



Bgp At Doors of Autonomous Systems is Simple (BADASS)

Résumé: Ce document est un sujet sur l'administration de réseau.

Table des matières

I	Préambule	2
II	Introduction	3
III	Consignes générales	4
IV	Partie obligatoire	5
IV.1	Partie 1 : Configuration de GNS3 avec Docker	6
IV.2	Partie 2 : Découverte d'un VXLAN	8
IV.3	Partie 3 : Découverte de BGP avec EVPN	11
V	Rendu et peer-évaluation	16

Chapitre I

Préambule



Chapitre II

Introduction

Ce projet a pour but d'approfondir vos connaissances apprises par NetPractice. Vous allez devoir simuler un réseau et le configurer en utilisant **GNS3** avec des images **docker**.

BGP EVPN repose sur BGP (RFC 4271) et ses extensions MP-BGP (RFC4760). BGP est le protocole de routage animant Internet. Via les extensions MP-BGP, il peut être utilisé pour transporter des informations d'accessibilité (NLRI) pour divers protocoles (IPv4, IPv6, L3 VPN et dans le cas qui nous intéresse, EVPN). EVPN est une famille spéciale permettant de publier des informations sur les adresses MAC et les équipements terminaux y donnant accès.

Chapitre III

Consignes générales

- L'intégralité de ce projet est à réaliser dans une **machine virtuelle**.
- Ce projet implique d'utiliser et donc d'installer **docker** ainsi que **GNS3**.
- Vous devez rendre tous les fichiers nécessaires à la configuration de votre projet dans les dossiers P1, P2 et P3.



ATTENTION: Ce projet utilise beaucoup de nouvelles notions. Il ne faut pas avoir peur de prendre un certain temps à lire le plus d'information sur le fonctionnement de BGP et des VXLAN. Beaucoup de nouveaux termes sont volontairement utilisées.

Chapitre IV

Partie obligatoire

Ce projet consiste à vous faire mettre en place plusieurs environnements en suivant des règles spécifiques.

Ce projet est découpé en trois parties à faire dans l'ordre indiqué :

- Partie 1 : Configuration de GNS3 avec Docker.
- Partie 2 : Découverte d'un VXLAN.
- Partie 3 : Découverte de BGP avec EVPN.



Il faut lire le sujet **entièrement** pour bien comprendre ce qui est demandé ici.

IV.1 Partie 1 : Configuration de GNS3 avec Docker

Pour cette première partie vous allez devoir configurer GNS3. Il faut donc installer et configurer GNS3 ainsi que `docker` dans votre machine virtuelle.

Maintenant que tout fonctionne vous devez utiliser deux images `docker` que vous devez fabriquer.

Une première image avec un système de votre choix contenant au minimum `busybox` ou une solution équivalente.



Alpine semble être une bonne solution.

Une seconde image utilisant un système de votre choix avec les contraintes suivantes :

- Un logiciel qui gère le routage de paquets (`zebra` ou `quagga`).
- Le service `BGPD` actif et configuré.
- Le service `OSPFD` actif et configuré
- Un service de moteur de routage `IS-IS`.
- `Busybox` ou une équivalence.



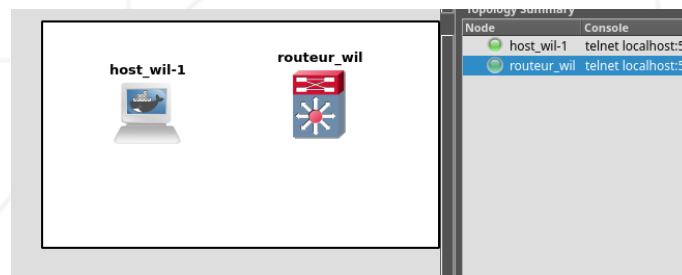
Il existe des images `pre-build` qu'il faudra configurer avec ce genre service. Vos conteneurs doivent fonctionner dans GNS3 avec les services demandés. Vous pouvez ajouter ce que vous souhaitez pour réaliser ce projet.



Attention: Vos images seront utilisées dans tout ce projet. Aucune adresse IP ne doit être configurée par défaut.

Bgp At Doors of Autonomous Systems is Simple (BADASS)

Vous devez utiliser ces deux images **docker** dans **GNS3** et réaliser ce petit schéma. Il faut que vos deux machines soient fonctionnelles. On doit pouvoir s'y connecter par **GNS3**.



Le nom des machines n'est pas mis au hasard il faudra avoir votre login dans le nom de chaque équipement (ici wil).

