TP4 – Bridges

Ce TP ne suit pas exactement tout le sujet. Des parties n'ont pas été faites par manque de temps ou de compréhension du sujet. Néanmoins il permet de comprendre le fonctionnement des réseaux avec des bridges.

Bridge au démarrage

1. Mise en place

Installation des Alpines

Reprendre le TP3, il faut cependant changer les scripts avec ceux qui suivent. En effet, pour que les machines puissent communiquer il faut qu'elles aient des adresses MAC différentes. Or, puisqu'elles sont des clones d'une même machine elles ont de base la même adresse. La ligne en gras permet donc de leur attribuer une adresse mac différente.

Voici donc les scripts de démarrage de mes VM Alpine :

Clone 1	*Clone 2*

`sudo nano clone1.sh`
 	
`qemu-system-x86_64 \`
`  -k fr \`
`  -vga virtio \`
`  -m 512 \`
`  -nic`
`tap,ifname=tap0,mac=02:ca:fe:f0:0d:01`
`\`
`  -boot d \`
`  -drive file=clone1.qcow2,if=virtio \`
`  -net nic \`
`  -display curses`
	`sudo nano clone2.sh`
	
	`qemu-system-x86_64 \`
	`  -k fr \`
	`  -vga virtio \`
	`  -m 512 \`
	`  -nic`
	`tap,ifname=tap1,mac=02:ca:fe:f0:0d:02`
	`\`
	`  -boot d \`
	`  -drive file=clone1.qcow2,if=virtio \`
	`  -net nic \`
	`  -display curses`

Installation des deux conteneurs LXC Debian

Nous avions créé des conteneurs LXC pendant le TP2 mais pas des Debian.
Afin de les créer j'ai repris le début de TP2. Cependant voici quelques modifications et informations utiles :

- La commande pour créer le conteneur est la suivante (sans l'option -buster l'installation ne fonctionne pas) :

```
sudo lxc-create -n debian -t debian -- -r buster
```

- Aussi, ma clef n'était plus à jour et l'installation ne pouvait donc pas se réaliser. J'ai suivi ce tuto et la création a fonctionné. La clef n'était plus valide depuis novembre 2021, il fallait donc télécharger la nouvelle clef qui est valable jusqu'en 2027.

Pour information, voici l'état de mes deux conteneurs après leur création et les lignes :

```
lxc-start -n debian
lxc-start -n debian2
```

```
NAME    STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
debian  RUNNING 0         -      10.0.4.53 -    false
debian2 RUNNING 0         -      10.0.4.86 -    false
```

Pour exécuter des commandes sur les conteneurs j'ai utilisé la commande :

```
lxc-attach -n debian -- la_commande
```

Il m'a été nécessaire de télécharger le paquet permettant l'utilisation de ping. En effet cette commande est nécessaire pour la suite du TP. Ce paquet n'est pas installé de base sur les conteneurs Debian. Pour l'installer utiliser la commande :

```
lxc-attach -n debian -- apt install iputils-ping
```

En fonction des préférences cette installation sera utile :

```
lxc-attach -n debian -- apt-get install nano
```

2. Configuration réseau privé hôte (private bridge) avec IP fixes

a. Configurez br0, un bridge au démarrage, ne comprenant aucune interface.

J'ai modifié le fichier « `/etc/netplan/00-installer-config.yaml` » pour donner une IP fixe à mon bridge br0 et lui attribuer aucune interface :

```
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
      dhcp4: true
  bridges:
    br0:
      dhcp4: false
      addresses: [192.168.0.1/24]
      nameservers:
        addresses: [8.8.8.8]
  version: 2
```

```
^X
```

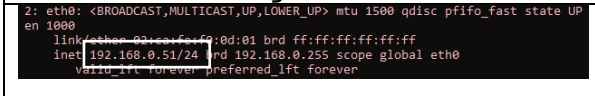
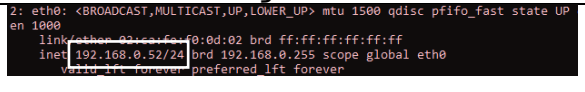
```
netplan apply
```

b. Démarrer les deux machines virtuelles Alpine Linux qui sont accessibles à l'aide d'un forward de port (SSH) et configurer leurs adresses IP statiques

