System Containers Unleashed

2 Container LXC sous Linux.

2.1 Installation de LXC.

Commande d'installation des packets : `apt-get install lxc bridge-utils lxc-templates debootstrap dnf debian-archive-keyring lxc-utils`

Commande pour verifier la bonne installation :

`lxc --version` -> 5.13.

`lxc-checkconfig` -> Donne les caractériqtique compatible avec la machine hôte.

Template VM possible :

```
julien@julien:/usr/share/lxc$ ls ./templates/
|
|lxc-alpine     lxc-busybox  lxc-debian    lxc-fedora         lxc-kali
lxc-openmandriva  lxc-plamo    lxc-slackware   lxc-ubuntu         |
|lxc-altlinux   lxc-centos   lxc-devuan    lxc-fedora-legacy  lxc-local
lxc-opensuse      lxc-pld      lxc-sparclinux  lxc-ubuntu-cloud   |
|lxc-archlinux  lxc-cirros   lxc-download  lxc-gentoo         lxc-oci
lxc-oracle        lxc-sabayon  lxc-sshd        lxc-voidlinux
```

2.2 Création d'un container LXC Debian buster.

Commande pour lancé la création d'un container : `sudo lxc-create --template debian --name debian-j1`

La création du deuxième container debian-j2 est plus rapide car il possède déja l'image OS.

Création d'un container fedora : `sudo lxc-create --template fedora --name fedora`

Création d'un container centos 7 avec l'architecture amd64 : `sudo lxc-create -t download -no-validate -n centos-j1 -- --dist centos --release 7 --arch amd64`

-template=download -no-validate = Télécharge la pré-distribution de l'OS selectionné et non pré validé pour éviter les problème de compatibilité.

5:

Démarrer le container : `sudo lxc-start -n debian-j1`

Arrêter le container : `sudo lxc-stop -n debian-j1`

Se rattacher au container :

```
sudo lxc-attach -n 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@if5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 00:16:3e:85:80:2e brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.3.70/24 brd 10.0.3.255 scope global dynamic eth0
       valid_lft 3591sec preferred_lft 3591sec
    inet6 fe80::216:3eff:fe85:802e/64 scope link
       valid_lft forever preferred_lft forever
```

Afficher le PID du container : `sudo lxc-info -n debian-j1`

```
julien@julien:/usr/share/lxc$ sudo lxc-info -n debian-j1
|Name:           debian-j1
|State:          RUNNING
|PID:            15432
|IP:             10.0.3.70
|Link:           vethgRjPng
| TX bytes:      1.56 KiB
| RX bytes:      2.02 KiB
| Total bytes:   3.58 KiB
```

```
julien@julien:/usr/share/lxc$ pstree 15432
systemd─┬─agetty
        ├─dhclient───3*[{dhclient}]
        ├─sshd
        └─systemd-journal
```

Limiter la memoire swap max du conteneur à 1 GB ou 512 MIB : `sudo lxc-cgroup -n debian-j1 memory.swap.max 1 GB`

Installer apache2 et le rendre accessible sur la VM hôte :