Voici un exemple de chaque images configurée dans **GNS3** :

```
/ # ps
PID  USER      TIME  COMMAND
  1  root         0:00  /bin/sh
 38  root         0:00  /gns3/bin/busybox script -qfc while true; do TERM=vt100 /g
 44  root         0:00  /bin/ash -c while true; do TERM=vt100 /gns3/bin/busybox sh
 45  root         0:00  /gns3/bin/busybox sh
 59  root         0:00  ps
/ # hostname
host_wil-1
/ #

/ # ps
PID  USER      TIME  COMMAND
  1  root         0:00  /sbin/tini -- /usr/lib/frr/docker-start
 38  root         0:00  /gns3/bin/busybox script -qfc while true; do TERM=vt100 /g
 45  root         0:00  /bin/ash -c while true; do TERM=vt100 /gns3/bin/busybox sh
 46  root         0:00  /gns3/bin/busybox sh
 94  root         0:00  tail -f /dev/null
136  frr          0:00  /usr/lib/frr/zebra -d -F traditional -A 127.0.0.1 -s 90000
141  frr          0:00  /usr/lib/frr/bgpd -d -F traditional -A 127.0.0.1
148  frr          0:00  /usr/lib/frr/ospfd -d -F traditional -A 127.0.0.1
151  frr          0:00  /usr/lib/frr/isisd -d -F traditional -A 127.0.0.1
254  root         0:00  ps
/ # hostname
routeur_wil
/ #
```

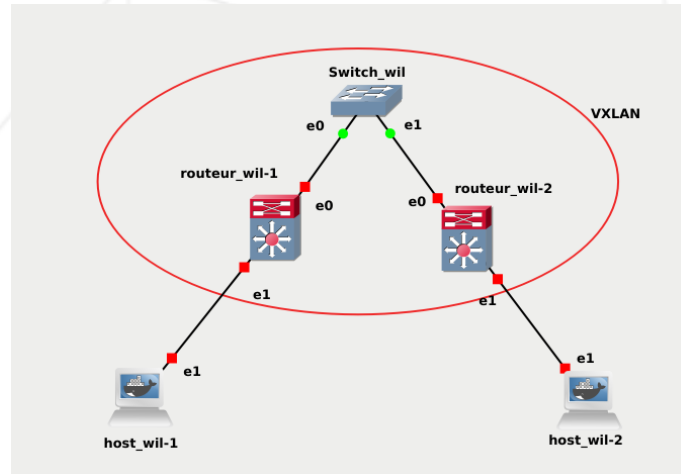
Vous devez rendre ce projet dans un dossier **P1** à la racine de votre dépôt git. Vous devez aussi ajouter les fichiers de configuration avec des commentaires pour expliquer la mise en place de chaque équipement.



Vous devez exporter ce projet avec une compression **ZIP** en incluant les bases image. Ce fichier doit être visible dans votre dépôt git.

IV.2 Partie 2 : Découverte d'un VXLAN

Vous avez maintenant une base fonctionnelle pour commencer à mettre en place votre premier réseau VXLAN (RFC 7348). Dans un premier temps en statique puis en dynamique multicast. Voici la topologie de votre premier vxlan :



Vous devez configurer ce réseau en utilisant un VXLAN avec pour ID 10 comme indiqué dans les exemples ci-dessous. Vous pouvez utiliser un nom de VXLAN que vous souhaitez ici : `vxlan10`. Vous devez mettre en place un bridge ici : `br0`. Vous devez configurer vos interfaces ETHERNET comme vous souhaitez. Voici un exemple de résultat attendu lorsque l'on va inspecter le trafic entre nos deux machines dans notre VXLAN :

```
host_wil-1 /usr/bin/zsh 80x24
# hostname
host_wil-1
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr DA:DA:AB:D7:54:C4
          inet addr:30.1.1.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::d8da:abff:fed7:54c4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3972 (3.8 KiB)  TX bytes:1216 (1.1 KiB)

# ping 30.1.1.2
PING 30.1.1.2 (30.1.1.2): 56 data bytes
64 bytes from 30.1.1.2: seq=0 ttl=64 time=0.443 ms
^C
--- 30.1.1.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.443/0.443/0.443 ms
#
```

```
host_wil-2 /usr/bin/zsh 80x24
# hostname
host_wil-2
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 92:E6:7C:16:F3:18
          inet addr:30.1.1.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::90e6:7c16:f318/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2180 (2.1 KiB)  TX bytes:1216 (1.1 KiB)
#
```

Packet capture window showing ICMP traffic:

