 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.128.0/24 [20/0] via 10.202.0.52, 00:03:41
B 1.0.129.0/24 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.131.0/24 [20/0] via 10.202.0.52, 00:03:41
B 1.0.139.0/24 [20/0] via 10.202.0.52, 00:03:41
B 1.0.142.0/24 [20/0] via 10.202.0.52, 00:03:41
B 1.0.160.0/19 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.160.0/21 [20/0] via 10.202.0.52, 00:03:41
B 1.0.192.0/18 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.192.0/19 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.192.0/21 [20/1] via 10.202.0.52, 00:03:41
B 1.0.208.0/22 [20/0] via 10.202.0.52, 00:03:41
B 1.0.212.0/23 [20/0] via 10.202.0.52, 00:03:41
B 1.0.214.0/24 [20/0] via 10.202.0.52, 00:03:41
B 1.0.224.0/19 [20/26500] via 10.202.0.52, 00:03:41
B 1.0.240.0/20 [20/0] via 10.202.0.52, 00:04:00
B 1.1.1.0/24 [20/1] via 10.202.0.52, 00:04:00
B 1.1.8.0/24 [20/0] via 10.202.0.52, 00:04:00
B 1.1.32.0/24 [20/0] via 10.202.0.52, 00:04:00
B 1.1.64.0/19 [20/1] via 10.202.0.52, 00:04:00
B 1.1.114.0/24 [20/1] via 10.202.0.52, 00:04:00
B 1.1.115.0/24 [20/1] via 10.202.0.52, 00:04:00
B 1.1.116.0/24 [20/1] via 10.202.0.52, 00:04:00
B 1.1.117.0/24 [20/1] via 10.202.0.52, 00:04:00
B 1.1.118.0/24 [20/1] via 10.202.0.52, 00:04:00
B 1.1.119.0/24 [20/1] via 10.202.0.52, 00:04:00
```

UTILISATION DES DIFFÉRENTS PROTOCOLES D'ADMINISTRATION SUR LE C800V

1. SNMP
2. NETCONF
3. RESTCONF

SNMP

J'ai mis en place le SNMP sur mon routeur Cisco. Il est en public et en lecture seule, ce qui me permet de récupérer toutes les informations que je veux.

Du côté routeur, j'active le serveur avec la commande `snmp-server community public RO`.

Sur mon client, j'ai installé les paquets `snmp` et `snmp-mibs-downloader` et j'ai effectué la commande `sudo wget http://pastebin.com/raw.php?i=p3QyuXzZ -O /usr/share/snmp/mibs/ietf/SNMPv2-PDU` pour enlever les possibles erreurs.

Après cela, j'ai donc pu communiquer avec mon routeur en SNMP :

```
mathieu@mathieu-pc ➜ snmpwalk -v1 -c public 10.202.17.200 -Cp -Ct | grep 8000
SNMPv2-MIB::sysName.0 = STRING: c8000v-math
IF-MIB::ifHighSpeed.6 = Gauge32: 8000
IF-MIB::ifHighSpeed.7 = Gauge32: 8000
ENTITY-MIB::entPhysicalDescr.1 = STRING: Cisco Catalyst 8000V Edge Chassis
ENTITY-MIB::entPhysicalDescr.10 = STRING: Cisco Catalyst 8000V Edge Route Processor
ENTITY-MIB::entPhysicalDescr.30 = STRING: Cisco Catalyst 8000V Edge Embedded Services Processor
ENTITY-MIB::entPhysicalDescr.50 = STRING: Cisco Catalyst 8000V Edge NIM controller
ENTITY-MIB::entPhysicalModelName.1 = STRING: C8000V
ENTITY-MIB::entPhysicalModelName.10 = STRING: C8000V
ENTITY-MIB::entPhysicalModelName.30 = STRING: C8000V
ENTITY-MIB::entPhysicalModelName.50 = STRING: C8000V
ENTITY-MIB::entLogicalDescr.1 = STRING: default logical entity for C8000V platform
Total traversal time = 3.625767 seconds
mathieu@mathieu-pc ➜ snmpwalk -v1 -c public 10.202.17.200 -Cp -Ct | grep ipAddressPrefix.ipv4
IP-MIB::ipAddressPrefix.ipv4."1.1.1.1" = OID: IP-MIB::ipAddressPrefixOrigin.6.ipv4."1.1.1.1".32
IP-MIB::ipAddressPrefix.ipv4."5.5.5.5" = OID: IP-MIB::ipAddressPrefixOrigin.7.ipv4."5.5.5.5".32
IP-MIB::ipAddressPrefix.ipv4."10.20.0.2" = OID: IP-MIB::ipAddressPrefixOrigin.1.ipv4."10.20.0.0".24
IP-MIB::ipAddressPrefix.ipv4."10.202.17.200" = OID: IP-MIB::ipAddressPrefixOrigin.2.ipv4."10.202.0.0".16
IP-MIB::ipAddressPrefix.ipv4."192.168.2.1" = OID: IP-MIB::ipAddressPrefixOrigin.3.ipv4."192.168.2.0".24
Total traversal time = 3.608827 seconds
mathieu@mathieu-pc ➜
```

NETCONF

Netconf utilise le modèle de données YANG pour communiquer avec les appareils réseaux. Yang est un langage de modélisation de données.

Côté routeur

Activation avec la commande `netconf-yang` dans le mode configuration terminal du routeur.

Pour obtenir le nom du `datasore dnetconf-yang` il existe la commande `show netconf-yang datastores`.

Version CLI

Voici un exemple d'utilisation CLI de netconf.

Connexion au routeur :

