Cours Virtualisation des réseaux

Nantes Ynov Campus – 2022-2023

Activité Pratique 5
Installation et configuration d'un commutateur virtuel Open vSwitch

Objectifs :

⇨ Découvrir et essayer pratiquement un commutateur virtuel

Pré requis

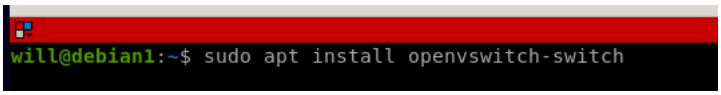
Pour réaliser ce TP vous devrez avoir terminé l'activité pratique sur l'installation du routeur logiciel pfsense et disposer d'une machine Debian 11 fraichement installée avec les configurations suivantes :

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

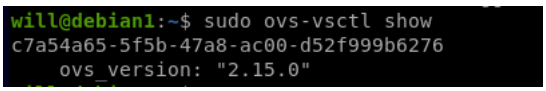
La machine Debian1 devra avoir un accès internet pour télécharger les paquets nécessaires à l'installation du switch.

Etape 1 : Installation et configuration d'open vSwitch

Installez le paquet openvswitch-switch :



Contrôlez maintenant la version d'Open vSwitch



- - Liste des principaux composants OVS installés - -

/usr/lib/openvswitch-common/ovs-vswitchd :
Service de commutation *(compatible OpenFlow)*.

/usr/bin/ovs-appctl :
Configuration CLI *(Command Line Interface)* du service.

/usr/bin/ovs-vsctl :
Configuration CLI du service via le serveur de Bdd OVS.

/usr/sbin/ovsdb-server :
Bdd contenant la configuration au niveau commutateur.

/usr/bin/ovsdb-client :
Outil de dialogue CLI avec le serveur de Bdd OVS.

/usr/bin/ovsdb-tool :
Configuration CLI des fichiers de la Bdd.

/usr/bin/ovs-dpctl :

Configuration CLI du module noyau d'Open vSwitch.

/usr/bin/ovs-ofctl :
Utilitaire CLI de contrôle des commutateurs OpenFlow.

La Bdd conf.db se situe dans /etc/openvswitch/
Les logs se situent dans /var/log/openvswitch/

Contrôlez le bon chargement du module :

```
will@debian1:~$ lsmod | grep openvswitch
openvswitch           167936  0
nsh                    16384  1 openvswitch
nf_conncount           24576  1 openvswitch
nf_nat                 57344  1 openvswitch
nf_conntrack          176128  3 nf_nat,openvswitch,nf_conncount
nf_defrag_ipv6         24576  2 nf_conntrack,openvswitch
libcrc32c              16384  3 nf_conntrack,nf_nat,openvswitch
```

Contrôlez l'activation du service openvswitch-switch :

```
will@debian1:~$ sudo systemctl status openvswitch-switch
● openvswitch-switch.service - Open vSwitch
     Loaded: loaded (/lib/systemd/system/openvswitch-switch.service; enabled; vendor preset: enabled)
     Active: active (exited) since Mon 2023-01-23 01:38:32 CET; 3min 29s ago
    Process: 2100 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 2100 (code=exited, status=0/SUCCESS)
        CPU: 4ms

janv. 23 01:38:32 debian1 systemd[1]: Starting Open vSwitch...
janv. 23 01:38:32 debian1 systemd[1]: Finished Open vSwitch.
```

Ainsi que celle des 2 services suivants : ovsdb-server , ovs-vswitchd

Etape 2 : Configuration de l'interface

Commencez par créer un bridge *(switch)* de nom br0 :

```
will@debian1:~$ sudo ovs-vsctl add-br br0
will@debian1:~$ sudo ovs-vsctl show
c7a54a65-5f5b-47a8-ac00-d52f999b6276
    Bridge br0
        Port br0
            Interface br0
                type: internal
    ovs_version: "2.15.0"
```

Un port et une interface virtuelle de même nom ont été associés au bridge. La commande del-br permet de supprimer l'interface

```
will@debian1:~$ 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.6/24 brd 192.168.1.255 scope global dynamic noprefixroute ens33
       valid_lft 85523sec preferred_lft 85523sec
    inet6 fe80::20c:29ff:fedd:af68/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 4a:ee:04:19:09:52 brd ff:ff:ff:ff:ff:ff
4: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 8a:e6:46:85:a1:42 brd ff:ff:ff:ff:ff:ff
will@debian1:~$
```

