Cours Virtualisation des réseaux

Nantes Ynov Campus – 2022-2023

Activité Pratique 6
Mise en place d'un réseau virtuel étendu VXLAN avec Open vSwitch

Objectifs :

⇨ Découvrir et essayer pratiquement la configuration d'un VXLAN

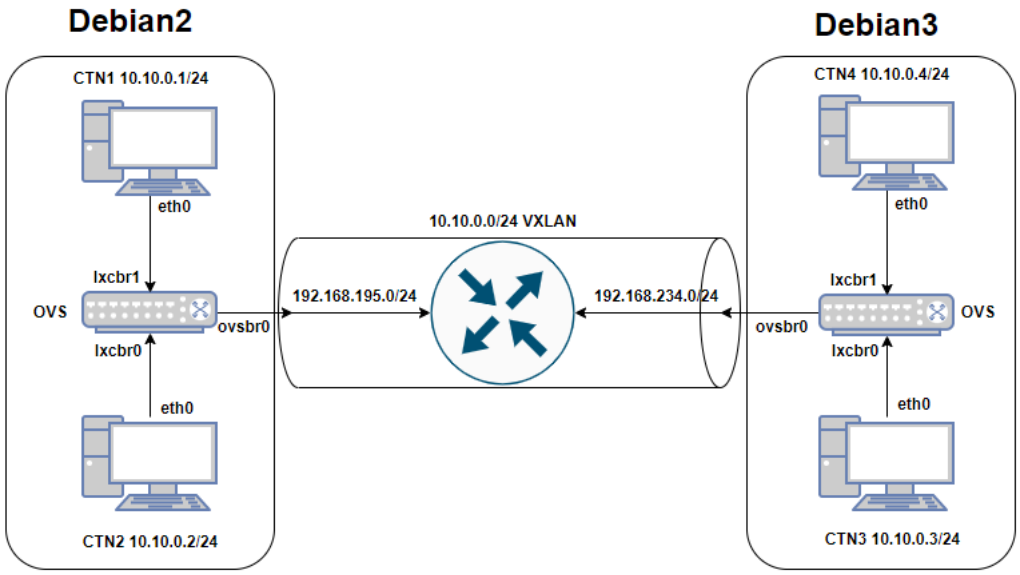
Pré requis

Pour réaliser ce TP vous devrez disposer d'au moins deux machines sur deux sous réseaux différents qui devront pouvoir communiquer entre elles.

Debian1 : 4Go RAM, 2vcpu, 15Go DD

La machine Debian2 devra avoir un accès internet pour télécharger les paquets nécessaires à l'installation du switch.

Schéma d'architecture VXLAN



Etape 1 : Créez un pont de mise en réseau de conteneurs

Vérification de la configuration

```
will@debian3:~$ sudo ovs-vsctl show
[sudo] Mot de passe de will :
373d5511-3013-45af-97bc-1625fa67652b
    ovs_version: "2.15.0"
```

Vérification des éléments installés

```
will@debian3:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g
roup default qlen 1000
    link/ether 00:0c:29:49:da:af brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.234.5/24 brd 192.168.234.255 scope global dynamic noprefixroute
 ens33
       valid_lft 6770sec preferred_lft 6770sec
    inet6 fe80::20c:29ff:fe49:daaf/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: lxcbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
 group default qlen 1000
    link/ether 00:16:3e:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.1/24 brd 10.0.3.255 scope global lxcbr0
       valid_lft forever preferred_lft forever
```

Création du port bridge

```
will@debian3:~$ sudo ovs-vsctl add-br ovsbr0
will@debian3:~$ sudo ovs-vsctl show
373d5511-3013-45af-97bc-1625fa67652b
    Bridge ovsbr0
        Port ovsbr0
            Interface ovsbr0
                type: internal
    ovs_version: "2.15.0"
```

Activer l'interface crée : sudo ip link set dev ovsbr0 up

```
will@debian3:~$ sudo ip link set dev ovsbr0 up
```

Modification du MTU : sudo ovs-vsctl set int ovsbr0 mtu_request=1416
Le MTU Ethernet du commutateur doit être réduit en raison de la tunnellisation et de la surcharge IPsec/VLAN

```
will@debian3:~$ sudo ovs-vsctl set int ovsbr0 mtu_request=1416
will@debian3:~$
```

