Table des matières

1) Compétences à acquérir :

1.1 Capacités

— Installer LXC

— Instancier un container LXC.

— Manipuler un container LXC (stop/start/clone...).

— Connecter un container sur un autre Réseau.

— Bâtir des "network NameSpace".

— Créer et manipuler un container LXD.

— Piloter à distance un Container LXD.

— Orchestrer une machine virtuelle avec LXD.

— Créer un Container avec systemd.

— Créer un container système Docker avec footloose.

1.2 Savoirs à acquérir :

— Comprendre les namespaces en utilisant un network namespace.

— Compendre les CGROUPS pour contraindre des processus : par exemple pour limiter la portéed'un déni de service

1.3 Organisation, recommandation et notation du TP.

Vous travaillerez individuellement sur une machine virtuelleVM Ubuntu 20.04 LTSportée par VM-wareWorkstation de préférence.Il vous explicitement demandé de faire valider votre travail au cours du TP par l'enseignant pourêtre noté. Faites impérativement un compte rendu au fur et à mesure avec des copies d'écran et lesconfigurations mises en oeuvre.Tous les travaux sont à déposer sur l'ENT indiqué par l'enseignant. Un travail doit être enregistreravec les noms des personnes dans le nom du fichier, et l'intitulé du fichier doit être clair (par ex :TP_intitulé_du_tp_Etudiant1_Etudiantn).Les délais sont parfois et exceptionnellement négociables mais une fois fixés doivent être respectéssous peine d'une note nulle.

2 Container LXC sous Linux.

2.1 Installation de LXC.

a) Je commence par installer LXC :

```
root@ubuntu:~# apt-get install lxc bridge-utils lxc-templates
debootstrap debian-archive-keyring
```

b) La commande suivante permet de vérifier la bonne installation de LXC :

```
root@ubuntu:~# lxc-checkconfig
Kernel configuration not found at /proc/config.gz; searching...
Kernel configuration found at /boot/config-5.4.0-26-generic
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled

--- Control groups ---
Cgroups: enabled
```

```
Cgroup v1 mount points:
/sys/fs/cgroup/systemd
/sys/fs/cgroup/hugetlb
/sys/fs/cgroup/cpuset
/sys/fs/cgroup/perf_event
/sys/fs/cgroup/cpu,cpuacct
/sys/fs/cgroup/freezer
/sys/fs/cgroup/blkio
/sys/fs/cgroup/memory
/sys/fs/cgroup/devices
/sys/fs/cgroup/pids
/sys/fs/cgroup/net_cls,net_prio
/sys/fs/cgroup/rdma

Cgroup v2 mount points:
/sys/fs/cgroup/unified

Cgroup v1 clone_children flag: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled, not loaded
Macvlan: enabled, not loaded
Vlan: enabled, not loaded
Bridges: enabled, loaded
Advanced netfilter: enabled, not loaded
CONFIG_NF_NAT_IPV4: missing
CONFIG_NF_NAT_IPV6: missing
CONFIG_IP_NF_TARGET_MASQUERADE: enabled, not loaded
CONFIG_IP6_NF_TARGET_MASQUERADE: enabled, not loaded
CONFIG_NETFILTER_XT_TARGET_CHECKSUM: enabled, loaded
CONFIG_NETFILTER_XT_MATCH_COMMENT: enabled, not loaded
FUSE (for use with lxcfs): enabled, not loaded

--- Checkpoint/Restore ---
checkpoint restore: enabled
CONFIG_FHANDLE: enabled
CONFIG_EVENTFD: enabled
CONFIG_EPOLL: enabled
CONFIG_UNIX_DIAG: enabled
CONFIG_INET_DIAG: enabled
CONFIG_PACKET_DIAG: enabled
CONFIG_NETLINK_DIAG: enabled
```

```
File capabilities:

Note : Before booting a new kernel, you can check its
configuration
usage : CONFIG=/path/to/config /usr/bin/lxc-checkconfig
```