```
ssh admin1@10.202.17.200 -p 830 -s netconf
```

Réaliser le handshake :

```
<?xml version="1.0" encoding="UTF-8"?><hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><capabilities>  
<capability>urn:ietf:params:netconf:base:1.0</capability></capabilities></hello>]]>]]>
```

Pour récupérer toutes les interfaces :

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><get><filter><interfaces  
xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/></filter></get></rpc>]]>]]>
```

Et finalement pour fermer la session :

```
<rpc message-id="9999999" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  
<close-session/></rpc>]]>]]>
```

Côté client

Côté du client (administrateur), on peut utiliser du python pour réaliser notre administration. (Le faire manuellement étant bien trop fastidieux)

Initialisation du code

Code pour initier la connexion :

```
from ncclient import manager
import xml.dom.minidom

m = manager.connect (
    host="10.202.17.200",
    port=830,
    username="admin1",
    password="Root123#",
    hostkey_verify=False
)
```

Récupérer la running-config

```
netconf_filter = """
<filter xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native"/>
</filter>
"""

netconf_reply = m.get_config(source="running", filter=netconf_filter) #connexion et récupération de la config du
datastores

print (xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

[Retour du code dans le fichier conf-xml-c800v.xml](#)

Changer le hostname

```
#configuration du nouveaux nom
netconf_hostname = ""

<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <hostname>c8000v-math</hostname>
  </native>
</config>
"""

netconf_reply = m.edit_config(target="running", config=netconf_hostname) #edition de la configuration

print (xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml()) #affiche du retour
```

Retour :

```
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:1847ee8b-e768-4be2-a516-a3a0bcfeff24">
  <ok/>
</rpc-reply>
```


Ajouter une loopback

```
#ici est present la configuration de la lo que je veux rajouter
netconf_loopback = """
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <interface>
      <Loopback>
        <name>1</name>
        <description>loopback</description>
        <ip>
          <address>
            <primary>
              <address>1.1.1.1</address>
              <mask>255.255.255.255</mask>
            </primary>
          </address>
        </ip>
      </Loopback>
    </interface>
  </native>
</config>
"""

netconf_reply = m.edit_config(target="running", config=netconf_loopback)

print (xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

On obtient le même retour que précédemment.

Coté routeur la configuration a bien été prise :

```
c8000v-math#sh running-config interface loopback 1
Building configuration...

Current configuration : 85 bytes
!
interface Loopback1
  description loopback
  ip address 1.1.1.1 255.255.255.255
end
```

RESTCONF

Netconf RESTCONF fournit un sous-ensemble simplifié de fonctionnalités NETCONF.

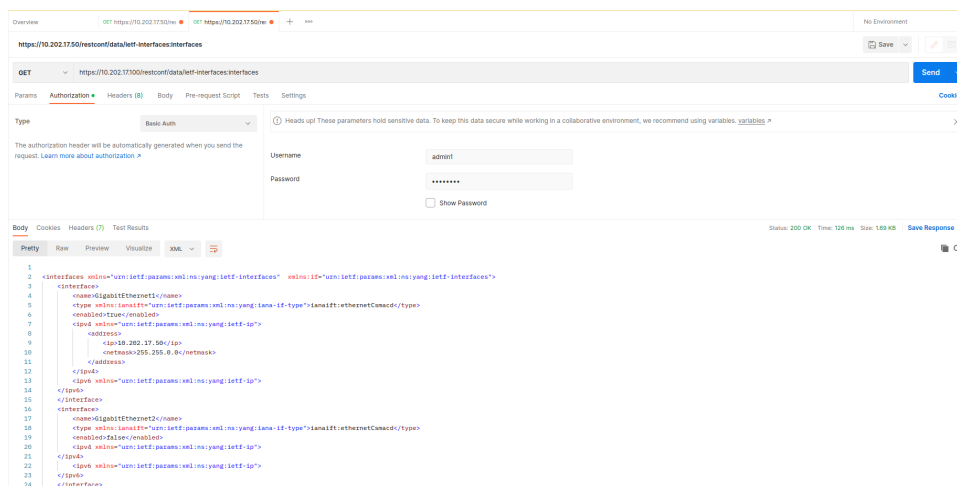
Côté routeur

Activation avec la commande `restconf` dans le mode configuration terminal du routeur et les commandes suivantes pour activer le serveur HTTPS et le mode d'authentification : `ip http secure-server` , `ip http authentication local` .