Rattachez à présent l'interface ens33 au bridge br0 :

```
will@debian1:~$ sudo ovs-vsctl add-port br0 ens33
will@debian1:~$ sudo ovs-vsctl show
c7a54a65-5f5b-47a8-ac00-d52f999b6276
    Bridge br0
        Port br0
            Interface br0
                type: internal
        Port ens33
            Interface ens33
    ovs_version: "2.15.0"
```

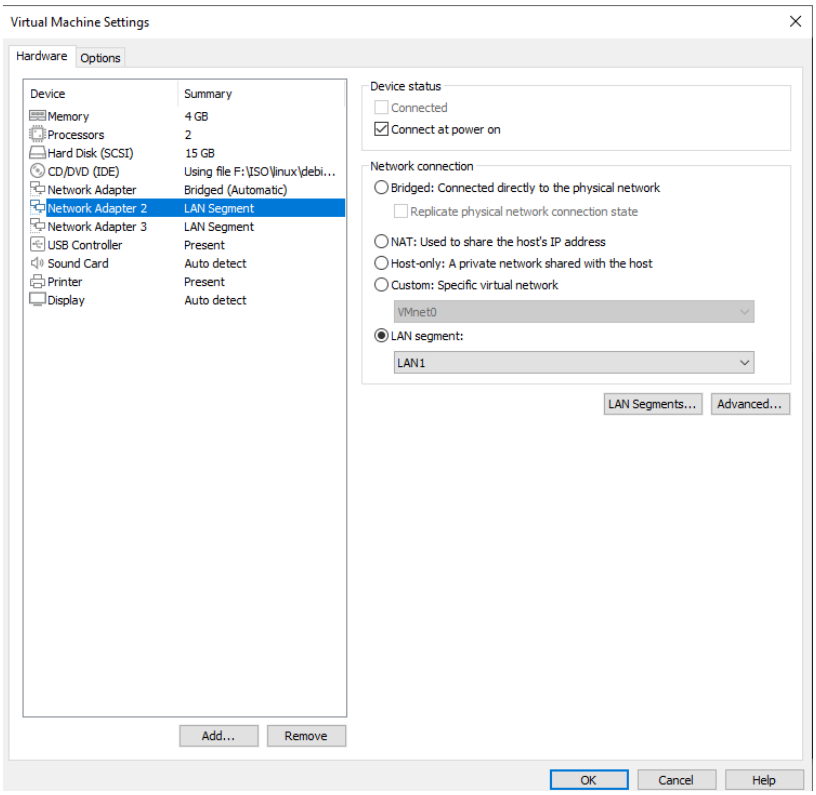
Le port virtuel ens33 et l'interface de réseau virtuel ens33 ne font plus qu'un.
Observez de nouveau le retour de la commande ip a :

```
will@debian1:~$ 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.1.6/24 brd 192.168.1.255 scope global dynamic noprefixroute ens33
       valid_lft 85224sec preferred_lft 85224sec
    inet6 fe80::20c:29ff:fedd:af68/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 4a:ee:04:19:09:52 brd ff:ff:ff:ff:ff:ff
4: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
will@debian1:~$
```

L'interface br0 possède à présent la même adresse MAC que ens33.

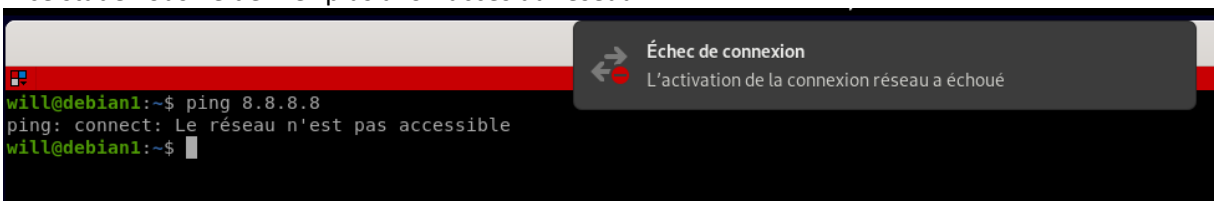
Etape 3 : Intégration de la VM ovs dans le réseau virtuel

Arrêter la VM Debian1 puis ajoutez lui deux cartes réseaux chacune appartenant à un LAN segment bien distinct comme indiqué dans le schéma d'architecture de départ :