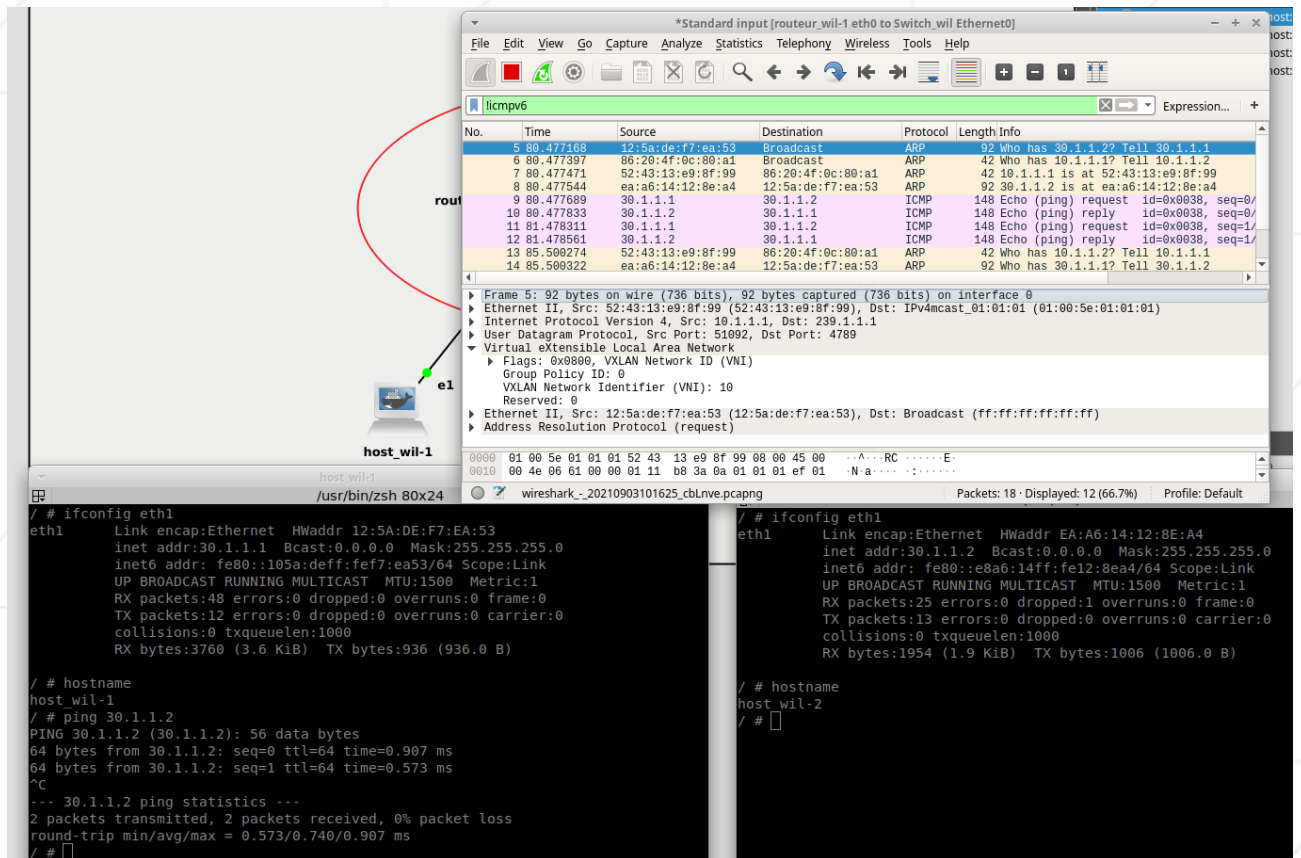
No.	Time	Source	Destination	Protocol	Length	Info
5	36.796751	30.1.1.1	30.1.1.2	ICMP	148	Echo (ping) request id=0x0039, seq=0/0
6	36.796971	30.1.1.2	30.1.1.1	ICMP	148	Echo (ping) reply id=0x0039, seq=0/0

Frame 5: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0

- Ethernet II, Src: ca:f0:be:0d:3b:fa (ca:f0:be:0d:3b:fa), Dst: 46:41:59:cc:db:8f (46:41:59:cc:db:8f)
- Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.1.2
- User Datagram Protocol, Src Port: 59899, Dst Port: 4789
- Virtual eXtensible Local Area Network
 - Flags: 0x0000, VXLAN Network ID (VNI)
 - Group Policy ID: 0
 - VXLAN Network Identifier (VNI): 10
 - Reserved: 0
- Ethernet II, Src: da:da:ab:d7:54:c4 (da:da:ab:d7:54:c4), Dst: 92:e6:7c:16:f3:18 (92:e6:7c:16:f3:18)
- Internet Protocol Version 4, Src: 30.1.1.1, Dst: 30.1.1.2
- Internet Control Message Protocol

Bgp At Doors of Autonomous Systems is Simple (BADASS)

On va maintenant voir la même chose en utilisant les groupes dont le but sera de pouvoir de faire du multicast dynamique.



```
/ # ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 12:5A:DE:F7:EA:53
          inet addr:30.1.1.1  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::105a:deff:fe7f:ea53/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:48 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3760 (3.6 KiB)  TX bytes:936 (936.0 B)

/ # hostname
host_wil-1
/ # ping 30.1.1.2
PING 30.1.1.2 (30.1.1.2): 56 data bytes
64 bytes from 30.1.1.2: seq=0 ttl=64 time=0.907 ms
64 bytes from 30.1.1.2: seq=1 ttl=64 time=0.573 ms
^C
--- 30.1.1.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.573/0.740/0.907 ms
/ #
```

```
/ # ifconfig eth1
eth1      Link encap:Ethernet  HWaddr EA:A6:14:12:8E:A4
          inet addr:30.1.1.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::e8a6:14ff:fe12:8ea4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25 errors:0 dropped:1 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1954 (1.9 KiB)  TX bytes:1006 (1006.0 B)

/ # hostname
host_wil-2
/ #
```

