System Containers Unleashed ....

Jean-Marc Pouchoulon

Décembre 2019

1 Objectifs du TP et organisation.

1.1 Compétences à acquérir :

— Choisir un type de container adapté au besoin rencontré parmi les solutions actuelles de containérisation.

— Créer un container rapidement pour des besoins de tests ou de production.

— Gérer les limites et les capacités d'un container.

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 portée un déni de service.

1.3 Organisation, recommandation et notation du TP.

Vous travaillerez individuellement sur une machine virtuelle *VM Ubuntu 18.04LTS* portée par VMware-Workstation. L'OVA est accessible sur `http://store.iutbeziers.fr`

Il vous explicitement demandé de faire valider votre travail au cours du TP par l'enseignant au fur et à mesure de votre avancement 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 purge lxd
apt-get install lxc  lxc-templates lxc-utils bridge-utils debootstrap yum libvirt0 libpam-cgfs
```

   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 stretch.

1. En positionnant la variable MIRROR sur le repo Debian de l'IUT de Béziers (debian.iutbeziers.fr) et à l'aide de la commande *lxc-create* créez un container Debian stretch 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 *centos 7* appelé centos7-1.
4. 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.
   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*.
5. 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.
6. 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 le fichier yml (50-cloud-init.yaml pour une VM multipass) sous /etc/netplan :

```
network:
  version: 2
  renderer: networkd
  ethernets:
     eth0:
       dhcp4: yes
     eth1:
       dhcp4: yes

bridges:
  monbr0:
    dhcp4: yes
  interfaces:
    - eth1
```

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

Donnez une explication du fonctionnement de ce type de réseau.

7. 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 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 :

— 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. Utiliser 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. Verifier que la répartition CPU 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 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.

3.1 Installation de LXD sous Ubuntu 16

1. Installation des éléments nécessaires au TP : Installer LXD en utilisant snap :

```
apt install snapd 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 :

```
    lxc remote list
```

   En déduire indirectement la liste des distribution Linux supportée.

3.2 Création de containers sous LXD

1. Installer les containers en suivant les instructions suivantes :

```
lxc launch images:debian/stretch debian
lxc launch ubuntu:16.04 ubuntu
lxc launch images:centos/7 centos
```

2. Lister les containers LXD.
3. Lancer un process bash dans le container centos 7.
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": "16.04"}}' a/1.0/containers | jq .
```

   A quoi sert cette commande (listez les containers) ? 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

4 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.

5 Autres solutions de containers systèmes

5.1 Faire un container système avec Docker

Un container système avec Docker est possible. Il suffit d'activer systemd dans le container. La société Weaveworks a créé le logiciel footloose afin de générer facilement des containers systèmes.

Installez l'outil en suivant les instructions de `https://github.com/weaveworks/footloose`

Créer ensuite 3 machines Debian et connectez vous en ssh dessus.

5.2 Systemd sait tout faire même des containers...

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