J'ai modifié le fichier « `nano /etc/network/interfaces` » de chacune des deux machines Alpine ne me connectant dessus :

Clone 1	*Clone 2*

`nano /etc/network/interfaces`
 	
`auto lo`
`iface lo inet loopback`
 	
`auto eth0`
`iface eth0 inet static`
`address 192.168.0.51`
`netmask 255.255.255.0`
`network 192.168.0.0`
`broadcast 192.168.0.255`
`gateway 192.168.0.1`
`hostname alpine`
 	
`auto eth1`
`iface eth1 inet dhcp`
`hostname alpine`
`^X`
`service networking restart`
	`nano /etc/network/interfaces`
	
	`auto lo`
	`iface lo inet loopback`
	
	`auto eth0`
	`iface eth0 inet static`
	`address 192.168.0.52`
	`netmask 255.255.255.0`
	`network 192.168.0.0`
	`broadcast 192.168.0.255`
	`gateway 192.168.0.1`
	`hostname alpine`
	
	`auto eth1`
	`iface eth1 inet dhcp`
	`hostname alpine`
	`^X`
	`service networking restart`

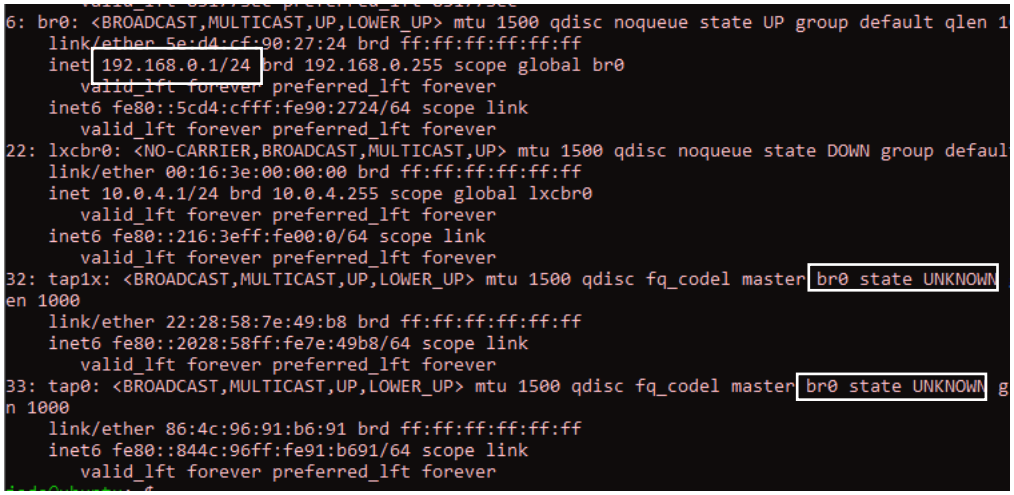
L'adresse de passerelle est donc l'IP attribuée au bridge. Ne pas oublier de vérifier que les modifications ont bien été prises en compte.

c. Les connecter à br0 par l'intermédiaire d'interfaces TAP

Puis j'ai connecté les interfaces tap au bridge br0 sur l'hôte avec les commandes :

```
ip link set dev br0 up
ip link set tap0 master br0
ip link set tap1 master br0
```

Voici ce que la commande « `ip a show` » a renvoyé sur l'hôte Ubuntu après ces configurations :



J'ai pu remarquer que l'adresse de passerelle était bien attribuée au bridge. Les interfaces appartiennent bien à br0, elles sont UP et le bridge également.

Remarque :
La commande « `ip link set dev br0 up` » laissera le bridge à `DOWN` tant qu'il ne sera pas master d'au moins une interface.

d. Tester les échanges possibles entre les différentes machines :

Entre…	Et …	Les échanges …
Le clone 1 Alpine	Le clone 2 Alpine	Les deux clones arrivent à communiquer entre elles. Cf. figure 1.
Les clones Alpine	L'hôte	Les clones peuvent communiquer avec les deux interfaces de l'hôte et l'hôte peut communiquer avec les invités.

Figure 1

e. Créez et configurez deux conteneurs Debian, tous deux en IP fixe, connectés à br0. Démarrez-les.