On peut remarquer que nos machines possèdent maintenant un groupe (ici 239.1.1.1 vous pouvez modifier cette partie) :

```
/ # ip -d link show vxlan10
3: vxlan10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master br0 state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether da:83:84:19:04:d5 brd ff:ff:ff:ff:ff:ff promiscuity 1 minmtu 68 maxmtu 65535
    vxlan id 10 group 239.1.1.1 dev eth0 srcport 0 dstport 4789 ttl auto ageing 300 udpchecksum noudp6zerocsumtx noudp6zerocsumrx
    bridge_slave state forwarding priority 32 cost 100 hairpin off guard off root_block off fastleave off learning on flood on flood on port id 0x8002 port no 0x2 designated port 32770 designated cost 0 designated bridge 8000.DA:83:84:19:04:D5 designated root 8000.DA:83:84:19:04:D5 hold timer 0.00 message_age timer 0.00 forward_delay timer 0.00 topology_change ack 0 config pending 0 proxy_arp off proxy_arp_wifi off mcast_router 1 mcast_fast_leave off mcast_flood on mcast_to_unicast off neigh_suppress off group_fwd_mask 0 group_fwd_mask_str 0x0 vlan_tunnel off isolated off addrgenmode eui64 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
/ # hostname
routeur_wil-1
/ #
```

```
/ # ip -d link show vxlan10
3: vxlan10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue master br0 state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether ae:df:08:4c:0c:0d brd ff:ff:ff:ff:ff:ff promiscuity 1 minmtu 68 maxmtu 65535
    vxlan id 10 group 239.1.1.1 dev eth0 srcport 0 dstport 4789 ttl auto ageing 300 udpchecksum noudp6zerocsumtx noudp6zerocsumrx
    bridge_slave state forwarding priority 32 cost 100 hairpin off guard off root_block off fastleave off learning on flood on flood on port id 0x8002 port no 0x2 designated port 32770 designated cost 0 designated bridge 8000.02:AF:63:0B:5E:00 designated root 8000.02:AF:63:0B:5E:00 hold timer 0.00 message_age timer 0.00 forward_delay timer 0.00 topology_change ack 0 config pending 0 proxy_arp off proxy_arp_wifi off mcast_router 1 mcast_fast_leave off mcast_flood on mcast_to_unicast off neigh_suppress off group_fwd_mask 0 group_fwd_mask_str 0x0 vlan_tunnel off isolated off addrgenmode eui64 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
/ # hostname
routeur_wil-2
/ #
```

Bgp At Doors of Autonomous Systems is Simple (BADASS)

Un exemple d'affichage de notre table d'adresse mac dans nos deux routeurs :

/usr/bin/zsh 59x24				/usr/bin/zsh 63x24			
# hostname				# hostname			
routeur_wil-1				routeur_wil-2			
# brctl showmacs br0				# brctl showmacs br0			
port no	mac addr	is local?	ageing time	port no	mac addr	is local?	ageing timer
2	02:af:63:db:5e:00	no	136.98	1	02:af:63:db:5e:00	yes	0.00
1	12:5a:de:f7:ea:53	no	202.52	1	02:af:63:db:5e:00	yes	0.00
2	ae:df:08:4c:0c:0d	no	196.37	2	12:5a:de:f7:ea:53	no	169.23
2	da:83:84:19:04:d5	yes	0.00	2	ae:df:08:4c:0c:0d	yes	0.00
2	da:83:84:19:04:d5	yes	0.00	2	da:83:84:19:04:d5	no	21.77
2	ea:a6:14:12:8e:a4	no	215.32	1	ea:a6:14:12:8e:a4	no	182.03
1	ee:3f:ab:b3:36:38	yes	0.00	#			
1	ee:3f:ab:b3:36:38	yes	0.00	#			

Vous devez rendre ce projet dans un dossier P2 à la racine de votre dépôt git. Vous devez ajouter les fichiers de configuration avec des commentaires pour expliquer la mise en place de chaque équipement.



Vous devez exporter ce projet avec une compression ZIP en incluant les bases image. Ce fichier doit être visible dans votre dépôt git.



Vous devez utiliser des noms correctes et cohérents pour vos équipements ici avec le login d'un des membres du groupe.