3) Voici la liste des distributions que je peux containeriser sur ma VM :

```
root@ubuntu:/usr/share/lxc# cd templates/
root@ubuntu:/usr/share/lxc/templates# ls
lxc-alpine     lxc-centos    lxc-fedora         lxc-oci
lxc-plamo      lxc-sparclinux    lxc-voidlinux
lxc-altlinux   lxc-cirros    lxc-fedora-legacy  lxc-openmandriva
lxc-pld        lxc-sshd
lxc-archlinux  lxc-debian    lxc-gentoo         lxc-opensuse
lxc-sabayon    lxc-ubuntu
lxc-busybox    lxc-download  lxc-local          lxc-oracle
lxc-slackware  lxc-ubuntu-cloud
```

2.2 Création d’un container LXC Debian buster.

1) Je crée un container debian buster nommé debian-j1 :

```
lxc-create -t debian -n debian-j1 -- -a amd64
```

2) je recrée un container du même type nommé debian-j2 :

```
root@ubuntu:/usr/share/lxc/templates# lxc-create -t debian -n
debian-j2
debootstrap is /usr/sbin/debootstrap
Checking cache download in /var/cache/lxc/debian/rootfs-stable-
amd64 ...
Copying rootfs to /var/lib/lxc/debian-j2/rootfs...Generating
locales (this might take a while)...
  fr_FR.UTF-8... done
  fr_FR.UTF-8... done
Generation complete.
update-rc.d: error: cannot find a LSB script for checkroot.sh
update-rc.d: error: cannot find a LSB script for umountfs
Failed to disable unit, unit hwclock.sh.service does not exist.
update-rc.d: error: cannot find a LSB script for hwclockfirst.sh
Creating SSH2 RSA key; this may take some time ...
```

```
2048 SHA256:0qydpSP/KRkJ9kLnFB3YPk8mzHyIffOTVja9FAtmDxI
root@ubuntu (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:/BuPegBCZKHk8Wtrv7M3qBkrZ715Bd+yrH+EAfK+/Ek root@ubuntu
(ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:xLKKJm8+QSuK0aOLG7k9Th44Y74mGdPYetkOu8z9sjw root@ubuntu
(ED25519)
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.

Current default time zone: 'Etc/UTC'
Local time is now:      Wed Nov 11 19:33:52 UTC 2020.
Universal Time is now:  Wed Nov 11 19:33:52 UTC 2020.
```

Je vois que sa création s'est fait plus rapidement que la première. La création s'est servit des fichiers du premier container et les a copié pour faire le deuxième container.

3) Je crée un container fedora nommé fedora :

```
lxc-create -t fedora -n fedora
```

4) Je crée un container centos avec le template :

```
lxc-create --name centos --template=download -- --dist=centos –
release=8 --arch=amd64
```

5) **Sur le container debian-j1 :**

a) Je démarre mon container debian-j1 :

```
root@ubuntu:~# lxc-start debian-j1
```

b) J'arrête mon container :

```
root@ubuntu:~# lxc-stop debian-j1
```

c) Je redémarre mon container en mode démon :

```
root@ubuntu:~# lxc-start debian-j1 -d
```

d) Je m'attache au container et je lance la commande ip a :

```
root@ubuntu:~# lxc-attach debian-j1
root@debian-j1:~# 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: eth0@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
    link/ether 00:16:3e:28:88:41 brd ff:ff:ff:ff:ff:ff link-
netnsid 0
    inet 10.0.3.208/24 brd 10.0.3.255 scope global dynamic eth0
       valid_lft 3429sec preferred_lft 3429sec
    inet6 fe80::216:3eff:fe28:8841/64 scope link
       valid_lft forever preferred_lft forever
```

e) Je fais un ps -AH pour voir le PID ainsi que les process enfants de lxc-start :

```
  25151 ?        00:00:00   lxc-start
  25152 ?        00:00:00     systemd
  25202 ?        00:00:00       systemd-journal
  25227 ?        00:00:00       dhclient
  25249 pts/1    00:00:00       agetty
  25250 ?        00:00:00       sshd
```

f) Je modifie la RAM de mon container :

```
root@ubuntu:~# lxc-cgroup -n debian-j1 memory.limit_in_bytes 512M
root@ubuntu:~# lxc-attach debian-j1
root@debian-j1:~# free
              total        used        free      shared
buff/cache   available
Mem:         524288       12884      497808        8184
13596      511404
Swap:             0           0           0
```

g) J’installe apache2 dans mon container :

```
root@ubuntu:~# lxc-attach debian-j1
root@debian-j1:~# apt install apache2
```

h) Pour faire en sorte que le container démarre automatiquement au démarrage de l’hôte on fait un autostart :

```
root@ubuntu:~# lxc-autostart debian-j1
```

i) Je fige et je relance mon container :

```
root@ubuntu:~# lxc-freeze debian-j1
root@ubuntu:~# lxc-unfreeze debian-j1
```

j) Je commence par éteindre mon container et ensuite je le copie :