Redémarrez la VM

A ce stade vous ne devriez plus avoir accès au réseau



Editez pour cela le fichier réseau interfaces :

```
will@debian1:~$ sudo nano /etc/network/interfaces
```

Et modifiez-le comme ci-dessous en faisant bien sur correspondre les adresses avec votre réseau :

```
  GNU nano 5.4                                          /etc/network/interfaces *
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# Configuration Open vSwitch
# Activation de l'interface br0
auto br0
allow-ovs br0

# Configuration IP de l'interface br0
iface br0 inet static
address 192.168.1.81
netmask 255.255.255.0
gateway 192.168.1.254
ovs_type OVSBridge
ovs_ports ens33

# Attachement du port/interface ens33 au bridge br0
allow-br0 ens33
iface ens33 inet manual
ovs_bridge br0
ovs_type OVSPort
```

Redémarrez ensuite le service réseau :

```
will@debian1:~$ sudo systemctl restart networking
will@debian1:~$
```

Contrôlez le résultat avec la commande ip a :

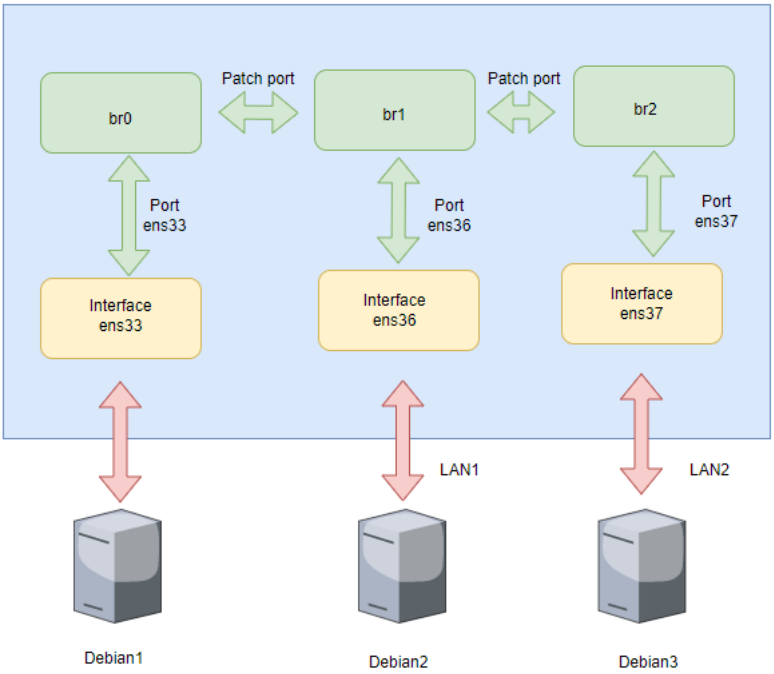
```
will@debian1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
3: ens36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:72 brd ff:ff:ff:ff:ff:ff
    altname enp2s4
4: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:7c brd ff:ff:ff:ff:ff:ff
    altname enp2s5
    inet6 fe80::20c:29ff:fedd:af7c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 4a:ee:04:19:09:52 brd ff:ff:ff:ff:ff:ff
6: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.81/24 brd 192.168.1.255 scope global br0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fedd:af68/64 scope link
       valid_lft forever preferred_lft forever
```

Constat :

- 2 nouvelles interfaces réseau ens36 et ens37
- L'interface br0 a la même adresse MAC que ens33
- L'interface br0 possède l'adresse IP 192.168.1.81
- Le ping vers l'adresse lP 192.168.1.254 fonctionne

Raccordement des 2 clients Debian sur OVS

Vous lierez les 3 bridges à l'aide de ports patch comme schématisé ci-dessous :



L'ensemble ainsi raccordé *(ports patch)* peut être vu comme un seul pont.
Créez les bridges br1 et br2 :

```
will@debian1:~$ sudo ovs-vsctl add-br br1
[sudo] Mot de passe de will :
will@debian1:~$ sudo ovs-vsctl add-br br2
will@debian1:~$
```

Reliez maintenant br0 avec br1 :

sudo ovs-vsctl -- add-port br0 br0-patch0 -- set interface br0-patch0 type=patch options:peer=br1-patch0

sudo ovs-vsctl -- add-port br1 br1-patch0 -- set interface br1-patch0 type=patch options:peer=br0-patch0

```
will@debian1:~$ sudo ovs-vsctl -- add-port br0 br0-patch0 -- set interface br0-patch0 type=patch options:peer=br1-patch0
will@debian1:~$ sudo ovs-vsctl -- add-port br1 br1-patch0 -- set interface br1-patch0 type=patch options:peer=br0-patch0
will@debian1:~$
```