Etape 2 : Lier le commutateur à l'interface réseau de nos futurs VM

Attacher le port au commutateur et lui affecter un tag
sudo ovs-vsctl add-port ovsbr0 lxcbr0 tag=100

```
will@debian3:~$ sudo ovs-vsctl show
373d5511-3013-45af-97bc-1625fa67652b
    Bridge ovsbr0
        Port lxcbr0
            tag: 100
            Interface lxcbr0
        Port ovsbr0
            Interface ovsbr0
                type: internal
    ovs_version: "2.15.0"
will@debian3:~$
```

Créer le port VXLAN qui établira le tunnel :

sudo ovs-vsctl add-port ovsbr0 vxlan1 -- set interface vxlan1 type=vxlan options:remote_ip=192.168.195.20

```
will@debian3:~$ sudo ovs-vsctl add-port ovsbr0 vxlan1 -- set interface vxlan1 ty
pe=vxlan options:remote_ip=192.168.195.20
will@debian3:~$ sudo ovs-vsctl show
373d5511-3013-45af-97bc-1625fa67652b
    Bridge ovsbr0
        Port lxcbr0
            tag: 100
            Interface lxcbr0
        Port vxlan1
            Interface vxlan1
                type: vxlan
                options: {remote_ip="192.168.195.20"}
        Port ovsbr0
            Interface ovsbr0
                type: internal
    ovs_version: "2.15.0"
will@debian3:~$
```

Etape 3 : Créer un conteneur de TEST

Créer le conteneur qui servira de test

sudo lxc-create -n ctn3 -t download -- -d debian -r bullseye -a amd64

```
will@debian3:~$ sudo lxc-create -n ctn3 -t download -- -d debian -r bullseye -a
amd64
Setting up the GPG keyring
Downloading the image index
Downloading the rootfs
Downloading the metadata
The image cache is now ready
Unpacking the rootfs

---
You just created a Debian bullseye amd64 (20230131_05:24) container.

To enable SSH, run: apt install openssh-server
No default root or user password are set by LXC.
will@debian3:~$
```

Etape 4 : Fixer l’adresse IP de notre conteneur

Modifier la configuration du conteneur pour fixer l’adresse IP

```
will@debian3:~$ sudo nano /var/lib/lxc/ctn3/config
```

```
# (Be aware this has security implications)


# Distribution configuration
lxc.include = /usr/share/lxc/config/common.conf
lxc.arch = linux64

# Container specific configuration
lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1
lxc.rootfs.path = dir:/var/lib/lxc/ctn3/rootfs
lxc.uts.name = ctn3

# Network configuration
lxc.net.0.type = veth
lxc.net.0.link = lxcbr0
lxc.net.0.flags = up
lxc.net.0.ipv4.address = 10.10.0.3/24
```

Démarrer le conteneur

```
will@debian3:~$ sudo lxc-start -n ctn3
will@debian3:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4 IPV6 UNPRIVILEGED
ctn3 RUNNING 0         -      -    -    false
will@debian3:~$
```

Arrêter et désactiver le service systemd-networkd afin que le conteneur puisse conserver vos paramètres IP puis rebooter le conteneur

```
will@debian3:~$ sudo lxc-attach -n ctn3
root@ctn3:/# systemctl stop systemd-networkd
Warning: Stopping systemd-networkd.service, but it can still be activated by:
  systemd-networkd.socket
root@ctn3:/# systemctl disable systemd-networkd
Removed /etc/systemd/system/sockets.target.wants/systemd-networkd.socket.
Removed /etc/systemd/system/network-online.target.wants/systemd-networkd-wait-on
line.service.
Removed /etc/systemd/system/dbus-org.freedesktop.network1.service.
Removed /etc/systemd/system/multi-user.target.wants/systemd-networkd.service.
root@ctn3:/# reboot
root@ctn3:/#
will@debian3:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
ctn3 RUNNING 0         -      10.10.0.3 -    false
will@debian3:~$
```