Pour vérifier que tout est en état de fonctionnement, il existe la commande `show platform software yang-management process` .

Version graphique

On peut utiliser les différentes méthodes en version graphique avec des outils comme postman. Pour cela, on doit désactiver les certificats SSL, choisir la méthode à utiliser, donner la requête souhaitée et quelques autres étapes ce qui nous permet au final d'obtenir, une fois envoyée, un résultat.



Côté client

Côté du client (administrateur), on va utiliser du python pour réaliser notre administration.

Utilisation GET

Ici, on va utiliser la méthode GET, pour obtenir la configuration de nos différentes interfaces réseaux.

```
import json
import requests
requests.packages.urllib3.disable_warnings()

##### GET #####

api_url="https://10.202.17.200/restconf/data/ietf-interfaces:interfaces" #requête que l'on souhaite effectuer

headers={"Accept":"application/yang-data+json","Content-type":"application/yang-data+json"} # entête de la requête
```

```
basicauth=("admin1", "Root123#") #credentier de notre routeur

resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False) #requête "complete"

print(resp) # affichage du code reponse

response_json = resp.json()

print(json.dumps(response_json, indent=4)) # affichage de la reponse sous forme de json
```

Retour de la requête précédente :

```
<Response [200]>
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet1",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "10.20.0.2",
              "netmask": "255.255.255.0"
            }
          ]
        },
        "ietf-ip:ipv6": {}
      },
      {
        "name": "GigabitEthernet2",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "10.202.17.200",
              "netmask": "255.255.0.0"
            }
          ]
        },
        "ietf-ip:ipv6": {}
      },
      {
        "name": "GigabitEthernet4",
        "type": "iana-if-type:ethernetCsmacd",
```

```

    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "192.168.2.1",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  },
  {
    "name": "Loopback1",
    "description": "loopback",
    "type": "iana-if-type:softwareLoopback",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "1.1.1.1",
          "netmask": "255.255.255.255"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
]
}
}

```

Ajout d'une loopback avec la méthode post

Ici, on va utiliser la méthode POST, pour créer une interface réseau.

```

import json
import requests
requests.packages.urllib3.disable_warnings()

headers={"Accept":"application/yang-data+json","Content-type":"application/yang-data+json"} #requête que l'on
souhaite effectuer

basicauth=("admin1", "Root123#") #credentier de notre routeur

api_url2 = "https://10.202.17.200/restconf/data/ietf-interfaces:interfaces/interface=Loopback2" # entête de la
requête

#crop de notre requête :
YangConfig = {

```

```
"ietf-interfaces:interface": {  
  "name": "Loopback2",  
  "description": "La loopback deux retour",  
  "type": "iana-if-type:softwareLoopback",  
  "enabled": True,  
  "ietf-ip:ipv4": {  
    "address": [  
      {  
        "ip": "5.5.5.5",  
        "netmask": "255.255.255.255"  
      }  
    ]  
  },  
  "ietf-ip:ipv6": {}  
}  
  
resp = requests.put(api_url2,data=json.dumps(YangConfig),auth=basicauth,headers=headers,verify=False)  
  
if(resp.status_code >= 200 and resp.status_code <= 299):  
    print("STATUS OK: {}".format(resp.status_code)) #affichage du status  
else:  
    print("Error. Code d'état : {} \n Message d'erreur : {}".format(resp.status_code, resp.json())) #affichage du status
```

Retour :

STATUS OK: 201

Côté routeur, la configuration a bien été prise :

```
c8000v-math#sh running-config interface loopback 2  
Building configuration...  
  
Current configuration : 100 bytes  
!  
interface Loopback2  
  description La loopback deux retour  
  ip address 5.5.5.5 255.255.255.255  
end
```

(BONUS) LES TP CISCO

TP1

7.0.3-lab---install-the-csr1000v-vm_fr-FR.pdf

Ici tout va bien, c'est normal, il est en anglais.

TP2

7.6.3-lab---automated-testing-using-pyats-and-genie_fr-FR.pdf

Ici de même, tout va bien le seul problème étant le code qui est mal interprété par le PDF ce qui cause la création d'espaces lors du copier-coller.

TP3

8.3.5-lab---explore-yang-models_fr-FR.pdf

Ici tout va bien.