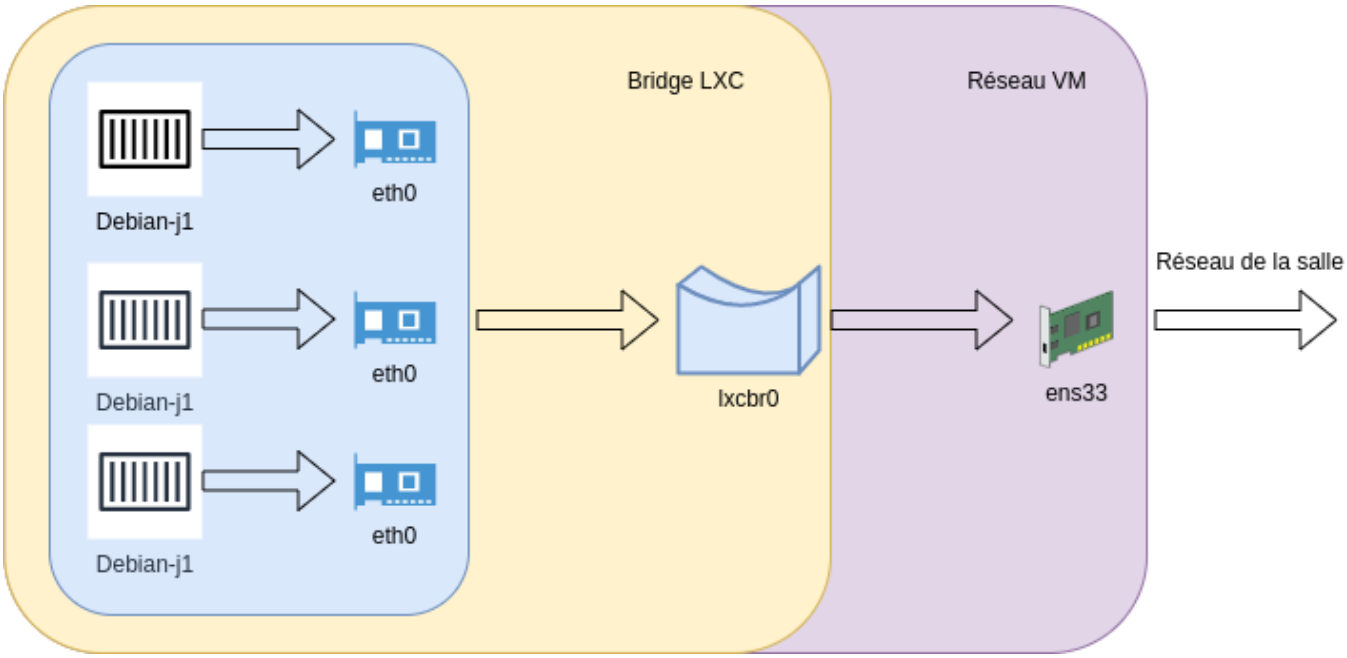
```
|   11  apt install apache2|
|   12  systemctl start apache2|
|   13  systemctl enable apache2|
|   16  sed -i -e
's,PrivateTmp=true,PrivateTmp=false\nNoNewPrivileges=yes,g'
/lib/systemd/system/apache2.service                          |
|   17  systemctl daemon-reload|
|   21  apt install iptables
|   22  iptables -A INPUT -i eth0@if10 -p tcp --dport 80 -j ACCEPT|
|   23  ip a|
|   30  systemctl restart apache2
|   31  systemctl status apache2
```

Commande de auto-restart : `sudo lxc-autorestart -n debian-j1`

Commande pour freeze/unfreeze le conteneur : `sudo lxc-freeze -n debian-j1` ou unfreeze pour relancer.

Commande pour cloné le conteneur : `sudo lxc-copy -s -n debian-j1 -N pouet2` Clone à partir du snapshot du conteneur visé et il doit être éteint.

6: Le conteneur est connecté à la carte réseau bridge : `lxcbr0` depuis sa carte réseau nommé `eth0`.



Création du bridge :

```
sudo brctl addbr eth1
sudo brctl addif eth1 ens33
sudo ip link set up eth1
sudo dhclient -v eth1
```

Les containers sont stockés dans le répertoir : /var/lib/lxc/

Modification du mots de passe depuis les fichiers locaux des container :

```
chroot /var/lib/lxc/debian-j1/rootfs
passwd root
```

2.3 Containers non privilégiés

1. Création d'un container non privilégié avec le user student :

2. Modifiez /etc/subuid et /etc/subgid et expliquez ce mapping :

```
root@ubuntu:~# cat /etc/subuid
root:100000:65536
student:165536:65536
root@ubuntu:~# cat /etc/subgid
root:100000:65536
student:165536:65536
```

Ce changement permet de définir la plage pour isoler les uid ou gid de l'utilisateur student ici entre 165536 à 65536.

3. Création du fichier :

```
 /home/student/.config/lxc/default.conf
lxc.idmap = u 0 165536 65536
lxc.idmap = g 0 165536 65536
lxc.mount.auto = proc:mixed sys:ro cgroup:mixed
lxc.net.0.type = veth
lxc.net.0.link = lxcbr0
lxc.net.0.flags = up
```

4. Permet à l'utilisateur student de créer "du réseau" : `echo "$USER veth lxcbr0 2" | sudo tee -a /etc/lxc/lxc-usernet`

5. Création et démarrage d'un container ubuntu "jammy" : `lxc-create -t download -n ubuntu1 -- -r jammy -a amd64` Malheureusement après de nombreux essaie je n'ai pas réussi à faire fonctionner le conteneur j'ai réussi à le créer et le vois dans ma table comme crée par un utilisateur restreint.

