LAB n°1: Linux Containers

Description du thème

Propriétés	Description
Intitulé long	Linux Containers
Formation concernée	M2- Expert Réseaux
Matière	Cloud Computing
Présentation	L'objectif de ce Labo est de comprendre la conteneurisation sous Linux et de faire la différence avec les solutions de virtualisations traditionnelles. Ce Labo comporte 2 activités : -Activité 1 : Chroot. -Activité 2 : LXC
Compétences	-Chrooter un shell et des commandes -Manipuler des conteneurs Linux
Transversalité	Virtualisation avancées
Prérequis	-Commandes de base d'administration d'un système Linux. -Notions de virtualisation.
Outils	Un serveur physique ou virtuel avec une distribution Linux 64 bits (ici Centos 7– version stable actuelle).
Mots-clés	chroot – jail- conteneur linux
Auteur	Slim Marghli
Version	V1.0
Date de publication	Septembre 2020

Présentation de l’isolation

L'isolation (aussi appelé cloisonnement) est une technique qui intervient au sein d’un même système d'exploitation. Elle permet de séparer un système en plusieurs contextes ou environnements. Chacun d'entre eux est régi par l'OS hôte, mais les programmes de chaque contexte ne peuvent communiquer qu'avec les processus et les ressources associées à leur propre contexte.

Il est ainsi possible de partitionner un serveur en plusieurs dizaines de contextes, presque sans ralentissement.

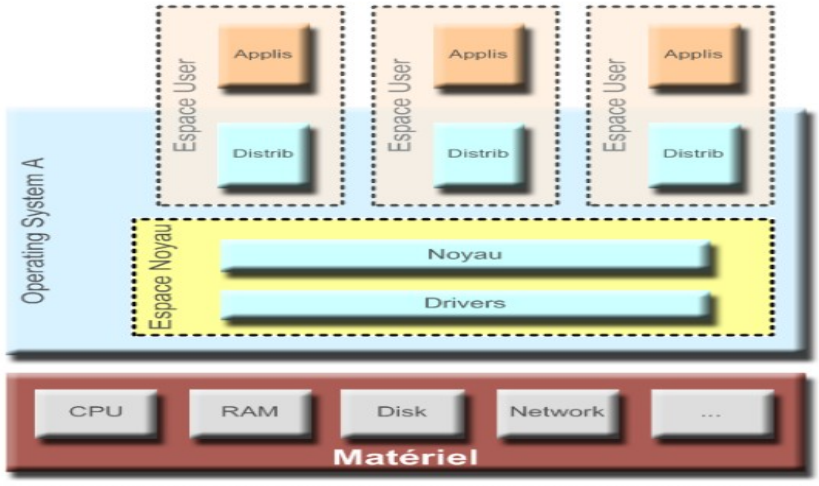
L'isolation est utilisée sous Unix depuis longtemps pour protéger les systèmes. Via des mécanismes comme chroot ou jail il est possible d'exécuter des applications dans un environnement qui n'est pas celui du système hôte, mais un « mini système » ne contenant que ce dont l'application a besoin, et n’ayant que des accès limités aux ressources. Il est possible également de lancer des programmes dans une autre distribution que celle du système principal.

Avec l’isolation, l’espace noyau n’est pas différencié, il est unique, partagé entre les différents contextes. Mais on définit de multiples espaces utilisateurs cloisonnés. C’est ainsi que l’on peut faire cohabiter différentes distributions de système d’exploitation, à condition qu’elles partagent le même noyau.

L’isolation des contextes est une solution légère, tout particulièrement dans les environnements Linux.

L’unicité du noyau reste bien sûr une petite limitation. D’une part en termes de robustesse, puisqu’un plantage du noyau – fort heureusement très rare dans le monde Linux – plante simultanément tous les environnements. D’autre part dans les utilisations possibles, puisque typiquement ce mode ne conviendra pas pour valider une nouvelle version de noyau.

Mais pour les besoins les plus courants de la virtualisation, la simplicité de mise en œuvre et le faible overhead sont d’excellents arguments.