Avant de les démarrer, j'ai changé leur fichier de configurations «`/var/lib/lxc/`*`nom_machine`*`/config` » :

Debian	*Debian 2*
nano /var/lib/lxc/debian/config lxc.net.0.type = veth lxc.net.0.link = br0 lxc.net.0.flags = up lxc.net.0.hwaddr = 00:16:3e:50:97:92 lxc.rootfs.path = dir:/var/lib/lxc/debian/rootfs lxc.net.0.ipv4.address = 192.168.0.61/24 lxc.net.0.ipv4.gateway = 192.168.0.1 ^X	nano /var/lib/lxc/debian2/config lxc.net.0.type = veth lxc.net.0.link = br0 lxc.net.0.flags = up lxc.net.0.hwaddr = 00:16:3e:50:97:92 lxc.rootfs.path = dir:/var/lib/lxc/debian2/rootfs lxc.net.0.ipv4.address = 192.168.0.62/24 lxc.net.0.ipv4.gateway = 192.168.0.1 ^X

Ne pas oublier de relancer le service LXC :

```
service lxc stop
service lxc start
```

```
root@ubuntu:/home/jade# root@ubuntu:/home/jade# lxc-ls -f
NAME    STATE   AUTOSTART GROUPS IPV4         IPV6 UNPRIVILEGED
debian  RUNNING 0         -      192.168.0.61 -    false
debian2 RUNNING 0         -      192.168.0.62 -    false
```

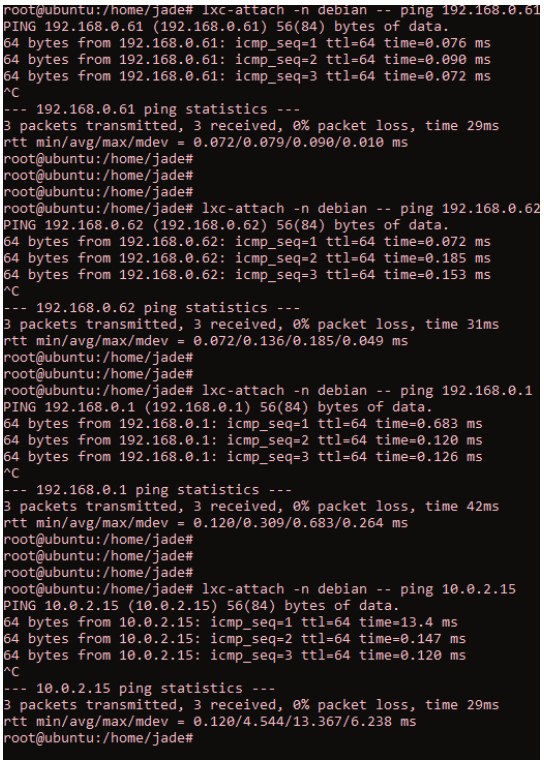
Je vérifie avec la ligne de commande « `ip a show` » leurs adresses IP ainsi que leur connexion au bridge br0.

```
6: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 5e:d4:cf:90:27:24 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 metric 100 brd 10.0.3.255 scope global dynamic br0
       valid_lft 81997sec preferred_lft 81997sec
    inet6 fe80::5cd4:cfff:fe90:2724/64 scope link
       valid_lft forever preferred_lft forever
22: 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 10.0.4.1/24 brd 10.0.4.255 scope global lxcbr0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe00:0/64 scope link
       valid_lft forever preferred_lft forever
28: vethJPEeLp@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br0 state UP group default qlen 1000
    link/ether fe:d1:4e:2b:5b:03 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::fcd1:4eff:fe2b:5b03/64 scope link
       valid_lft forever preferred_lft forever
29: vethrDe360@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br0 state UP group default qlen 1000
    link/ether fe:24:5d:1b:aa:f0 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::fc24:5dff:fe1b:aaf0/64 scope link
       valid_lft forever preferred_lft forever
```

ip a show sur la machine Ubuntu hôte

f. Tester les échanges possibles entre les différentes machines :

Entre…	Et …	Les échanges …
Le conteneur 1 Debian	Le conteneur 2 Debian	Fonctionnent
Les conteneurs Debian	L'hôte	Fonctionnent

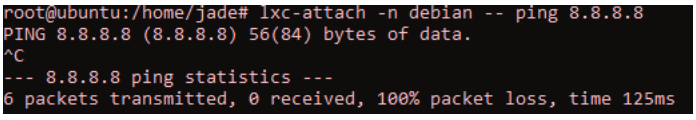


Les communications entre conteneur/conteneur ainsi que conteneurs/les deux interfaces de l'hôte fonctionnent

g. Relancer les deux machines Alpine et tester les échanges possibles entre les différentes machines :

Entre…	Et …	Les échanges …
Les conteneurs Debian	Les clones Alpine	Fonctionnent
Un invité Debian	L'hôte	Fonctionnent
Une invitée Alpine	L'hôte	Fonctionnent

Toutes les machines arrivent donc à communiquer entre elles. Cependant les conteneurs et les machines virtuelles ne peuvent pas accéder à internet.