IV.3 Partie 3 : Découverte de BGP avec EVPN

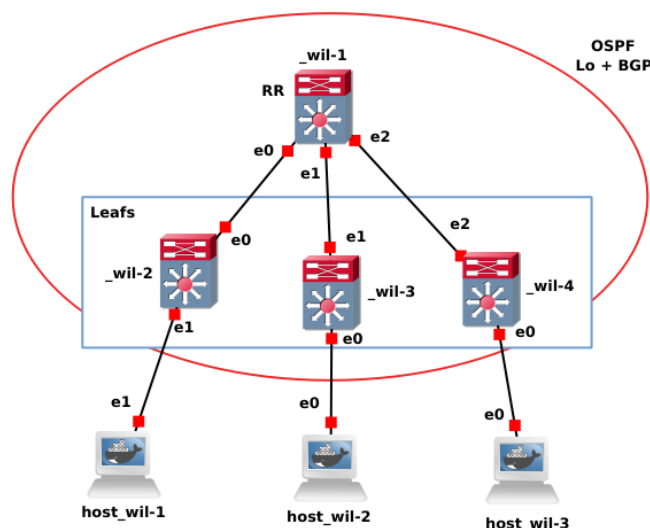
Maintenant que vous maîtrisez le principe de base du VXLAN on va aller encore un peu plus loin et explorer le principe du BGP EVPN (rfc 7432) sans utiliser MPLS pour simplifier les choses. Le contrôleur va apprendre les adresses MAC. On va utiliser notre VXLAN avec pour ID 10 vu dans la partie précédente.

Comme dans la seconde partie on commence par la topologie du réseau attendu. On va utiliser le principe de la réfection de route (=RR). Nos leafs (VTEP) seront configurés pour avoir des relations dynamique.

Ce schéma représente un petit Datacenter.



Par soucis de lisibilité les noms sont plus court ici. Vous allez devoir utiliser OSPF pour simplifier l'évaluation.



On peut voir notre visibilité depuis notre VTEP _wil-4 les 3 VTEPs 1.1.1.[1.4] :

```
wil-4(config-router)# do 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, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

O>* 1.1.1.32 [110/10000] via 10.1.1.9, eth2, weight 1, 00:00:44
O>* 1.1.1.3/32 [110/20000] via 10.1.1.9, eth2, weight 1, 00:00:44
O 1.1.1.4/32 [110/0] is directly connected, lo, weight 1, 00:00:58
C>* 1.1.1.4/32 is directly connected, lo, 00:00:58
O>* 10.1.1.0/30 [110/20000] via 10.1.1.9, eth2, weight 1, 00:00:44
O>* 10.1.1.4/30 [110/20000] via 10.1.1.9, eth2, weight 1, 00:00:44
O 10.1.1.8/30 [110/10000] is directly connected, eth2, weight 1, 00:00:49
C>* 10.1.1.8/30 is directly connected, eth2, 00:00:58
wil-4(config-router)#
```

Bgp At Doors of Autonomous Systems is Simple (BADASS)

On ne possède qu'une seule route pour le moment avec notre contrôleur (RR) :

```
wil-4(config-router)# do sh bgp summary

IPv4 Unicast Summary:
BGP router identifier 1.1.1.4, local AS number 1 vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 1, using 14 KiB of memory

Neighbor      V      AS  MsgRcvd  MsgSent  TblVer  InQ  OutQ  Up/Down  State/PfxRcd  PfxSnt
1.1.1.1        4        1         7         7         0     0     0 00:02:12      0           0

Total number of neighbors 1

L2VPN EVPN Summary:
BGP router identifier 1.1.1.4, local AS number 1 vrf-id 0
BGP table version 0
RIB entries 1, using 192 bytes of memory
Peers 1, using 14 KiB of memory

Neighbor      V      AS  MsgRcvd  MsgSent  TblVer  InQ  OutQ  Up/Down  State/PfxRcd  PfxSnt
1.1.1.1        4        1         7         7         0     0     0 00:02:12      0           1

Total number of neighbors 1
wil-4(config-router)#
```

Lorsqu'il n'y a pas d'host en cours de fonctionnement on peut voir notre VNI (10 ici) ainsi que nos routes pre config (type 3). Aucune route type 2 ne semble exister et c'est tout à fait normal.

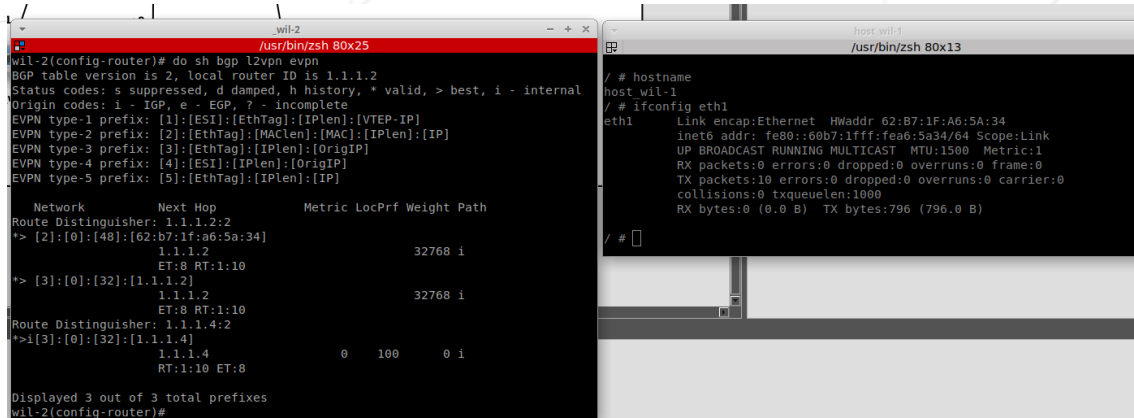
```
wil-4(config-router)# do sh bgp l2vpn evpn
BGP table version is 1, local router ID is 1.1.1.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[ESI]:[EthTag]:[IPlen]:[VTEP-IP]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

  Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.2:2
*>i[3]:[0]:[32]:[1.1.1.2]
  1.1.1.2                      0    100      0 i
  RT:1:10 ET:8
Route Distinguisher: 1.1.1.4:2
*> [3]:[0]:[32]:[1.1.1.4]
  1.1.1.4                      32768 i
  ET:8 RT:1:10

Displayed 2 out of 2 total prefixes
wil-4(config-router)#
```