Etape 5 : configurer la seconde Machine

Maintenant que vous avez un réseau fonctionnel reprendre les étapes précédentes sur la seconde machine en adaptant les configurations réseaux. Ce qui devrait donner un résultat similaire à ceci :

```
will@debian2:~$ sudo ovs-vsctl show
edcf2b83-1d10-4a4f-ae05-cef0b15c4095
    Bridge ovsbr0
        Port vxlan1
            Interface vxlan1
                type: vxlan
                options: {remote_ip="192.168.234.5"}
        Port lxcbr0
            tag: 100
            Interface lxcbr0
        Port ovsbr0
            Interface ovsbr0
                type: internal
    ovs_version: "2.15.0"
will@debian2:~$ sudo lxc-create -n ctn2 -t download -- -d debian -r bullseye -a
amd64
```

Etape 6 : effectuer un test de liaison

Une fois les configurations terminées, tester un ping sur chacun des conteneurs :

```
will@debian2:~$ sudo lxc-attach -n ctn2 -- ping -c3 10.10.0.3
PING 10.10.0.3 (10.10.0.3) 56(84) bytes of data.
64 bytes from 10.10.0.3: icmp_seq=1 ttl=64 time=6.69 ms
64 bytes from 10.10.0.3: icmp_seq=2 ttl=64 time=1.78 ms
64 bytes from 10.10.0.3: icmp_seq=3 ttl=64 time=1.82 ms

--- 10.10.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 1.777/3.428/6.690/2.306 ms
will@debian2:~$
```

```
will@debian2:~$ sudo lxc-attach -n ctn2 -- ping -c3 10.10.0.3
PING 10.10.0.3 (10.10.0.3) 56(84) bytes of data.
64 bytes from 10.10.0.3: icmp_seq=1 ttl=64 time=6.69 ms
64 bytes from 10.10.0.3: icmp_seq=2 ttl=64 time=1.78 ms
64 bytes from 10.10.0.3: icmp_seq=3 ttl=64 time=1.82 ms

--- 10.10.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2012ms
rtt min/avg/max/mdev = 1.777/3.428/6.690/2.306 ms
will@debian2:~$
```

Exercice :

En vous inspirant des étapes ci-dessus et du schéma d'architecture initial, mettez en place l'autre configuration qui passera par un port nommé lxcbr1 auquel vous ajouterez le tag 200. Assurez-vous que les nouveaux conteneurs seront rattachés à ce port et pourront communiquer chacun entre eux afin d'obtenir le résultat ci-dessous :

Debian 3 :

```
will@debian3:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
ctn3 RUNNING 0         -      10.10.0.3 -    false
ctn4 RUNNING 0         -      10.10.0.4 -    false
will@debian3:~$ sudo lxc-attach -n ctn4 -- ping -c3 10.10.0.1
PING 10.10.0.1 (10.10.0.1) 56(84) bytes of data.
64 bytes from 10.10.0.1: icmp_seq=1 ttl=64 time=3.51 ms
64 bytes from 10.10.0.1: icmp_seq=2 ttl=64 time=1.83 ms
64 bytes from 10.10.0.1: icmp_seq=3 ttl=64 time=1.79 ms

--- 10.10.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 1.793/2.376/3.509/0.801 ms
will@debian3:~$
```

Debian 2 :

```
will@debian2:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
ctn1 RUNNING 0         -      10.10.0.1 -    false
ctn2 RUNNING 0         -      10.10.0.2 -    false
will@debian2:~$ sudo lxc-attach -n ctn1 -- ping -c3 10.10.0.4
PING 10.10.0.4 (10.10.0.4) 56(84) bytes of data.
64 bytes from 10.10.0.4: icmp_seq=1 ttl=64 time=2.86 ms
64 bytes from 10.10.0.4: icmp_seq=2 ttl=64 time=1.87 ms
64 bytes from 10.10.0.4: icmp_seq=3 ttl=64 time=1.75 ms

--- 10.10.0.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 1.753/2.160/2.858/0.495 ms
will@debian2:~$
```

ANNEXE

Quelques commandes LXC utiles

lxc-stop -n ctn1 > Stoppe ctn1

lxc-snapshot -n ctn1 > Crée un snapshot de ctn1

lxc-destroy -s -n ctn1 > Détruit ctn1 et ses snapshots

lxc-info -n ctn1 > Fournit des informations sur ctn1

lxc-copy -n ctn1 > Crée un clone de ctn1

sudo lxc-start -n ctn1 --logfile=ctn1-start.log --logpriority=INFO > créer un fichier de log ctn1-start.log dans le répertoire courant au démarrage du conteneur afin de pouvoir débugger

sudo brctl addbr ovsbr2 > crée l'interface ovsbr2