Exemple d'un conteneur qui ne peut pas communiquer avec Google

3. Configuration pontée en DHCP

a. Configurer br1, un bridge au démarrage comprenant l'interface de l'hôte, celle configurée en NAT

J'ai modifié le fichier « `00-installer-config.yaml` » :

```
sudo nano /etc/netplan/00-installer-config.yaml
```

```
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
      dhcp4: false
  bridges:
```

```
    br0:
      dhcp4: true
    br1:
      dhcp4: true
      interfaces:
       -enp0s8
  version: 2
^X

  sudo netplan apply
```

Il est également possible de créer le bridge directement en ligne de commandes :

```
ip link add name br1 type bridge
```

Pour verifier:

```
brctl show
```

```
jade@ubuntu:~$ brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.5ed4cf902724       no
br1             8000.62234d582817       no              enp0s8
```

b. Démarrer les deux machines virtuelles Alpine Linux qui sont accessibles à l'aide d'un forward de port (SSH) et les connecter à br0 par l'intermédiaire d'interfaces TAP

J'ai lancé mes deux machines Alpine et j'ai tapé ces deux commandes dans mon hôte Ubuntu :

```
sudo ip link set tap0 up
sudo ip link set tap1 up
ip link set tap0 master br0
ip link set tap1 master br0
```

Il faut modifier le fichier « `/etc/network/interfaces` » pour mettre l'interface en DHCP et ne plus avoir une adresse statique.

Avec la commande « `ip a show` » j'ai pu récupérer leur adresse IP. Puisque les invités sont en DHCP leur adresse était :

- Clone 1 : 10.0.3.17
- Clone 2 : 10.0.3.18

Tester les échanges possibles entre les différentes machines :

Entre…	Et …	Les échanges …
Le clone 1 Alpine	Le clone 2 Alpine	Les deux clones peuvent communiquer. Cf. figure 2.
Les clones Alpine	L'hôte	Les clones ne peuvent pas communiquer avec l'hôte

```
alpine:~# ping 10.0.3.18
PING 10.0.3.18 (10.0.3.18): 56 data bytes
64 bytes from 10.0.3.18: seq=0 ttl=64 time=3.429 ms
64 bytes from 10.0.3.18: seq=1 ttl=64 time=4.140 ms
64 bytes from 10.0.3.18: seq=2 ttl=64 time=2.616 ms
^C
--- 10.0.3.18 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.616/3.395/4.140 ms
alpine:~#
```

```
alpine:~# ping 10.0.3.17
PING 10.0.3.17 (10.0.3.17): 56 data bytes
64 bytes from 10.0.3.17: seq=0 ttl=64 time=2.258 ms
64 bytes from 10.0.3.17: seq=1 ttl=64 time=2.664 ms
64 bytes from 10.0.3.17: seq=2 ttl=64 time=2.586 ms
^C
--- 10.0.3.17 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.258/2.502/2.664 ms
alpine:~#
```

Figure 2

c. *Configurer les deux conteneurs Debian, tous deux en DHCP, connectés à **br1**. Démarrer les deux conteneurs LXC Debian.*

J'ai changé leur fichier de configurations « `/var/lib/lxc/nom_machine/config` » :

Debian	*Debian 2*

`nano /var/lib/lxc/debian/config`
…
`lxc.net.0.type = veth`
`lxc.net.0.link = br1`
…
`^X`
	`nano /var/lib/lxc/debian2/config`
	…
	`lxc.net.0.type = veth`
	`lxc.net.0.link = br1`
	…
	`^X`

S'il y avait les lignes pour attribuer des IP fixes il faut les supprimer pour permettre au DHCP de fonctionner.