TP4

8.3.6-lab---use-netconf-to-access-an-ios-xe-device_fr-FR.pdf

Problème à la page 4, il manque une ">".

```
-----  
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"  
  <capabilities>  
    <capability>urn:ietf:params:netconf:base:1.0 </capability>  
  </capabilities>  
</hello>  
]]>]]>
```

TP5

8.3.7-lab---use-restconf-to-access-an-ios-xe-device_fr-FR.pdf

Premier problème page 7, des simples quotes a la place des doubles sont utilisés.

- f. Retournez à Postman et envoyez à nouveau votre demande GET. Vous devriez maintenant voir les informations d'adressage IPv4 dans la réponse JSON, comme indiqué ci-dessous. Dans la partie suivante, vous allez copier ce format JSON pour créer une nouvelle interface.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet1",
    "description": "VBox",
    "type": "iana-if-type:ethernetCsmacd",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "192.168.56.101",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

Partie 5 : Utiliser Postman pour envoyer une demande de PUT

De même page 8.

pouvez également copier ce qui suit dans la section Corps de votre demande PUT. Notez que le type d'interface doit être défini sur **SoftwareLoopback**.

```
{
  "ietf-interfaces:interface": {
    "name": "Loopback1",
    "description": "My first RESTCONF loopback",
    "type": "iana-if-type:softwareLoopback",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "10.1.1.1",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

- c. Cliquez sur **Send** pour envoyer la demande PUT au CSR1kv. Sous la section Corps, vous devriez voir le code de réponse HTTP **Status: 201 Created**. Cela indique que la ressource a été créée avec succès.

De mêmes page 12.

```
basicauth = ("cisco", "cisco123!")
```

- d. Créez une variable de dictionnaire Python **YangConfig** qui contiendra les données YANG requises pour créer la nouvelle interface Loopback2. Vous pouvez utiliser le même dictionnaire que vous avez utilisé précédemment dans Postman. Cependant, modifiez le numéro et l'adresse de l'interface. Aussi, sachez que les valeurs booléennes doivent être capitalisées en Python. Par conséquent, assurez-vous que le **T** est mis en majuscule dans la paire clé/valeur pour **"enabled": True**.

```
YangConfig = {
  "ietf-interfaces:interface": {
    "name": "Loopback2",
    "description": "My second RESTCONF loopback",
    "type": "iana-if-type:softwareLoopback",
    "enabled": True,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "10.2.1.1",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

Ici, page 12, headers a été traduit.

- a. Avant d'entrer des instructions, veuillez noter que cette spécification de variable ne doit être que sur une seule ligne de votre script. Entrez les instructions suivantes :

Remarque : Cette spécification de variable doit se trouver sur une ligne dans votre script.

```
resp = requests.put (api_url, data=json.dumps (YangConfig), auth=basicauth, en-têtes  
=, verify=false)
```

Ici, encore page 12, il manque des "\".

- b. Entrez le code ci-dessous pour gérer la réponse. Si la réponse est l'un des messages de succès HTTP, le premier message sera imprimé. Toute autre valeur de code est considérée comme une erreur. Le code de réponse et le message d'erreur seront imprimés dans le cas où une erreur a été détectée.

```
if(resp.status_code >= 200 and resp.status_code <= 299):  
    print("STATUS OK: {}".format(resp.status_code))  
else:  
    print('Error. Code d'état : {} \n Message d'erreur : {}'.format  
(resp.status_code, resp.json ()))
```

Correction :

```
if(resp.status_code >= 200 and resp.status_code <= 299):  
    print("STATUS OK: {}".format(resp.status_code))  
else:  
    print('Error. Code d'état : {} \n Message d'erreur : {}'.format(resp.status_code, resp.json()))
```


SIGNATURE DE MR. ALLEAUME JULIEN

Présents :

ATELIERS
JOSEPH

Fabricant de fenêtre bois

Menuiserie Bois à l'ancienne - Menuiserie Bois à recouvrement - Fenêtres passives Haute performances
Chassis de ventilation naturelle automatisé - Coulissant Bois - Chassis feu et non feu - Huisseries bois sur stock

Je soussigné Alleaume Julien,
Possédant bien le réseau 192.168.1.0/24 et le réseau
192.168.10.0/24.

Je confirme que mon collègue et ami MR. Puig
Mathieu et bien mon voisin BGP, et de plus qu'il
arrive bien à accéder à ma machine virtuelle
Debian qui possède l'IP 192.168.10.2

Alleaume Julien
le 12/01/2022 à Béziers

FENÊTRE BOIS POUR CHÂTEAUX ET DOMAINES VITICOLES
34500 BEZIERS - Tél. 04 67 31 82 69
email : commercial@ateliers-joseph.fr - www.ateliers-joseph.fr