Bgp At Doors of Autonomous Systems is Simple (BADASS)

Une machine `host_wil-1` est maintenant fonctionnelle. On peut remarquer que sans assignation d'adresse IP notre VTEP (`wil_2`) découvre automatiquement l'adresse MAC des machines fonctionnelles. On peut voir aussi la création automatique d'une route type 2 :



The image shows two terminal windows. The left window is titled `_wil-2` and shows the output of the command `do sh bgp l2vpn evpn` on a router. It displays BGP table version 2, local router ID 1.1.1.2, and various EVPN prefixes. The right window is titled `host_wil-1` and shows the output of the command `ifconfig eth1`, displaying Ethernet interface details like MAC address, MTU, and statistics.

```
_wil-2(config-router)# do sh bgp l2vpn evpn
BGP table version is 2, local router ID is 1.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[ESI]:[EthTag]:[IPlen]:[VTEP-IP]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

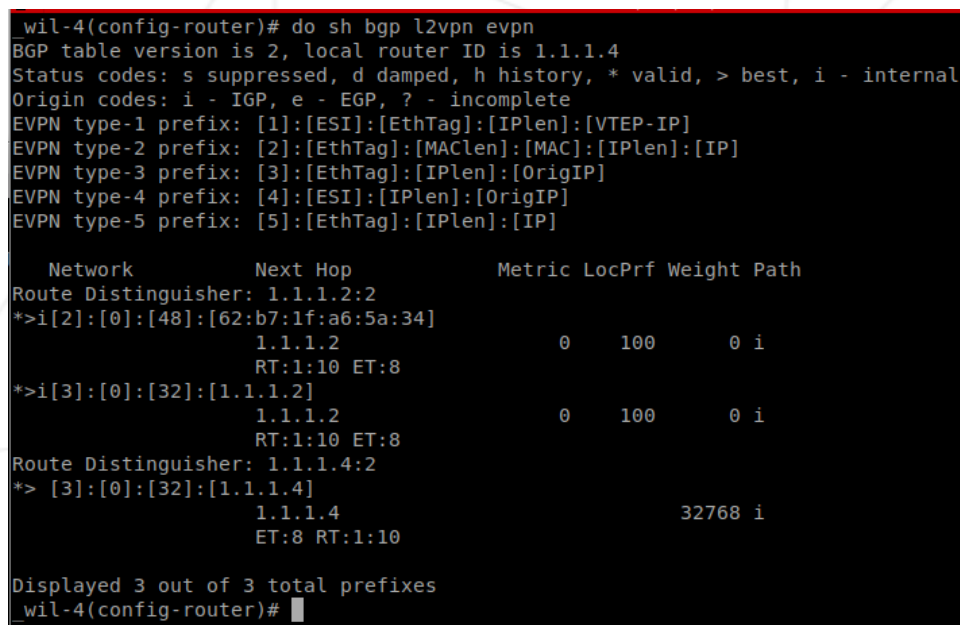
  Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.2:2
*> [2]:[0]:[48]:[62:b7:1f:a6:5a:34]
    1.1.1.2
    ET:8 RT:1:10
    32768 i
*> [3]:[0]:[32]:[1.1.1.2]
    1.1.1.2
    ET:8 RT:1:10
    32768 i
Route Distinguisher: 1.1.1.4:2
*>i[3]:[0]:[32]:[1.1.1.4]
    1.1.1.4
    RT:1:10 ET:8
    0 100 0 i
Displayed 3 out of 3 total prefixes
_wil-2(config-router)#
```

```
host_wil-1
/usr/bin/zsh 80x13

/ # hostname
host_wil-1
/ # ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 62:B7:1F:A6:5A:34
          inet6 addr: fe80::60b7:1fff:fe6a:5a34/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:796 (796.0 B)

/ #
```

De la même façon lorsque l'on regarde un second VTEP (`_wil-4`) on peut remarquer la création d'une nouvelle route type 2 générée par notre RR :