3 Focus sur les briques de bases des containers

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

cgclear supprime les groupes lscgroup liste les groupes cat /proc/mount/ montre ce qui est monté (FS cgroup) cat/proc/cgroups permet de voir les cgroups ou issubsys

Pour modfier la version de CGROUP et utilisé la v1 :

```
cd /etc/default/
vim grub
GRUB_CMDLINE_LINUX="systemd.unified_cgroup_hierarchy=0"
```

— Sur votre machine physique passez les commandes suivantes :

```
xhost ip_de_votre_vm # xhost + ouvre à toutes les IP
ssh -X ip_de_votre_vm
```

Sur la VM : — Ne conservez qu'un seul CPU virtuel sur la VM si ce n'est pas le cas. — Si besoin modifiez la configuration SSH dans /etc/ssh/sshd_config :

```
...
X11Forwarding yes
X11UseLocalhost no
...
```

```
#Redémarrage de SSH pour la prise en compte des modifiaction.
systemctl ssh restart
#Installation des outils pour la gestion des cgroup et xterm
apt-get install cgroup-tools xterm
apt install x11-xserver-utils
```

Lancement des CPU test :

```
#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. La répartition du CPU entre les deux commandes est égale 50/50 de l'utilisation du CPU.

2. Utilisez cgcreate , cgset, cgexec afin d'affecter 80% du CPU :

Lancez les commandes suivantes :

```
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
```

Fonctionne correctement, même si j'ai perdu 1h à debugger.

Contenue des fichiers cpu.weight dans les cgroups respectif :

```
sys/fs/cgroup/vingtpourcentcpu$ cat cpu.weight
20
/sys/fs/cgroup/quatrevingtpourcentcpu$ cat cpu.weight
80
```

3.2 Manipulation du network namespace.

Création des carte réseau :

```
#Création des network :
julien@julien:~$ sudo ip netns add netns1
julien@julien:~$ sudo ip netns add netns2

#Lister les network crée :
julien@julien:~$ sudo ip netns list

netns2
netns1

#Strace pour appeler la création d'un network namespace :
julien@julien:~$ sudo strace -f -e trace=network ip netns add netns1

socket(AF_NETLINK, SOCK_RAW|SOCK_CLOEXEC, NETLINK_ROUTE) = 3*
setsockopt(3, SOL_SOCKET, SO_SNDBUF, [32768], 4) = 0
setsockopt(3, SOL_SOCKET, SO_RCVBUF, [1048576], 4) = 0
setsockopt(3, SOL_NETLINK, NETLINK_EXT_ACK, [1], 4) = 0
bind(3, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 0
getsockname(3, {sa_family=AF_NETLINK, nl_pid=23355, nl_groups=00000000},
[12]) = 0
setsockopt(3, SOL_NETLINK, NETLINK_GET_STRICT_CHK, [1], 4) = 0
socket(AF_NETLINK, SOCK_RAW|SOCK_CLOEXEC, NETLINK_ROUTE) = 4
setsockopt(4, SOL_SOCKET, SO_SNDBUF, [32768], 4) = 0
setsockopt(4, SOL_SOCKET, SO_RCVBUF, [1048576], 4) = 0
setsockopt(4, SOL_NETLINK, NETLINK_EXT_ACK, [1], 4) = 0
bind(4, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 0
getsockname(4, {sa_family=AF_NETLINK, nl_pid=-1293278300,
nl_groups=00000000}, [12]) = 0
Cannot create namespace file "/run/netns/netns1": File exists
+++ exited with 1 +++
```

Rattachement de netns1 :

```
sudo ip netns exec netns1 /bin/bash
```

La seule interface apparente est un loopback.

La commande `ethtool -k lo` donne toute les informations lié à la carte réseau.

La device étant une loopback elle n'est pas migrable, mais une carte réseau classique il est possible de la migrer entre netns.

Création de la carte réseau virtuelle :

```
#Création de la cartes virtuelles réseaux, ajoute de la device veth
ip link add name vethnetns type veth peer name vethnetns-peer

#Affectation de ethnetns-peer sur netns2
sudo ip link set vethnetns-peer netns netns2
```

4 Création de containers LXD

4.1 Installation de LXD sous Ubuntu si nécessaire

1. Installation des éléments nécessaires au TP :

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

2. lxc remote list -> Donne la liste des serveurs LXD distants que le client LXC connais. lxc image list images: -> Affiche une liste des distribution prédement télécharger et stockée sur le serveur local LXD. J'en déduis donc que les distributions supporté sont dépendente de se qu'on à téléchargé comme image LXD(Les type de distribution ubuntu, debian, centos, etc... Sont compatible du moment que leur versions est mise à disposition sur les repository officiel).

4.2 Création de containers sous LXD

1. Installation des containers en suivant les instructions suivantes :

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

2. Listage des conteneurs à l'aide de la commande précédente et je vois bien la debian, ubuntu et centos8.

