System Containers Unleashed ....

Jean-Marc Pouchoulon

Juin 2022

1 Compétences à acquérir :

1.1 Capacités

— Installer LXC
— Instancier un container LXC.
— Manipuler un container LXC (stop/start/clone...).
— Bâtir un nouveau bridge et connecter un container sur un autre Réseau.
— Bâtir des "network NameSpace".
— Créer un container non privilégié.
— 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.
— Comprendre les CGROUPS pour contraindre des processus : par exemple pour limiter la portée d'un déni de service.

1.3 Organisation, recommandation et notation du TP.

Vous travaillerez individuellement sur une machine virtuelle *VM Ubuntu 22.04 LTS* portée par VMwareWorkstation de préférence.

Il vous explicitement demandé de faire valider votre travail au cours du TP par l'enseignant pour être noté. Faites impérativement un compte rendu au fur et à mesure avec des copies d'écran et les configurations mises en oeuvre.

Tous les travaux sont à déposer sur l'ENT indiqué par l'enseignant. Un travail doit être enregistrer avec 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 respectés sous peine d'une note nulle.

2 Container LXC sous Linux.

2.1 Installation de LXC.

1. Installation des éléments nécessaires au TP :

a) Installez LXC :

```
apt-get install lxc bridge-utils lxc-templates debootstrap dnf debian-archive-keyring
```

b) Trouvez la commande qui permet de vérifier la bonne installation de LXC. (Les commandes lxc commencent toutes par lxc-...)

c) Quels sont les types de distribution Linux que vous pouvez containeriser sur cette VM ? (Allez voir le contenu de /usr/share/lxc/template).

2.2 Création d'un container LXC Debian buster.

1. Créez un container Debian buster 64 bits appelé *debian-j1*.
2. Récréez un container du même type appelé debian-j2. Que constatez-vous sur la vitesse de création ? qu'en déduisez-vous ?
3. A l'aide de la commande *lxc-create* créez un container *fedora* appelé fedora.
4. A l'aide de la commande *lxc-create* créez un container *centos 8* appelé centos. Essayez avec l'option –template=download et l'argument –no-validate. A quoi sert cette option et comment fonctionne-t-elle ?
5. Manipulation courante d'un container.
   Utilisez debian-j1 :
   a) Démarrez votre container.
   b) Arrêtez votre container.
   c) Redémarrez votre container en mode daemon.
   d) Attachez-vous à ce container pour lancer la commande "ip a".
   e) Retrouvez le PID du process du container et visualisez l'arborescence des processus rattachés à ce PID.
   f) Limitez la mémoire du container à 512 M et à 1 GIGA de Swap compris en modifiant la configuration du container (le CGROUP memory" est déjà autorisé dans le Kernel).
   g) Installez Apache2 à l'intérieur du container.
   Vous aurez les commandes suivantes à passer :

```
sudo sed -i -e 's,PrivateTmp=true,PrivateTmp=false\nNoNewPrivileges=yes,g' /lib/systemd/system/apache2.service
sudo systemctl daemon-reload
```

   h) Faites en sorte que le container démarre au démarrage de l'hôte.
   i) Figez et relancez le container.
   j) Clonez le container *debian-j1* en *clonedebian-j1*.
6. Comment le container est-il connecté au réseau ? Dessinez un schéma.
   Utilisez les commandes suivantes pour comprendre :

```
ip a
brctl show
iptables -L -n -v -t nat
```

   Allez voir aussi le contenu du fichier /etc/default/lxc .
   Connectez-vous au container afin de voir comment est paramétré le réseau vu du container.
7. Créez un nouveau bridge afin de connecter le container debian-j2 directement au réseau de la salle.
   Pour cela :
   — Créez une nouvelle interface eth1 sur votre VM directement connectée au réseau de la salle.
   — Modifiez /etc/netplan/01-netcfg.yaml afin de créer un nouveau bridge :

```
network:
 version: 2
 renderer: networkd
 ethernets:
   eth0:
       dhcp4: yes
   eth1:
       dhcp4: yes
 bridges:
   monbr0:
    interfaces: [eth1]
    dhcp4: yes
```

Redémarrez la couche réseau et vérifiez que le bridge monbr0 est bien créé :

```
netplan apply
```

— Modifiez la configuration du container sous /var/lib/lxc/debian-j2/config

Donnez une explication du fonctionnement de ce type de réseau.

8. A quel endroit sont stockés les containers sur votre machine physique ? Servez-vous de cette information afin de changer le mot de passe de votre container. Utilisez pour cela les commandes *chroot* et *passwd* .