The image shows a terminal window titled `_wil-4` displaying the output of the command `do sh bgp l2vpn evpn`. It shows BGP table version 2, local router ID 1.1.1.4, and various EVPN prefixes. The output includes a table of routes with Network, Next Hop, Metric, LocPrf, Weight, and Path columns.

```
_wil-4(config-router)# do sh bgp l2vpn evpn
BGP table version is 2, local router ID is 1.1.1.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[ESI]:[EthTag]:[IPlen]:[VTEP-IP]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

  Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.2:2
*>i[2]:[0]:[48]:[62:b7:1f:a6:5a:34]
    1.1.1.2
    RT:1:10 ET:8
    0 100 0 i
*>i[3]:[0]:[32]:[1.1.1.2]
    1.1.1.2
    RT:1:10 ET:8
    0 100 0 i
Route Distinguisher: 1.1.1.4:2
*> [3]:[0]:[32]:[1.1.1.4]
    1.1.1.4
    ET:8 RT:1:10
    32768 i
Displayed 3 out of 3 total prefixes
_wil-4(config-router)#
```


Bgp At Doors of Autonomous Systems is Simple (BADASS)

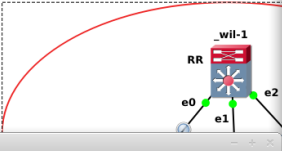
On recommence l'opération avec une seconde machine (host_wil-3). On peut remarquer la seconde route mise en place de type 2. Il n'y a pas d'assignation d'adresse IP :

```
wil-2(config-router)# do sh bgp l2vpn evpn
BGP table version is 2, local router ID is 1.1.1.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[ESI]:[EthTag]:[IPlen]:[VTEP-IP]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

  Network          Next Hop          Metric LocPrf Weight Path
Route Distinguisher: 1.1.1.2:2
*> [2]:[0]:[48]:[62:b7:1f:a6:5a:34]
    1.1.1.2                      32768 i
    ET:8 RT:1:10
*> [3]:[0]:[32]:[1.1.1.2]
    1.1.1.2                      32768 i
    ET:8 RT:1:10
Route Distinguisher: 1.1.1.4:2
*>i[2]:[0]:[48]:[1e:80:95:23:b8:45]
    1.1.1.4                      0    100    0 i
    RT:1:10 ET:8
*>i[3]:[0]:[32]:[1.1.1.4]
    1.1.1.4                      0    100    0 i
    RT:1:10 ET:8

Displayed 4 out of 4 total prefixes
wil-2(config-router)#
```

Pour notre vérification un simple ping nous permet de voir que l'on peut accéder à toute les machines via notre RR en utilisant les VTEPs. On peut voir le VXLAN configuré à 10 ainsi que nos packets ICMP. On voit aussi des packets OSPF configuré :



```
host_wil-1
# hostname
host_wil-1
# ping 20.1.1.2
PING 20.1.1.2 (20.1.1.2): 56 data bytes
64 bytes from 20.1.1.2: seq=0 ttl=64 time=0.993 ms
64 bytes from 20.1.1.2: seq=1 ttl=64 time=0.660 ms
64 bytes from 20.1.1.2: seq=2 ttl=64 time=0.612 ms
^C
--- 20.1.1.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.612/0.755/0.993 ms
# ifconfig eth1
eth1    Link encap:Ethernet  HWaddr 62:B7:1F:A6:5A:34
        inet addr:20.1.1.1 Bcast:0.0.0.0 Mask:255.255.255.0
        inet6 addr: fe80::60b7:1fff:fe6a:5a34/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:22 errors:0 dropped:0 overruns:0 frame:0
        TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1664 (1.6 KiB)  TX bytes:1384 (1.3 KiB)

#
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	ea:1e:d4:81:84:a6	2a:a6:79:c3:92:a0	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
2	0.000079	2a:a6:79:c3:92:a0	ea:1e:d4:81:84:a6	ARP	42	10.1.1.2 is at 2a:a6:79:c3:92:a0
3	4.000091	10.1.1.2	224.0.0.5	OSPF	82	Hello Packet
4	5.007560	10.1.1.1	224.0.0.5	OSPF	82	Hello Packet
5	14.440283	62:b7:1f:a6:5a:34	Broadcast	ARP	92	Who has 20.1.1.2? Tell 20.1.1.1
6	14.449575	1e:80:95:23:b8:45	62:b7:1f:a6:5a:34	ARP	92	20.1.1.2 is at 1e:80:95:23:b8:45
7	14.449728	20.1.1.1	20.1.1.2	ICMP	148	Echo (ping) request id=0x0029, seq=0
8	14.450029	20.1.1.2	20.1.1.1	ICMP	148	Echo (ping) reply id=0x0029, seq=0
9	14.631476	10.1.1.2	224.0.0.5	OSPF	82	Hello Packet
10	15.067560	10.1.1.1	224.0.0.5	OSPF	82	Hello Packet
11	15.450339	20.1.1.1	20.1.1.2	ICMP	148	Echo (ping) request id=0x0029, seq=1
12	15.450741	20.1.1.2	20.1.1.1	ICMP	148	Echo (ping) reply id=0x0029, seq=1
13	16.455903	20.1.1.1	20.1.1.2	ICMP	148	Echo (ping) request id=0x0029, seq=2
14	16.456291	20.1.1.2	20.1.1.1	ICMP	148	Echo (ping) reply id=0x0029, seq=2
15	19.456560	1e:80:95:23:b8:45	62:b7:1f:a6:5a:34	ARP	92	Who has 20.1.1.1? Tell 20.1.1.2
16	19.456742	62:b7:1f:a6:5a:34	1e:80:95:23:b8:45	ARP	92	20.1.1.1 is at 62:b7:1f:a6:5a:34
17	24.580155	ea:1e:d4:81:84:a6	2a:a6:79:c3:92:a0	ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
18	24.580223	2a:a6:79:c3:92:a0	ea:1e:d4:81:84:a6	ARP	42	10.1.1.2 is at 2a:a6:79:c3:92:a0
19	24.611523	10.1.1.2	224.0.0.5	OSPF	82	Hello Packet
20	25.068015	10.1.1.1	224.0.0.5	OSPF	82	Hello Packet