```
root@debian:~# lxc image list image
+-------+-------------+--------+-------------+--------------+------+------+-------------+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-------+-------------+--------+-------------+--------------+------+------+-------------+
root@debian:~# lxc image list
+-------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+--------------
--------------+
| ALIAS | FINGERPRINT  | PUBLIC |               DESCRIPTION               | ARCHITECTURE |      TYPE       |   SIZE   |        UPLOAD
 DATE          |
+-------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+--------------
--------------+
|       | 2fd167b424da | no     | ubuntu 20.04 LTS amd64 (release) (20230506) | x86_64       | VIRTUAL-MACHINE | 601.38MB | May 12, 2023 a
t 8:19am (UTC) |
+-------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+--------------
--------------+
|       | 4189b078eafc | no     | ubuntu 20.04 LTS amd64 (release) (20230506) | x86_64       | CONTAINER       | 436.08MB | May 12, 2023 a
t 8:26am (UTC) |
+-------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+--------------
--------------+
|       | a7bca638c748 | no     | Debian buster amd64 (20230511_15:32)        | x86_64       | CONTAINER       | 75.54MB  | May 12, 2023 a
t 8:25am (UTC) |
+-------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+--------------
```

3. Démarrage du container : `lxc start` et lancement d'un process bash dans le container centos8 : `lxc exec centos8-- /bin/bash`

4.
```
 julien@julien:~$ lxc network list
 |+--------+----------+---------+-----------------+------+------------
-+---------+---------+
 ||  NAME  |   TYPE   | MANAGED |      IPV4       | IPV6 | DESCRIPTION
| USED BY |  STATE  |
 |+--------+----------+---------+-----------------+------+------------
-+---------+---------+
 || ens33  | physical | NO      |                 |      |
| 0       |         |
 |+--------+----------+---------+-----------------+------+------------
-+---------+---------+
 || lxdbr0 | bridge   | YES     | 10.203.210.1/24 | none |
| 1       | CREATED |
 |+--------+----------+---------+-----------------+------+------------
-+---------+---------+

 julien@julien:~$ lxc network edit lxdbr0
 #Ouvre un fichier yaml representant la carte réseau.
```

5.

```
julien@julien:~$ curl -s --unix-socket /var/snap/lxd/common/lxd/unix.socket
-X POST -d '{"name": "xenial",
    "source": {"type": "image", "protocol": "simplestreams", "server":
    "https://cloud-images.ubuntu.com/daily", "alias": "18.04"}}'
a/1.0/containers | jq .
    {
    "type": "async",
    "status": "Operation created",
    "status_code": 100,
    "operation": "/1.0/operations/b5276357-d618-4467-b9a4-aa7541211681",
    "error_code": 0,
    "error": "",
    "metadata": {
        "id": "b5276357-d618-4467-b9a4-aa7541211681",
        "class": "task",
        "description": "Creating instance",
        "created_at": "2023-05-05T13:08:11.350362787Z",
        "updated_at": "2023-05-05T13:08:11.350362787Z",
        "status": "Running",
        "status_code": 103,
        "resources": {
        "containers": [
            "/1.0/containers/xenial"
        ],
        "instances": [
            "/1.0/instances/xenial"
        ]
        },
        "metadata": null,
        "may_cancel": false,
        "err": "",
        "location": "none"
    }
    }
```

La commande permet de crée un conteneur LXC nommé xenial à partir du répository `https://cloud-images.ubuntu.com/daily` une image ubuntu 18.04. La requêtes est au format JSON pour faciliter la lecture. Un socket du domaine Unix est un moyen de communiquer entre les processus. Exemple ici avec le curl qui intérragie avec LXD pour pouvoir crée et gérer les conteneurs.

6. Pilotage de lxd depuis un client réseau. Connection au lxd du voisins : `lxc remote add voisin1 ip_du_voisin`
   Je n'ai pas pu essayer mais j'imagine que le processus est le même que l'exécution de commande en local comme fait précédemment, une fois connecté au LXD distant.

4.3 Utilisation de LXD comme orchestrateur de machines virtuelles

L'API de LXD permet d'"orchestrer" aussi des machines virtuelles. Cloud-init est un outil qui permet l'initialisation et la configuration de machines virtuelles pour le CLOUD. cloud-config définit la configuration de votre future VM (`https://cloudinit.readthedocs.io/en/latest/topics/examples.html`)

1. Installation le package whois et générer un mot de passe pour l'utilisateur de votre VM : `apt install whois` et `mkpasswd --method=SHA-512 --rounds=4096`.