Ensuite br1 avec br2 :

sudo ovs-vsctl -- add-port br1 br1-patch1 -- set interface br1-patch1 type=patch options:peer=br2-patch0

sudo ovs-vsctl -- add-port br2 br2-patch0 -- set interface br2-patch0 type=patch options:peer=br1-patch1

```
will@debian1:~$ sudo ovs-vsctl -- add-port br1 br1-patch1 -- set interface br1-patch1 type=patch options:peer=br2-patch0
will@debian1:~$ sudo ovs-vsctl -- add-port br2 br2-patch0 -- set interface br2-patch0 type=patch options:peer=br1-patch1
will@debian1:~$
```

Vérifiez la prise en compte de la configuration :

```
will@debian1:~$ sudo ovs-vsctl show
c7a54a65-5f5b-47a8-ac00-d52f999b6276
    Bridge br2
        Port br2-patch0
            Interface br2-patch0
                type: patch
                options: {peer=br1-patch1}
        Port br2
            Interface br2
                type: internal
    Bridge br0
        Port br0
            Interface br0
                type: internal
        Port ens33
            Interface ens33
        Port br0-patch0
            Interface br0-patch0
                type: patch
                options: {peer=br1-patch0}
    Bridge br1
        Port br1-patch0
            Interface br1-patch0
                type: patch
                options: {peer=br0-patch0}
        Port br1-patch1
            Interface br1-patch1
                type: patch
                options: {peer=br2-patch0}
        Port br1
            Interface br1
                type: internal
    ovs_version: "2.15.0"
will@debian1:~$
```

Editez la configuration OVS du fichier interfaces :

```
will@debian1:~$ sudo nano /etc/network/interfaces
```

```
#Configuration IP interface br0
iface br0 inet static
        address 192.168.1.81
        netmask 255.255.255.0
        gateway 192.168.1.254
        ovs_type OVSBridge
        ovs_ports ens33 br0-patch0
#Attachement du port/interface ens33 au bridge br0
allow-br0 ens33
iface ens33 inet manual
        ovs_bridge br0
        ovs_type OVSPort

#Liaison patch br0 vers br1
allow-br0 br0-patch0
iface br0-patch0 inet manual
        ovs_bridge br0
        ovs_type OVSPatchPort
        ovs_patch_peer br1-patch0



#Configuration du bridge br1
auto br1
allow-ovs br1
iface br1 inet manual
        ovs_type OVSBridge
        ovs_ports ens36 br1-patch0 br1-patch1

allow-br1 ens36
iface ens36 inet manual
        ovs_bridge br1
        ovs_type OVSPort
```

```
allow-br1 br1-patch0
iface br1-patch0 inet manual
        ovs_bridge br1
        ovs_type OVSPatchPort
        ovs_patch_peer br0-patch0

#Liaison patch br1 vers br2
allow-br1 br1-patch1
iface br1-patch1 inet manual
        ovs_bridge br1
        ovs_type OVSPatchPort
        ovs_patch_peer br2-patch0

#Configuration du bridge br2
auto br2
allow-ovs br2
iface br2 inet manual
        ovs_type OVSBridge
        ovs_ports ens37 br2-patch0

allow-br2 ens37
iface ens37 inet manual
        ovs_bridge br2
        ovs_type OVSPort

allow-br2 br2-patch0
iface br2-patch0 inet manual
        ovs_bridge br2
        ovs_type OVSPatchPort
        ovs_patch_peer br1-patch1
```