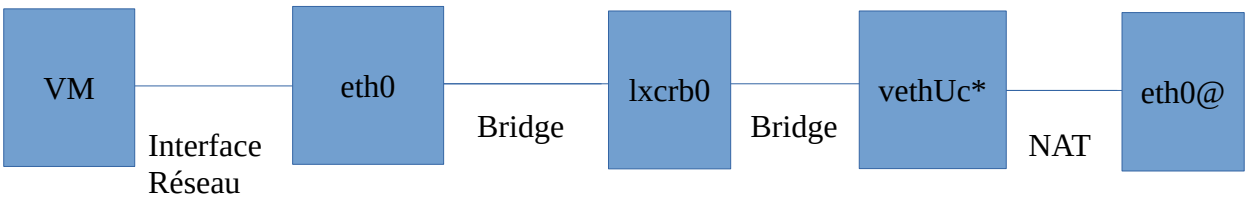
```
root@ubuntu:~# lxc-copy debian-j1 -N clonedebian-j1
root@ubuntu:~# lxc-ls
centos          centos7         clonedebian-j1 debian-j1
debian-j2       fedora
```

6) En faisant un ip a je vois que mon container se sert de la carte réseau de ma VM pour se connecter au réseau :

```
2: eth0@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
    link/ether 00:16:3e:28:88:41 brd ff:ff:ff:ff:ff:ff link-
netnsid 0
    inet 10.0.3.208/24 brd 10.0.3.255 scope global dynamic eth0
       valid_lft 3522sec preferred_lft 3522sec
    inet6 fe80::216:3eff:fe28:8841/64 scope link
       valid_lft forever preferred_lft forever
```

Avec un brctl show je vois que mon container crée une interface bridge pour se connecter à ma VM

```
root@ubuntu:~# brctl show
bridge name     bridge id       STP enabled     interfaces
docker0         8000.02423080e7ad       no
lxcbr0          8000.00163e000000       no              vethgE2iQj
virbr0          8000.525400b18a0b       yes             virbr0-nic
```



7) Je crée une nouvelle interface nommée eth1 (pour ça je dois éteindre ma VM)

Ensuite je modifie le fichier /etc/netplan/01-netcfg.yaml :

```
# This file describes the network interfaces available on your
system
# For more information, see netplan(5).
network:
  version: 2
```

```
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: yes
    eth1:
      dhcp4: yes
  bridges:
    monbr0:
      interfaces: [eth1]
      dhcp4: yes
```

Maintenant je redémarre la couche réseau et je vérifie que mon bridge a bien été crée :

```
root@ubuntu:~# netplan apply
```

Je peux voir en faisant un ip a que mon bridge est crée :

```
8: monbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
    link/ether 08:00:27:68:58:8b brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.62/24 brd 192.168.1.255 scope global dynamic
monbr0
       valid_lft 86393sec preferred_lft 86393sec
    inet6 2a01:cb1d:8af6:4b00:a00:27ff:fe68:588b/64 scope global
dynamic mngtmpaddr noprefixroute
       valid_lft 1798sec preferred_lft 598sec
    inet6 fe80::b860:57ff:fe58:5a32/64 scope link
       valid_lft forever preferred_lft forever
```

Je modifie la configuration de mon container sous /var/lib/lxc/debian-j2/config :

```
lxc.net.0.link = monbr0
```

Le container fait un bridge de monbr0 qui est lui-même un bridge. On a donc un bridge sur un bridge

8) Les containers sont stockés dans /var/lib/lxc/

```
root@ubuntu:/var/lib/lxc/debian-j1/rootfs# chroot
/var/lib/lxc/debian-j1/rootfs/
root@ubuntu:/# ls
bin  boot  dev    etc  home  lib    lib32  lib64  libx32  media
mnt  opt  proc  root  run    sbin  selinux  srv  sys  tmp  usr
var
```

Je vois les fichiers qui sont dans mon container donc cette commande permet de faire comme si je m'étais attaché au container.