Ne pas oublier de relancer le service LXC :

```
service lxc stop
service lxc start
```

Il ne faut pas oublier de rallumer les conteneurs avec les commandes :

```
lxc-start -n debian
lxc-start -n debian2
```

On remarque que les adresses des conteneurs sont bien dans le même réseau que le bridge de l'hôte ainsi que dans le réseau des Alpines :

```
root@ubuntu:/home/jade# lxc-ls -f
NAME    STATE   AUTOSTART GROUPS IPV4       IPV6 UNPRIVILEGED
debian  RUNNING 0         -      10.0.3.19  -    false
debian2 RUNNING 0         -      10.0.3.20  -    false
```

d. *Vérifier l'accès à internet*

L'accès à internet fonctionne puisqu'il est possible de pinguer Google :

```
root@ubuntu:/home/jade# lxc-attach -n debian -- ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=118 time=21.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=118 time=34.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=118 time=12.3 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 26ms
rtt min/avg/max/mdev = 12.293/22.749/34.862/9.289 ms
root@ubuntu:/home/jade#
```

e. *Tester les échanges possibles entre les différentes machines :*

Entre…	Et …	Les échanges …
Le conteneur 1 Debian	Le conteneur 2 Debian	Fonctionnent cf. figure 3
Les conteneurs Debian	L'hôte	Les conteneurs peuvent communiquer avec l'interface du bridge de l'hôte, cf. figure 3.

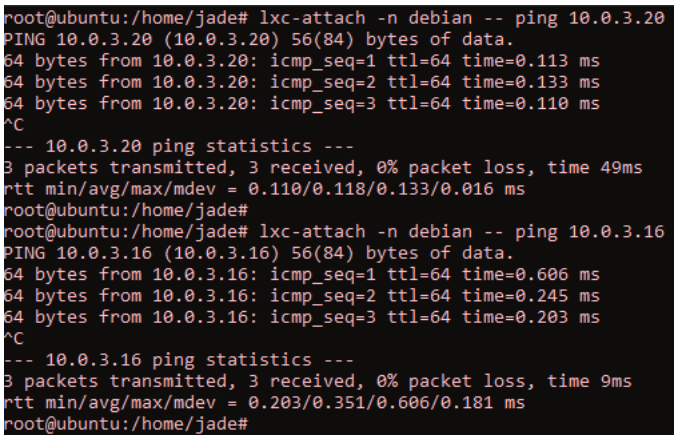


Figure 3

f. Relancer les deux machines Alpine et tester les échanges possibles entre les différentes machines :

Entre…	Et …	Les échanges …
Les conteneurs Debian	Les clones Alpine	Ne fonctionnent pas
Un invité Debian	L'hôte	Fonctionne avec l'interface de leur bridge
Une invitée Alpine	L'hôte	Ne fonctionnent pas

4. Open VSwitch

Attention les commandes du paquet **`bridge-utils`**, utilisées jusqu'à maintenant, ne fonctionnent pas avec Open VSwitch.

a. Télécharger les paquets nécessaires au fonctionnement d'Open VSwitch

```
sudo apt-get install openvswitch-switch
sudo apt-get install openvswitch-common
```

b. Configurer le bridge br0 ne contenant aucune interface système et le bridge br1 contenant l'interface de l'hôte permettant la connexion à internet

Création des bridges :

```
ovs-vsctl add-br switch-br0
ovs-vsctl add-br switch-br1
```

Rajout de l'interfaces demandée dans le bridge :

```
ovs-vsctl add-port switch-br1 enp0s8
```

On peut vérifier les configurations avec la commande « **`ovs-vsctl show`** » qui retourne dans notre cas :

```
jade@ubuntu:~$ sudo ovs-vsctl show
93e1f7e7-abd0-465e-8cfe-033bd1a53e80
    Bridge switch-br0
        Port switch-br0
            Interface switch-br0
                type: internal
    Bridge switch-br1
        Port enp0s8
            Interface enp0s8
        Port switch-br1
            Interface switch-br1
                type: internal
    ovs_version: "2.17.3"
```

c. *Configuration pour QeMU*

Ces commandes permettent de rajouter les interfaces tap dans le bridge br0

```
ovs-vsctl add-port switch-br0 tap0
ovs-vsctl add-port switch-br0 tap1
```

Ne pas oublier de mettre à UP tous les bridges ainsi que `ovs-system` avec la commande « `sudo ip link set dev ovs-system up` »

d. *Configuration pour LXC*

La configuration reste identique que pour les parties 2 et 3. En effet, elle se fait par le fichier de configuration « `sudo nano /var/lib/lxc/`*`nom_conteneur`*`/config`».