Redémarrez la VM pour appliquer la configuration puis contrôlez le statut du service réseau :

```
will@debian1:~$ sudo systemctl status networking
● networking.service - Raise network interfaces
     Loaded: loaded (/lib/systemd/system/networking.service; enabled; vendor pr
     Active: active (exited) since Mon 2023-01-23 02:53:51 CET; 2min 3s ago
       Docs: man:interfaces(5)
    Process: 672 ExecStart=/sbin/ifup -a --read-environment (code=exited, statu
   Main PID: 672 (code=exited, status=0/SUCCESS)
        CPU: 1.292s

janv. 23 02:53:50 debian1 systemd[1]: Starting Raise network interfaces...
janv. 23 02:53:50 debian1 ovs-vsctl[718]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:50 debian1 ovs-vsctl[799]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:50 debian1 ovs-vsctl[811]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:50 debian1 ovs-vsctl[848]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:51 debian1 ovs-vsctl[886]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:51 debian1 ovs-vsctl[949]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:51 debian1 ovs-vsctl[961]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:51 debian1 ovs-vsctl[998]: ovs|00001|vsctl|INFO|Called as ovs-vs
janv. 23 02:53:51 debian1 systemd[1]: Finished Raise network interfaces.
```

Contenu de la configuration OVS

Etape 4 : Test de bon fonctionnement du switch virtuel

Vérifiez à l'aide de la commande ping la conformité des résultats avec ceux indiqués sur la maquette réseau local virtuel.
Stoppez ensuite la VM ovs support d'Open vSwitch et assurez-vous que les 2 clients Debian ne peuvent plus communiquer entre eux.

Partie 2 : Installation de conteneur LXC

Etape 1 : Installation

```
will@debian1:~$ sudo apt install lxc lxc-templates
```

Vérifiez ensuite l'activation du routage au sein de la VM :

```
will@debian1:~$ sudo sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

Vérifiez aussi la bonne activation des services LXC :

```
will@debian1:~$ sudo systemctl status lxc
● lxc.service - LXC Container Initialization and Autoboot Code
     Loaded: loaded (/lib/systemd/system/lxc.service; enabled; vendor preset: enabled)
     Active: active (exited) since Mon 2023-01-23 14:37:02 CET; 1min 49s ago
       Docs: man:lxc-autostart
             man:lxc
    Process: 3103 ExecStartPre=/usr/libexec/lxc/lxc-apparmor-load (code=exited, status=0/SUCCESS)
    Process: 3112 ExecStart=/usr/libexec/lxc/lxc-containers start (code=exited, status=0/SUCCESS)
   Main PID: 3112 (code=exited, status=0/SUCCESS)
        CPU: 83ms

janv. 23 14:37:02 debian1 systemd[1]: Starting LXC Container Initialization and Autoboot Code...
janv. 23 14:37:02 debian1 systemd[1]: Finished LXC Container Initialization and Autoboot Code.
will@debian1:~$ sudo systemctl status lxc-net
● lxc-net.service - LXC network bridge setup
     Loaded: loaded (/lib/systemd/system/lxc-net.service; enabled; vendor preset: enabled)
     Active: active (exited) since Mon 2023-01-23 14:37:02 CET; 2min 1s ago
       Docs: man:lxc
    Process: 3039 ExecStart=/usr/libexec/lxc/lxc-net start (code=exited, status=0/SUCCESS)
   Main PID: 3039 (code=exited, status=0/SUCCESS)
      Tasks: 1 (limit: 4617)
     Memory: 1.8M
        CPU: 86ms
     CGroup: /system.slice/lxc-net.service
             └─3101 dnsmasq --conf-file=/dev/null -u dnsmasq --strict-order --bind-interfaces --pid-file=/run/lxc/dnsmasq.pid --listen-address 10.0.3.1 --dhcp-ra>

janv. 23 14:37:02 debian1 systemd[1]: Starting LXC network bridge setup...
janv. 23 14:37:02 debian1 dnsmasq[3101]: demarré, version 2.85 (taille de cache 150)
janv. 23 14:37:02 debian1 dnsmasq[3101]: options à la compilation : IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP conntrack ipset auth cryptohas>
janv. 23 14:37:02 debian1 dnsmasq-dhcp[3101]: DHCP, plage d'adresses IP 10.0.3.2 -- 10.0.3.254, durée de bail 1h
janv. 23 14:37:02 debian1 dnsmasq-dhcp[3101]: DHCP, sockets bound exclusively to interface lxcbr0
janv. 23 14:37:02 debian1 dnsmasq[3101]: Lecture de /etc/resolv.conf
janv. 23 14:37:02 debian1 dnsmasq[3101]: utilise le serveur de nom 192.168.1.254#53
janv. 23 14:37:02 debian1 dnsmasq[3101]: lecture /etc/hosts - 5 adresses
janv. 23 14:37:02 debian1 systemd[1]: Finished LXC network bridge setup.
lines 1-21/21 (END)
```

Vérifiez la nouvelle configuration réseau :
Un nouveau bridge de nom lxcbr0 a fait son apparition