Activité n°1: Utilisation d'une chroot pour chrooter un shell

Rappel:

Que fait la commande chroot ?

chroot est un appel système permettant de faire tourner un programme dans un environnement restreint. La commande chroot prend deux paramètres : le chemin du nouveau répertoire racine (/), et le chemin du programme à lancer (relatif au premier argument).

chroot /path/to/new/root /path/to/command

Problème lié à un environnement restreint

Le problème est lié aux librairies partagées qu'il va falloir identifier et recopier dans la jail. La commande **ldd** permet d'identifier les librairies partagées dont le programme a besoin.

Manipulations pratiques

1. Localisez le binaire relatif au bash.

```
$which bash
 /usr/bin/bash
```

2. Créez un répertoire qui va héberger notre mini-linux.

```
$ mkdir jail
```

3. Créez les répertoires de base dans votre jail.

```
$ mkdir -p jail/usr/bin
$ mkdir -p jail/lib64
```

4.Copiez bash dans votre répertoire bin situé dans le jail que vous avez crée

```
$ cp /usr/bin/bash jail/usr/bin
```

5.Utilisez la commande **ldd** pour localiser les bibliothèques nécessaires au fonctionnement de bash

```
$ldd /usr/bin/bash
      linux-vdso.so.1 =>  (0x00007ffd4a28e000)
      libtinfo.so.5 => /lib64/libtinfo.so.5 (0x00007ffb187e8000)
      libdl.so.2 => /lib64/libdl.so.2 (0x00007ffb185e4000)
      libc.so.6 => /lib64/libc.so.6 (0x00007ffb18216000)
      /lib64/ld-linux-x86-64.so.2 (0x00007ffb18a12000)
```

6. Copiez ces bibliothèques dans le répertoire convenable de votre jail

```
$cp /lib64/{ libtinfo.so.5, libdl.so.2, libc.so.6, ld-linux-x86-64.so.2}
jail/lib64
ou
# !/usr/bin/bash
echo "cmd name without path"
read cmd
echo " lib path in jail"
read lib_path
list="$(ldd /usr/bin/$cmd | egrep -o '/lib.*\.[0-9]')"
for i in $list; do cp  -v "$i" "$lib_path"; done
```

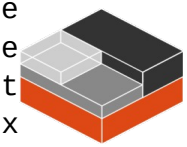
7. Créez l'environnement restreint avec chroot:

```
# chroot jail /usr/bin/bash
```

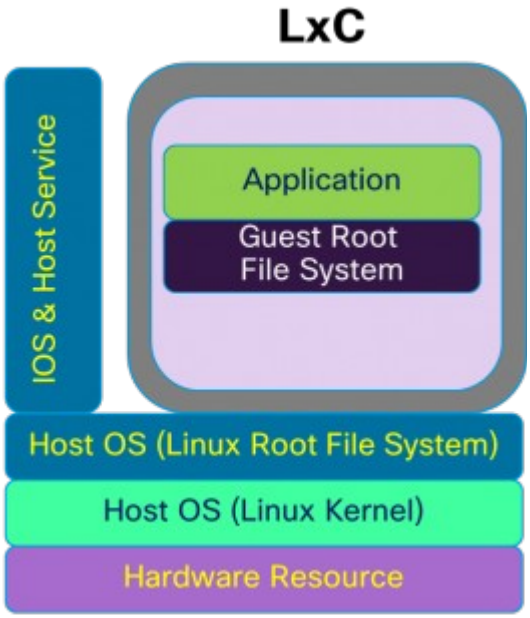
8. Exécutez la commande pwd dans votre jail. Que peut-on déduire?
9. Exécutez la commande ls dans votre jail. Procéder à la résolutions des problèmes détectés.