Cela rajoute automatiquement les conteneurs dans le bridge.

Je n'ai pas réussi à faire fonctionner le DHCP en suivant le cours* slide 54 et 55. J'ai donc rajouté une adresse IP dans le fichier de configuration « config » en statique.

Les conteneurs arrivaient à communiquer entre eux mais pas avec le reste.

**Cours disponible, attention les paramètres des fichiers de configurations sont dépréciés.*

Cours réseau privé	*Cous full bridge*

`sudo nano /var/lib/lxc/deb1/config`
 	
`lxc.network.veth.pair = br-deb1`
`lxc.network.ipv4 = 192.168.1.102/24`
 	
`^X`
 	
`Nano script0.sh`
 	
`ovs-vsctl add-br br0 ip addr add 192.168.1.1/24 dev br0 ip link set dev br0 up ip link set dev ovs-system up lxc-start -n deb0 -d ovs-vsctl add-port br0 br-deb0 lxc-start -n deb1 -d ovs-vsctl add-port br0 br-deb1`
	`sudo nano /var/lib/lxc/deb1/config`
	
	`lxc.network.veth.pair = br-deb1`
	
	`^X`
	
	`Nano script.sh`
	`piddhcp=$(pgrep -f dhcp)`
	`kill -9 $piddhcp`
	`ip link set dev eth0 down`
	`ip addr flush eth0`
	`ip link set dev eth0 up`
	`ovs-vsctl add-br br0`
	`ip link set dev br0 up`
	`ip link set dev ovs-system up`
	`ovs-vsctl add-port br0 eth0`
	`dhclient br0`
	`lxc-start -n deb0 -d`
	`ovs-vsctl add-port br0 br-deb0`

Pourquoi pas directement avec link ?

Ne retourne rien, et pas d'adresse allouées dynamiquement aux conteneurs

5. Toolstack

a. Installation de bases

Pour installer les paquets nécessaires :

```
sudo apt install libvirt-daemon-system virtinst
```

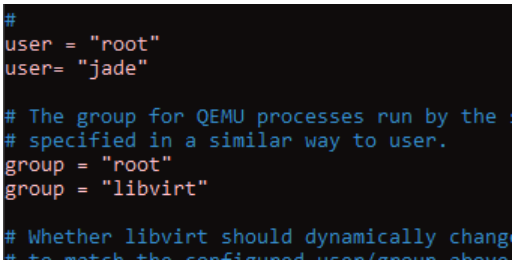
Cette commande permet d'installer une machine Alpine :

```
virt-install --name=v_alpine1 --ram=256 --vcpus=1 --import  --disk path=alpine-extended-3.10.0-x86_64.iso,device=cdrom --os-variant=alpinelinux3.10 --graphics vnc,listen=0.0.0.0 --noautoconsole
```

Afin de trouver le bon nom pour la version de l'os taper cette commande :

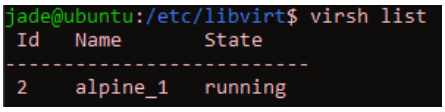
```
virt-install --osinfo list | grep alpine
```

Si une erreur concernant les droits apparait il faut décommenter et rajouter les options suivante (cf. image) dans le fichier «/etc/libvirt/qemu.conf » :



Pour vérifier l'installation de la machine :

```
virsh list
```



Le fichier de configuration de ma VM se trouve avec la commande :

```
sudo virsh  edit alpine_1
```

Pour allumer une machine :

```
sudo virsh  start alpine_1
```

Problème de connexion à console :

```
sudo virsh console alpine_1
```

La commande ne donne pas la main. On peut néanmoins en sortir avec la commande Ctrl+5.

Ces modifications ne fonctionnent pas.

```
sudo virsh  edit alpine_1
sudo nano /etc/default/grub
GRUB_CMDLINE_LINUX="serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1"
```

```
sudo update-grub
```

b. *Virsh et le réseau*

Pour donner une interface à une machine :

```
sudo virsh attach-interface --domain alpine_1 --type bridge \
--source virbr0 --model virtio --config –live
```

A FINIR