```
will@debian1:~$ 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: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet6 fe80::20c:29ff:fedd:af68/64 scope link
       valid_lft forever preferred_lft forever
3: ens36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:72 brd ff:ff:ff:ff:ff:ff
    altname enp2s4
    inet6 fe80::20c:29ff:fedd:af72/64 scope link
       valid_lft forever preferred_lft forever
4: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master ovs-system state UP group default qlen 1000
    link/ether 00:0c:29:dd:af:7c brd ff:ff:ff:ff:ff:ff
    altname enp2s5
    inet6 fe80::20c:29ff:fedd:af7c/64 scope link
       valid_lft forever preferred_lft forever
5: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 4a:ee:04:19:09:52 brd ff:ff:ff:ff:ff:ff
6: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:dd:af:68 brd ff:ff:ff:ff:ff:ff
    inet 192.168.110.81/24 brd 192.168.110.255 scope global br0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fedd:af68/64 scope link
       valid_lft forever preferred_lft forever
7: br1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:dd:af:72 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fedd:af72/64 scope link
       valid_lft forever preferred_lft forever
8: br2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:dd:af:7c brd ff:ff:ff:ff:ff:ff
    inet6 fe80::20c:29ff:fedd:af7c/64 scope link
       valid_lft forever preferred_lft forever
9: 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
will@debian1:~$
```

Ce bridge fourni par défaut avec LXC sera non exploité au profit des bridges br0 et br2 d'OVS.

Etape 2 : Création d'un conteneur LXC

Les configurations LXC seront stockées dans /etc/lxc qui contient pour l'instant un unique default.conf.

Des modèles de configuration sont disponibles dans /usr/share/doc/lxc/examples/.

La création de conteneurs peut être réalisée à l'aide de templates situés dans /usr/share/lxc/templates/ ou en téléchargeant une distribution spécifique sur Internet.

Créez le conteneur de nom ctn1

sudo lxc-create -n ctn1 -t download -- -d debian -r bullseye -a amd64

```
will@debian1:~$ sudo lxc-create -n ctn1 -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 (20230123_07:03) container.

To enable SSH, run: apt install openssh-server
No default root or user password are set by LXC.
will@debian1:~$
```

Vérifiez la création et le statut stoppé du conteneur :

```
will@debian1:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4 IPV6 UNPRIVILEGED
ctn1 STOPPED 0         -      -    -    false
```

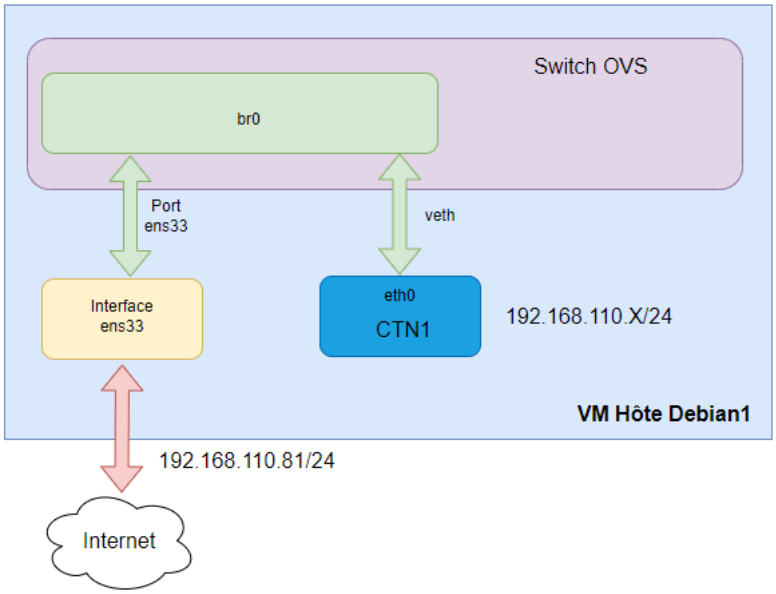
Afficher sa configuration par défaut :

```
will@debian1:~$ sudo cat /var/lib/lxc/ctn1/config
# Template used to create this container: /usr/share/lxc/templates/lxc-download
# Parameters passed to the template: -d debian -r bullseye -a amd64
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (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/ctn1/rootfs
lxc.uts.name = ctn1