Je change le mot de passe :

```
root@ubuntu:/# passwd
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd: password updated successfully
```

Le mot de passe a bien été changé

2.3 Containers privilégiés

1)

2) Il faut regarder le subuid et le subguid de l'user student (non privilégié).

```
student@ubuntu:/root$ cat /etc/sub{g,u}id |grep $USER
student:165536:65536
student:165536:65536
```

3) Je crée l'arborescence ~/.config/lxc

```
cd ~
mkdir -p .config/lxc
```

Je copie le fichier de conf.

```
cp /etc/lxc/default.conf ~/.config/lxc/
```

Je rajoute ensuite les deux premières lignes, avec les bons ID.

```
student@ubuntu:~/.config/lxc$ cat default.conf
lxc.idmap = u 0 165536 65536
lxc.idmap = g 0 165536 65536
lxc.net.0.type = veth
lxc.net.0.link = lxcbr0
lxc.net.0.flags = up
lxc.net.0.hwaddr = 00:16:3e:xx:xx:xx
```

Je donne les droits pour pouvoir créer les bridges.

```
echo "$USER veth lxcbr0 2" | sudo tee -a /etc/lxc/lxc-usernet
```

4) Je tente la création d'un container ubuntu "focal"

```
student@ubuntu:~/.config/lxc$ lxc-create -t download -n ubuntu1 --
-r focal -a amd64
```

Quand on nous demande la distribution on tape ubuntu

5) Je démarre le container et je rentre dedans :

```
root@ubuntu:~# lxc-start ubuntu1
root@ubuntu:~# lxc-attach ubuntu1
```

J'installe apache2

Je regarde les processus :

```
root@ubuntu1:/# ps -AH
    PID TTY          TIME CMD
    200 ?        00:00:00 bash
    204 ?        00:00:00   ps
  ...
    108 ?        00:00:00   apache2
    109 ?        00:00:00     apache2
    111 ?        00:00:00     apache2
```

Je compare avec la VM :

```
 13796 ?        00:00:00       apache2
  13797 ?        00:00:00         apache2
  13799 ?        00:00:00         apache2
```

Je vois les mêmes processus (as student).

Donc, lancer les containers en non-privilégié permet quand même de voir les processus en user non-privilégié.

Pas besoin d'avoir les droits pour voir les process.

6) On va faire du DNAT :

```
root@ubuntu:~# iptables -t nat -A PREROUTING -d 192.168.1.61 -p
tcp --dport 80 -j DNAT --to-destination 10.0.3.1
```

Je redirige le port 80 de eth0 vers mon container.

Je tape l'ip directement dans le navigateur de mon PC.

La page apache2 ubuntu s'affiche.

3 Focus sur les briques de bases des containers

3.1 Création de cgroups à l'aide de cgroup-bin.

Sur ma machine physique je fais :

```
samuel@samuel-MS-7B51:~$ xhost root@192.168.1.61
root@192.168.1.61 being added to access control list
samuel@samuel-MS-7B51:~$ ssh -X root@192.168.1.61
```

Maintenant sur ma VM je modifie la configuration ssh :

```
X11Forwarding yes
X11UseLocalhost no
```

Je passe les commandes suivantes :

```
systemctl restart ssh
apt-get install cgroup-tools xterm
apt install x11-xserver-utils
# lance un xterm de couleur orange très consommateur de CPU
xterm -bg orange -e "md5sum /dev/urandom" &
# lance un xterm de couleur bleu très consommateur de CPU
xterm -bg blue -e "md5sum /dev/urandom" &
```

1) Je fais un top pour voir la consommation de mon CPU par les deux xterm :

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM
TIME+ COMMAND
  19367 root      20   0    5492    588    524 R  99,7   0,0
4:01.41 md5sum
  19442 root      20   0    5492    524    456 R  99,3   0,0
3:46.68 md5sum
```

Je vois que chacun son tour le xterm consomme tout le CPU

2) J'utilise plusieurs commandes cg pour affecter 80% du CPU au xterm orange et 20% du CPU au xterm bleu :