```
root@debian:~# mkpasswd --method=SHA-512 --rounds=4096
Mot de passe : "pouet"
$6$rounds=4096$27EiIy5WP4BJRTIA$stGQbb5.j0fKiOoPEOa2CMiP3Ctqaq.kH7lKoaIF/YN
GGtZoXQszI0aa5jtXzdwCYxL24aB4RBliWtcMqa1xY
```

2. Suppressiosn des VMWareTools qui supplantent le module Kernel vsock standard nécessaire à lxd : `apt remove open-vm-tools` et `reboot`

3. Création d'un profil pour la VM lxd : `lxc profile create vmubuntu` :

```
root@debian:~# lxc profile create vmubuntu
Profile vmubuntu created
```

Création du fichier yaml suivant (le mot de passe est celui généré par mkpasswd) :

```
config:
  user.user-data: |
    #cloud-config
    apt_mirror: http://us.archive.ubuntu.com/ubuntu/
    ssh_pwauth: yes
    users:
      - name: ubuntu
          passwd:
"$6$rounds=4096$27EiIy5WP4BJRTIA$stGQbb5.j0fKiOoPEOa2CMiP3Ctqaq.kH7lKoaIF/Y
NGGtZoXQszI0aa5jtXzdwCYxL24aB4RBliWtcMqa1xY"
          lock_passwd: false
          groups: lxd
          shell: /bin/bash
          sudo: ALL=(ALL) NOPASSWD:ALL
          locale: fr_FR.UTF-8
          timezone: Europe/Paris
description: LXD VM profile
devices:
  eth0:
    name: eth0
    nictype: bridged
    parent: lxdbr0
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: vmubuntu
```

Lancez la commande suivante : lxc profile create vmubuntu cat | lxc profile edit vmubuntu Copiez-coller le fichier de configuration précédent et terminez par "CTRL D".

4. Création de la VM ubuntu orchestré par LXD : `lxc launch ubuntu:20.04 vmu20lxd --vm --profile vmubuntu`

5.

```
lxc operation ls

+--------------------------------------+------+------------------+--------
-+------------+----------------------+
| ID                                   | TYPE |    DESCRIPTION    |  STATUS
| CANCELABLE |     CREATED          |
+--------------------------------------+------+------------------+--------
-+------------+----------------------+
| 6377b880-f61d-4e3b-943a-c8294aa4ca6e | TASK | Creating instance | RUNNING
|    NO      | 2023/12/05 10:49 UTC |
+--------------------------------------+------+------------------+--------
-+------------+----------------------+
```

```
lxc list

+----------+---------+-------------------------+--------------------------
---------------------+-----------------+-----------+
|   NAME   |  STATE  |          IPV4           |
IPV6                    |          TYPE   | SNAPSHOTS |
+----------+---------+-------------------------+--------------------------
---------------------+-----------------+-----------+
| vmu20lxd | RUNNING | 10.250.25.158(enp5s0) |
fd42:dcc:9e96:b370:216:3eff:fe0e:8be8 (enp5s0)   | VIRTUAL-MACHINE |     0
|
+----------+---------+-------------------------+--------------------------
---------------------+-----------------+-----------+
*
```

```
lxc shell vmu20lxd

ssh ubuntu@10.250.25.158
```

J'ai bien accès en ssh et peux y accéder.

5 Autres solutions de containairisation système

5.1 Systemd sait tout faire même des containers...

Crééz des containers avec systemd-nspawn en vous aidant de : https://blog.selectel.com/systemd-containers-introduction-systemd-nspawn/

5.2 Footloose fait aussi des containers systèmes à partir d'images Docker

Utilisez footloose https://github.com/weaveworks/footloose afin de créer un container système ubuntu

20 (Attention utilisez la version 6.3). Vérifiez que systemd et ssh fonctionnent.