# Network configuration
lxc.net.0.type = veth
lxc.net.0.link = lxcbr0
lxc.net.0.flags = up
will@debian1:~$
```

Etape 3 : Configuration réseau du conteneur

LXC configure de base une connexion virtuelle de type veth pour raccorder ctn1 au bridge lxcbr0. Le type veth permet de créer une interface réseau sur le bridge lxcbr0 et une autre sur ctn1.
La configuration de ctn1 doit être modifiée pour raccorder celui-ci sur le bridge br0 d'OVS.

Configuration à créer :



Editez la configuration du conteneur et modifiez la partie réseau comme ci-dessous :

```
# Template used to create this container: /usr/share/lxc/templates/lx>
# Parameters passed to the template: -d debian -r bullseye -a amd64
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (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/ctn1/rootfs
lxc.uts.name = ctn1

# Network configuration
lxc.net.0.type = veth
lxc.net.0.link = lxcbr0
lxc.net.0.flags = up
lxc.net.0.script.up = /etc/lxc/ovsup-ctn1
lxc.net.0.script.down = /etc/lxc/ovsdown-ctn1
```

Créez ensuite le script ovsup-ctn1 attachera celui-ci à OVS au démarrage

```
will@debian1:~$ sudo nano /etc/lxc/ovsup-ctn1
```

```
#!/bin/bash

BRIDGE="br0"
ovs-vsctl --may-exist add-br $BRIDGE
ovs-vsctl --if-exists del-port $BRIDGE $5
ovs-vsctl --may-exist add-port $BRIDGE $5
```

Créez le script ovsdown-ctn1 qui détachera celui-ci d'OVS à l'arrêt du conteneur

```
#!/bin/bash

BRIDGE="br0"
ovs-vsctl --if-exists del-port $BRIDGE $5
```

Rendez les 2 scripts exécutables : sudo chmod +x /etc/lxc/ovs*

```
will@debian1:~$ sudo chmod +x /etc/lxc/ovs*
will@debian1:~$
```

Etape 4 : Démarrage du conteneur

Pour démarrer ctn1, lancez la Cde suivante : sudo lxc-start -n ctn1

```
will@debian1:~$ sudo lxc-start -n ctn1
will@debian1:~$
```

Vérifiez ensuite le statut du conteneur : sudo lxc-ls -f

```
will@debian1:~$ sudo lxc-start -n ctn1
will@debian1:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4            IPV6 UNPRIVILEGED
ctn1 RUNNING 1         -      192.168.110.147 -    false
```

Vérifiez aussi la création d'une interface réseau veth :

```
9: lxcbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master ovs-s
ystem state UP 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
    inet6 fe80::216:3eff:fe00:0/64 scope link
       valid_lft forever preferred_lft forever
13: vethDHXf0n@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue mas
ter lxcbr0 state UP group default qlen 1000
    link/ether fe:07:40:30:9a:4a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::fc07:40ff:fe30:9a4a/64 scope link
       valid_lft forever preferred_lft forever
will@debian1:/etc/lxc$
```

Etape 5 : Exécution de commande à l'intérieur du conteneur

Le conteneur a été créé par défaut sans MDP root.
La Cde lxc-attach permet d'exécuter des Cdes dans un conteneur, ceci en tant que root.
Utilisez celle-ci pour vous connecter sur ctn1 :

```
will@debian1:/etc/lxc$ sudo lxc-attach -n ctn1
root@ctn1:/#
```

Vérifiez ensuite que vous arrivez bien à faire un ping sur les adresses de votre réseau et que vous avez accès à internet.

Etape 6 : Démarrage automatique du conteneur

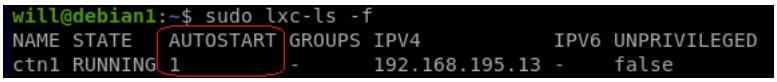
Il est possible de démarrer automatiquement ctn1 au boot de l'hôte LXC soit la VM ovs.
Pour cela, éditez la configuration du conteneur :