```
cgcreate -g cpu,cpuset:quatrevingtpourcentcpu
cgcreate -g cpu,cpuset:vingtpourcentcpu
cgset -r cpu.shares=2 vingtpourcentcpu
cgset -r cpu.shares=8 quatrevingtpourcentcpu
cgget -r cpu.shares quatrevingtpourcentcpu
cgget -r cpu.shares vingtpourcentcpu
```

```
cgexec -g cpu:quatrevingtpourcentcpu xterm -bg orange -e "md5sum /
dev/urandom" &
cgexec -g cpu:vingtpourcentcpu xterm -bg blue -e "md5sum
/dev/urandom" &
top -d2
```

3) La répartition du CPU est bien effectuée sur les deux xterm

4)

3.2 Manipulation du network namespace.

1) Je me crée un network namespace nommé netns1 et un autre netns2 :

```
root@ubuntu:~# ip netns add netns1
root@ubuntu:~# ip netns add netns2
root@ubuntu:~# ip netns list
netns2
netns1
```

2) a) Je me rattache au ns netns1 :

```
root@ubuntu:~# ip netns exec netns1 \
>
```

Je n'ai plus qu'à passer une commande

b) Le device crée par défaut est le loopback :

```
root@ubuntu:~# ip netns exec netns1 \
> ip a s
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

c) Je fais une ethtool dans netns1

```
root@ubuntu:~# ip netns exec netns1 ethtool -k lo
Features for lo:
rx-checksumming: on [fixed]
tx-checksumming: on
     tx-checksum-ipv4: off [fixed]
     tx-checksum-ip-generic: on [fixed]
     tx-checksum-ipv6: off [fixed]
     tx-checksum-fcoe-crc: off [fixed]
     tx-checksum-sctp: on [fixed]
```

```
scatter-gather: on
	tx-scatter-gather: on [fixed]
	tx-scatter-gather-fraglist: on [fixed]
tcp-segmentation-offload: on
	tx-tcp-segmentation: on
	tx-tcp-ecn-segmentation: on
	tx-tcp-mangleid-segmentation: on
	tx-tcp6-segmentation: on
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off [fixed]
rx-vlan-offload: off [fixed]
tx-vlan-offload: off [fixed]
ntuple-filters: off [fixed]
receive-hashing: off [fixed]
highdma: on [fixed]
rx-vlan-filter: off [fixed]
vlan-challenged: on [fixed]
tx-lockless: on [fixed]
netns-local: on [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: off [fixed]
tx-gre-csum-segmentation: off [fixed]
tx-ipxip4-segmentation: off [fixed]
tx-ipxip6-segmentation: off [fixed]
tx-udp_tnl-segmentation: off [fixed]
tx-udp_tnl-csum-segmentation: off [fixed]
tx-gso-partial: off [fixed]
tx-sctp-segmentation: on
tx-esp-segmentation: off [fixed]
tx-udp-segmentation: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off [fixed]
loopback: on [fixed]
rx-fcs: off [fixed]
rx-all: off [fixed]
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off [fixed]
hw-tc-offload: off [fixed]
esp-hw-offload: off [fixed]
esp-tx-csum-hw-offload: off [fixed]
rx-udp_tunnel-port-offload: off [fixed]
tls-hw-tx-offload: off [fixed]
tls-hw-rx-offload: off [fixed]
```

```
rx-gro-hw: off [fixed]
tls-hw-record: off [fixed]
```

Il nous donne des informations spécifiques à la carte réseau

d) Non ce n'est pas migrable d'un netns à un autre

3) a) J'ajoute un devise veth :

```
root@ubuntu:~# ip link add name vethnetns type veth peer name
vethnetns-peer
```

b) J'affecte vethnetns-peer à netns2 :

```
root@ubuntu:~# ip link set vethnetns-peer netns netns2
```

c) Je donne des IP dans le même LAN et je ping l'ip de l'autre netns :

```
root@ubuntu:~# ip netns exec netns2 ip a a 192.168.1.1/24 dev
vethnetns-peer
root@ubuntu:~# ip netns exec netns2 ip link set up dev vethnetns-
peer
root@ubuntu:~# ip a a 192.168.1.2/24 dev vethnetns
root@ubuntu:~# ip link set up dev vethnetns
root@ubuntu:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=3.69 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=2.94 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=4.44 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 2.944/3.690/4.442/0.611 ms
```