2.3 Containers non privilégiés

1. Créez un container non privilégié avec le user student. Connectez vous en ssh sans sudo 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
```

3. Créez le 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. permettez à l'utilisateur student de créer "du réseau"

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

5. Créez et démarrer un container ubuntu "jammy"

```
lxc-create -t download -n ubuntu1 -- -r jammy -a amd64
```

6. Installez et lancez apache2. Comparer les processus Apache sur l'hôte et dans le container. Qu'en déduisez-vous du fonctionnement de LXC en mode non privilégié ?
7. Rendez à l'aide d'iptables le serveur apache accessible sur l'hôte.

3 Focus sur les briques de bases des containers

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

Les CGROUPs sont une brique de base de la containérisation. Ils permettent de contrôler les ressources affectées à un groupe de processus et donc à un ou plusieurs processus dans un container. A savoir :

- La commande cgclear vous permet de supprimer les cgroups.
- lscgroup vous permet de lister les cgroups.
- cat /proc/mount vous montre ce qui est ... monté (en particulier le FS cgroup)
- cat /proc/cgroups permet de voir les cgroups ou lssubsys

On va lancer deux process xterm, consommateurs de cpu et on va attribuer 80% du CPU au premier (xterm orange) et 20% du CPU au second (xterm bleu).

Comme vous n'avez pas d'interface graphique sur votre VM on va donc se servir de ssh pour forwarder la session X.

- 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
```

- passez les commandes suivantes :

```
systemctl ssh restart
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. Que donne la répartition du CPU entre les deux commandes ? Utilisez la commande top pour le voir.
2. Utilisez cgcreate , cgset, cgexec afin d'affecter 80% du CPU au premier xterm (orange) et 20% du CPU au second xterm (bleu). 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
```

3. Verifiez que la répartition CPU entre les deux process est bien maintenant de 80/20 entre les deux process.
4. Sous /sys/fs/cgroup retrouvez les modifications faites par les commandes précédentes. Expliquez le fonctionnement des commandes cg...

3.2 Manipulation du network namespace.

On se propose de créer deux networks namespaces, de leur affecter une paire de cartes virtuelles et de les lier par IP. C'est ce principe qui est couramment appliqué pour créer des containers. Pour connaitre les options liées à la manipulation des networks namespaces tapez :

```
ip netns help
```

1. Créez un network namespace *netns1* et network namespace *netns2* (Utilisez strace pour donner l'appel système utilisé pour la création d'un network namespace).
2. Exploration du netns1.
   a) Rattachez-vous à netns1.
   b) Quels est le device créé par default ?
   c) Que vous donne comme informations la demande ethtool -k ?
   d) Le device est-il migrable d'un netns à l'autre ?
3. Création de cartes virtuelles réseaux.
   a) Ajoutez un device veth :

      ```
      ip link add name vethnetns type veth peer name vethnetns-peer
      ```

   b) Affectez vethnetns-peer sur netns2 (ip link set ...) .
   c) Affectez via deux ip dans le même LAN et pinguez la carte de l'autre netns.

4 Création de containers LXD

LXD s'appuie sur LXC (liblxc) et permet de manipuler des containers à l'aide d'une API REST depuis une machine distante. Les commandes de gestion des containers sont différentes de LXC même si la couche sous-jacente est identique.

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. Listez les repositories des templates LXD et la liste des images sur le repository "images" :

   ```
   lxc remote list
   lxc image list images:
   ```

   En déduire indirectement la liste des distributions Linux supportées.

4.2 Création de containers sous LXD

1. Installer les 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. Lister les containers LXD.
3. Lancer un process bash dans le container centos 8.
4. Visualiser le bridge utilisé par lxd au travers des commandes :

```
lxc network list
lxc network edit nom_du_bridge_manage
```

5. Utiliser la commande suivante pour vous connecter à l'API rest de LXD.

```
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 .
```

   A quoi sert cette commande ? Qu'est qu'une socket du domaine Unix ?
6. Pilotage de lxd depuis un client réseau.
   Connectez-vous au lxd de votre voisin. Utilisez les commandes suivantes :

```
lxc remote add voisin1 ip_du_voisin
```

   Listez les containers de votre voisin faite un stop/start d'un de ses containers

L'idée de piloter des machines virtuelles comme des containers est vite devenue évidente afin de converger vers un système d'orchestration manipulant VM et containers. En voilà quelques exemples :

4.3 Utilisation de LXD comme orchestrateur de machines virtuelles

[1]

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 [2].

1. Installez le package whois et générer un mot de passe pour l'utilisateur de votre VM.

```
apt install whois
mkpasswd --method=SHA-512 --rounds=4096
```

2. Enlever les VMWareTools qui supplantent le module Kernel vsock standard nécessaire à lxd.

```
apt remove open-vm-tools
reboot
```

3. Créer un profil pour votre VM lxd.

1. voir `https://discuss.linuxcontainers.org/t/running-virtual-machines-with-lxd-4-0/7519` et `https://blog.simos.info/how-to-use-virtual-machines-in-lxd/`
2. https ://cloudinit.readthedocs.io/en/latest/topics/examples.html

```
lxc profile create vmubuntu
```

Créer le 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\$6UeFlXzgc\$m3i2tgsZGws0PuUoLJ71ew4y21UC....."
        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. Créez votre VM ubuntu orchestré par LXD :

```
lxc launch ubuntu:20.04 vmu20lxd --vm  --profile vmubuntu
```

5. Vérifiez-en le bon fonctionnement :

```
lxc list
+---------+---------+----------------------+----------------------------------------------+---------------+----------+
|  NAME   |  STATE  |         IPV4         |                    IPV6                      |     TYPE      | SNAPSHOTS |
+---------+---------+----------------------+----------------------------------------------+---------------+----------+
| vmu20lxd | RUNNING | 10.185.158.232 (enp5s0) | fd42:dcc:9e96:b370:216:3eff:fe0e:8be8 (enp5s0) | VIRTUAL-MACHINE | 0        |
+---------+---------+----------------------+----------------------------------------------+---------------+----------+
lxc shell vmu20lxd
ssh ubuntu@10.185.158.232
```

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.