```
will@debian1:/etc/lxc$ sudo nano /var/lib/lxc/ctn1/config
```

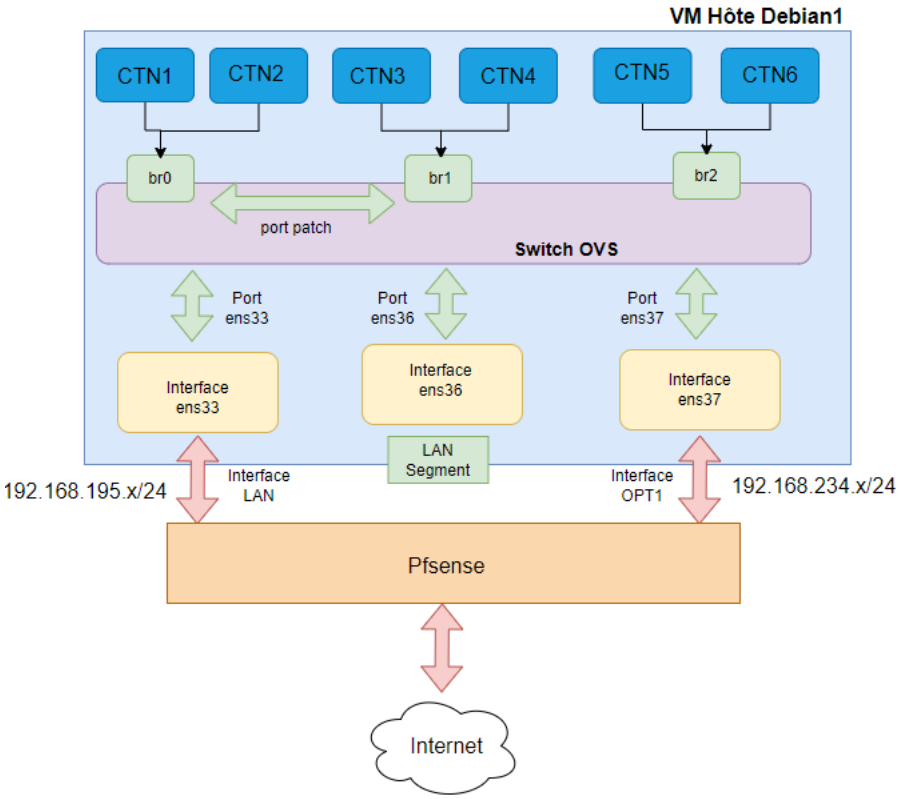
Ajoutez ce contenu à la fin du fichier :

```
# Template used to create this container: /usr/share/lxc/templates/lx>
# Parameters passed to the template: -d debian -r bullseye -a amd64
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (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/ctn1/rootfs
lxc.uts.name = ctn1

# Network configuration
lxc.net.0.type = veth
lxc.net.0.link = lxcbr0
lxc.net.0.flags = up
lxc.net.0.script.up = /etc/lxc/ovsup-ctn1
lxc.net.0.script.down = /etc/lxc/ovsdown-ctn1

# Démarrage auto du conteneur au bout de 10s
lxc.start.auto = 1
lxc.start.delay = 10
```

Puis rebootez la VM ovs et contrôlez le statut de ctn1 : sudo lxc-ls -f


will@debian1:~$ sudo lxc-ls -f
NAME STATE AUTOSTART GROUPS IPV4 IPV6 UNPRIVILEGED
ctn1 RUNNING 1 - 192.168.195.13 - false

Exercice :



1- Pour la suite, créer un second conteneur qui sera aussi rattaché à l'interface br0
2- Créer deux autres conteneurs qui seront rattachés à l'interface br1
3- Couper la liaison effectuée entre l'interface br1 et br2
4- Créer deux conteneurs qui seront attachés à l'interface br2
5- Brancher l'interface LAN de votre routeur pfsense sur votre interface Br0 et vérifiez que vos conteneurs (1, 2, 3 et 4) reçoivent bien une adresse dhcp de votre interface LAN pfsense.

```
will@debian1:~$ sudo lxc-ls -f
NAME STATE   AUTOSTART GROUPS IPV4           IPV6 UNPRIVILEGED
ctn1 RUNNING 1         -      192.168.195.13 -    false
ctn2 RUNNING 0         -      192.168.195.15 -    false
ctn3 RUNNING 0         -      192.168.195.16 -    false
ctn4 RUNNING 0         -      192.168.195.17 -    false
ctn5 RUNNING 0         -      192.168.234.5  -    false
ctn6 RUNNING 0         -      192.168.234.6  -    false
```

6- Brancher l'interface OPT1 de votre routeur pfsense sur l'interface br2 et vérifier que vos conteneurs (5 et 6) reçoivent bien une @IP de votre routeur
7- Vérifier que vos conteneurs ont tous bien accès à internet

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 débuger