4 Création de containers LXD

4.1 Installation de LXD sous Ubuntu 20 si nécessaire

1) Je commence par installer lxd :

```
apt install snap zfsutils-linux
. /etc/profile.d/apps-bin-path.shsn
apt install lxd
export PATH=/snap/bin:$PATH
lxd init# ( choisissez zfs et permettez l'accès distant
```

Je liste les repositories des templates LXD :

```
root@ubuntu:~# lxc remote list
To start your first instance, try: lxc launch ubuntu:18.04

+-----------------+-----------------------------------------
+---------------+-------------+--------+--------+
|      NAME       |                  URL                    |
PROTOCOL    |  AUTH TYPE  | PUBLIC | STATIC |
+-----------------+-----------------------------------------
+---------------+-------------+--------+--------+
| images          | https://images.linuxcontainers.org      |
simplestreams | none        | YES    | NO     |
+-----------------+-----------------------------------------
+---------------+-------------+--------+--------+
| local (default) | unix://                                 | lxd
| file access | NO     | YES    |
+-----------------+-----------------------------------------
+---------------+-------------+--------+--------+
| ubuntu          | https://cloud-images.ubuntu.com/releases |
simplestreams | none        | YES    | YES    |
+-----------------+-----------------------------------------
+---------------+-------------+--------+--------+
| ubuntu-daily    | https://cloud-images.ubuntu.com/daily    |
simplestreams | none        | YES    | YES    |
+-----------------+-----------------------------------------
+---------------+-------------+--------+--------+
```

Toutes les distributions Linux supportés

4.2 Création de containers sous LXD

1) J'installe plusieurs container LXD :

```
lxc launch images:debian/buster debian
lxc launch ubuntu:20.04 ubuntu
lxc launch images:centos/8 centos
```

2) Je liste les containers LXD :

```
root@ubuntu:~# lxc ls
+--------+---------+----------------------
+-----------------------------------------------+-----------
+-----------+
|  NAME  |  STATE  |         IPV4          |
IPV6                       |   TYPE    | SNAPSHOTS |
+--------+---------+----------------------
+-----------------------------------------------+-----------
+-----------+
```

```
| debian | RUNNING | 10.30.225.165 (eth0) |
fd42:4892:99e0:bec4:216:3eff:fe34:e4db (eth0) | CONTAINER | 0
|
+--------+---------+----------------------
+-----------------------------------------------+-----------
+-----------+
| ubuntu | RUNNING | 10.30.225.245 (eth0) |
fd42:4892:99e0:bec4:216:3eff:fed3:ecdf (eth0) | CONTAINER | 0
|
+--------+---------+----------------------
+-----------------------------------------------+-----------
+-----------+
```

Je ne vois pas le container centos car il a bug pendant l'installation

3) Je lance un process bash dans mon container debian :

```
root@ubuntu:~# lxc exec debian /bin/bash
```

4) je visualise le bridge utilisé :

```
root@ubuntu:~# lxc network list
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
|  NAME   |   TYPE   | MANAGED |      IPV4      |           IPV6
| DESCRIPTION | USED BY |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| docker0 | bridge   | NO      |                |
|             | 0       |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| eth0    | physical | NO      |                |
|             | 0       |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| eth1    | physical | NO      |                |
|             | 0       |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| lxcbr0  | bridge   | NO      |                |
|             | 0       |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| lxdbr0  | bridge   | YES     | 10.30.225.1/24 |
fd42:4892:99e0:bec4::1/64 |             | 3       |
```

```
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| monbr0  | bridge   | NO      |                |
|             | 0       |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
| virbr0  | bridge   | NO      |                |
|             | 0       |
+---------+----------+---------+----------------
+---------------------------+-------------+---------+
```

Je peux visualiser le fichier de configuration de ma carte réseau :

```
root@ubuntu:~# lxc network edit lxdbr0
```

5)