```
host_wil-3
# ifconfig eth0
eth0    Link encap:Ethernet  HWaddr 1E:80:95:23:B8:45
        inet addr:20.1.1.2 Bcast:0.0.0.0 Mask:255.255.255.0
        inet6 addr: fe80::1c80:95ff:fe23:b845/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:8 errors:0 dropped:0 overruns:0 frame:0
        TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:588 (588.0 B)  TX bytes:1314 (1.2 KiB)

#
```

Bgp At Doors of Autonomous Systems is Simple (BADASS)

Vous devez rendre ce projet dans un dossier P3 à la racine de votre dépôt git. Vous devez ajouter les fichiers de configuration avec des commentaires pour expliquer la mise en place de chaque équipement.



Vous devez exporter ce projet avec une compression ZIP en incluant les bases image. Ce fichier doit être visible dans votre dépôt git.



Vous devez utiliser des noms correctes et cohérents pour vos équipements ici avec le login d'un des membres du groupe.

Chapitre V

Rendu et peer-évaluation

Rendez votre travail dans votre dépôt Git comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

En résumé :

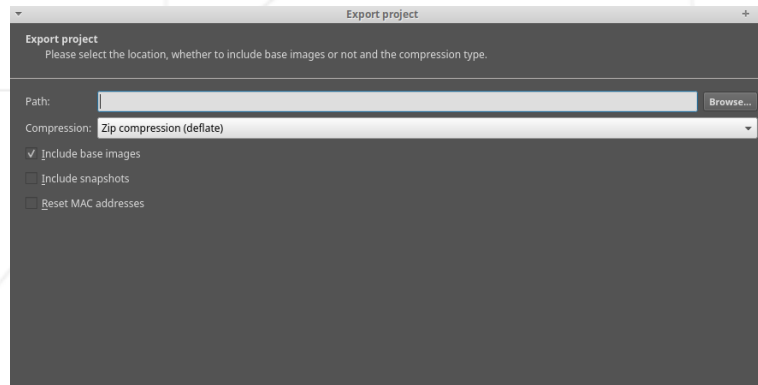
- Vous devez rendre à la racine de votre dépôt la partie obligatoire dans trois dossiers : P1, P2 et P3.

Voici un exemple de structure attendue dans votre rendu :

```
> find -maxdepth 2 -ls
424242      4 drwxr-xr-x  6 wandre wil42      4096 sept. 17 23:42 .
424242      4 drwxr-xr-x  3 wandre wil42      4096 sept. 17 23:42 ./P1
424242      4 -rw-r--r--  1 wandre wil42      XXXX sept. 17 23:42 ./P1/P1.gns3project
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P1/_wil-1_host
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P1/_wil-2
424242      4 drwxr-xr-x  3 wandre wil42      4096 sept. 17 23:42 ./P2
424242      4 -rw-r--r--  1 wandre wil42      XXXX sept. 17 23:42 ./P2/P2.gns3project
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P2/_wil-1_g
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P2/_wil-1_host
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P2/_wil-1_s
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P2/_wil-2_g
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P2/_wil-2_host
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P2/_wil-2_s
424242      4 drwxr-xr-x  3 wandre wil42      4096 sept. 17 23:42 ./P3
424242      4 -rw-r--r--  2 wandre wil42      4096 sept. 17 23:42 ./P3/P3.gns3project
424242      4 -rw-r--r--  2 wandre wil42      4096 sept. 17 23:42 ./P3/_wil-1
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P3/_wil-1_host
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P3/_wil-2
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P3/_wil-2_host
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P3/_wil-3
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P3/_wil-3_host
424242      4 -rw-r--r--  2 wandre wil42      XXXX sept. 17 23:42 ./P3/_wil-4
> file P3/P3.gns3project
P3/P3.gns3project: Zip archive data, at least v2.0 to extract
```

Bgp At Doors of Autonomous Systems is Simple (BADASS)

Pour exporter vos projets au format zip il faut aller dans le menu file puis export portable project :



Durant l'évaluation vous allez devoir expliquer les termes utilisés dans le sujet. Nous vous invitons fortement à prendre du temps pour comprendre chacun de ceux-ci.



L'évaluation se déroulera sur l'ordinateur du groupe évalué.