Activité n°2: Les conteneurs LXC

Rappel: LXC



LXC, contraction de l'anglais **Linux Containers** est un système de virtualisation, utilisant l'isolation comme méthode de cloisonnement au niveau du système d'exploitation. Il est utilisé pour faire fonctionner des environnements Linux isolés les uns des autres dans des conteneurs, partageant le même noyau et une plus ou moins grande partie du système hôte. Le conteneur apporte une virtualisation de l'environnement d'exécution (processeur, mémoire vive, réseau, système de fichier…) et non pas de la machine. Pour cette raison, on parle de « conteneur » et non de « machine virtuelle ».
[source : https://fr.wikipedia.org/wiki/LXC]



Manipulations pratiques

1. Installez LXC sur votre machine. Pour pouvoir créer un container LXC sur CentOS, vous devez installer 3 paquets principaux :**lxc, lxc-templates, libvirt**. Le paquet libvirt permet de disposer d'une interface réseau virtuelle du type virbr0.

```
#yum install epel-release
#yum install libvirt lxc lxc-templates
```

2. Une fois installés, les modèles de containers disponibles sur CentOS sont présents dans le dossier **/usr/share/lxc/templates**. Listez le contenu de ce dossier.

```
$ls -l /usr/share/lxc/templates
```

3. Vérifiez l'environnement LXC en tapant **lxc-checkconfig** sur la ligne de commandes.

```
$lxc-checkconfig
```

4. Activez et démarrez les services libvirtd et lxc :

```
#systemctl enable libvirtd
```

#systemctl enable lxc
#systemctl start libvirtd
#systemctl start lxc

5. Affichez la liste des commandes de LXC.

$lxc- [tabulation]

6. Les containers LXC se créent à l'aide de la commande **lx-create** . Par défaut, ils sont stockés dans le dossier **/var/lib/lxc/**. Le paramètre **-t** représente le modèle de système Linux. Créez un container centoslxc en se basant sur le template centos.

#lxc-create -t centos -n centoslxc

7. Le mot de passe temporaire du container est stocké dans **/var/lib/lxc/centoslxc/tmp_root_pass**. Affichez ce mot de passe.

#cat /var/lib/lxc/centoslxc/tmp_root_pass

8. Pour changer le mot de passe sans avoir à se connecter au container LXC, Il faut "chrooter" son système de fichier. Changez le mot de passe du container.

#setenforce 0
#chroot /var/lib/lxc/centoslxc/rootfs passwd

9. Démarrez votre container : utilisez la commande **lxc-start (**Pour démoniser ce container, autrement dit pour le lancer en service , employez le commutateur **-d**)

#lxc-start -n centoslxc -d

10. Vérifiez que votre container est en exécution.

#lxc-info -n centoslxc

11. Exécutez la commande **lxc-info -n centoslxc -i.** Que permet le commutateur **-i ?**

Le commutateur -i permet de connaître l'adresse ip du container, afin de s'y connecter en SSH.

12. Le fichier de configuration du container centoslxc se trouve à l'emplacement **/var/lib/lxc/centoslxc/config**. Modifiez ce fichier afin que le conteneur démarre automatiquement.

Il faut ajouter à ce fichier :
lxc.start.auto = 1
Le fichier **/var/lib/lxc/centoslxc/config** est lui-même hérité du fichier **/etc/lxc/default.conf**.

13. Connectez en mode root au container avec la commande l**xc-attach** et exécuter dedans les commandes yum install openssh-server, systemctl start sshd et systemctl enable sshd .

lxc-attach -n centoslxc
La commande lxc-attach permet d'exécuter des commandes dans le container à partir de la machine physique en tant que root.
lxc-attach -n centoslxc yum install openssh-server
lxc-attach -n centoslxc systemctl start sshd
lxc-attach -n centoslxc systemctl enable sshd

14. Connectez vous à votre container et créer un fichier dans son système de fichier.

lxc-attach -n centoslxc

```
date>fdate
exit
```

15. A partir du terminal de votre machine hôte, accédez au système de fichier de votre container et vérifiez l'existence du fichier crée lors de la question précédente.

```
cd /var/lib/lxc/centoslxc/rootfs/root
ls
fdate
```

16. Arrêter le container avec la commande **ssh root@adresse_ip_du_container 'poweroff'**

```
#ssh root@adresse_ip_du_container 'poweroff'
```

il est possible aussi d'utiliser la commande: lxc-stop -n centoslxc

17. Veuillez détruire le container créé précédemment.

```
lxc-destroy -n centoslxc
rm -fr /var/lib/lxc/centoslxc
```