Containers et Virtualisation

Les containers et les machines virtuelles (VM) sont deux technologies de virtualisation différentes, utilisées pour isoler et exécuter
des applications de manière efficace. Voici une brève explication de chacune d'entre elles :

Containers (Conteneurs) :

1 Les containers sont une forme de virtualisation légère qui permet d'isoler les applications et leurs dépendances au niveau du système d'exploitation.

2 Ils partagent le noyau du système d'exploitation de l'hôte, ce qui les rend plus légers en termes de ressources par rapport aux machines virtuelles.

3 Les containers encapsulent l'application, ses bibliothèques et ses dépendances, créant ainsi un environnement exécutable indépendant.

Machines Virtuelles (VM) :

1 Les machines virtuelles sont des environnements d'exécution complets et indépendants qui émulent des ordinateurs physiques.
Chaque VM dispose de son propre système d'exploitation, de son propre noyau et de ses propres ressources matérielles virtuelles.

2Elles permettent d'exécuter plusieurs systèmes d'exploitation sur une seule machine physique.

3Les VM nécessitent plus de ressources que les containers car elles incluent l'ensemble du système d'exploitation, même si plusieurs
VM peuvent partager les ressources physiques de la machine hôte.

VM vs Containers

Pourquoi virtualiser ?

La virtualisation offre plusieurs avantages significatifs dans le domaine de l'informatique et de la gestion des systèmes.
Voici quelques-unes des raisons courantes pour lesquelles la virtualisation est largement utilisée :

Optimisation des ressources : La virtualisation maximise l'utilisation des ressources matérielles en exécutant plusieurs machines virtuelles ou
containers sur une seule machine physique, améliorant ainsi l'efficacité du processeur, de la mémoire et du stockage.

Isolation et sécurité : Les machines virtuelles et les containers offrent une isolation entre les applications, assurant que la compromission
d'un conteneur ou d'une machine virtuelle n'affecte pas les autres, renforçant ainsi la sécurité du système.

Flexibilité et agilité : La virtualisation permet une création, un déploiement et un déplacement rapides des machines virtuelles ou des conteneurs,
facilitant la gestion des charges de travail, la mise à l'échelle des applications et réduisant les délais de déploiement.

Comment ?

Un hyperviseur est un logiciel responsable de la création, de l'exécution et de la gestion des machines virtuelles. Il virtuellement partage les ressources de la machine hôte entre différentes VM. Il existe deux types d'hyperviseurs pour les machines virtuelles :

Type 1 (Bare metal, native) : S'exécute directement sur le matériel de l'hôte, sans nécessiter un système d'exploitation hôte supplémentaire. Exemples : VMware ESXi, Ms Hyper-V, Oracle VM, KVM.

Type 2 (hosted, hébergé) : Est une solution logicielle qui s'exécute sur un système d'exploitation existant. Exemples : Oracle VirtualBox, VMware Workstation.

Comparatifs

Comparatifs

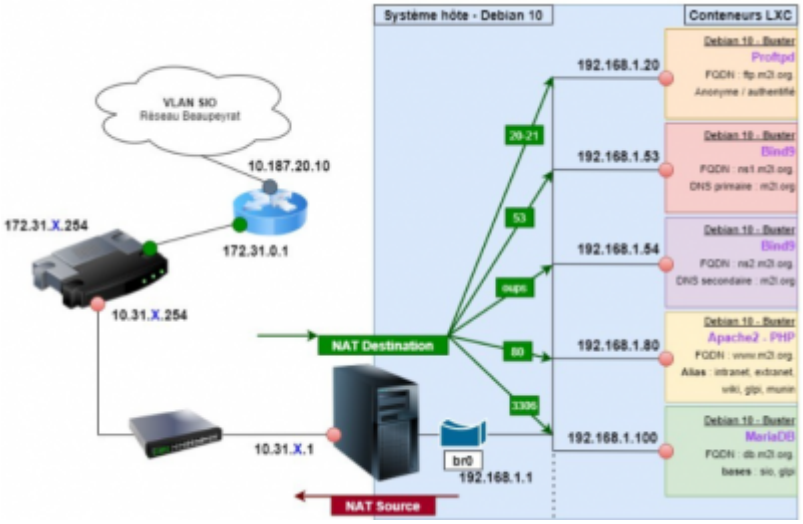
Machine Virtuelles	**Containers**
Volumineuses	Légers
Performances limitées (par l'hyperviseur)	Performances de l'OS hôte
Un OS par machine	Un OS (hôte) pour tous les conteneurs
Virtualisation Hardware	Virtualisation OS
Démarrage lent	Démarrage « instantané »
Besoins en mémoire importants	Besoins en mémoire faibles
Isolation totale = sécurité	Isolation niveau processus

Mise en place de conteneurs LXC : Stratégie

Mise en place dans un réseau privée

1 Leur accès à Internet est assuré, au niveau du serveur, par une règle iptables de NAT Source.
2 L'accès aux services qu'ils offrent depuis l'extérieur est assuré, au niveau du serveur, par des

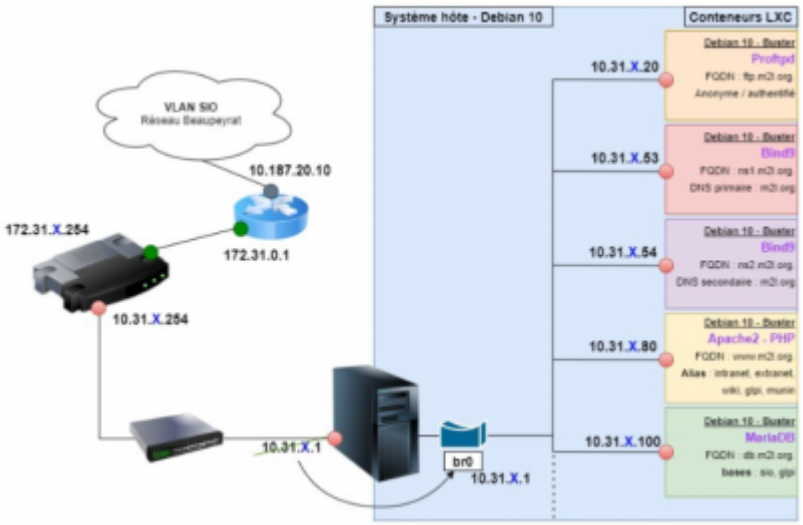
réglesiptables de NAT Destination



La solution protège les conteneurs en évitant leur exposition directe sur Internet, assurant ainsi la sécurité des serveurs internes.
Cependant, l'accès au serveur DNS secondaire depuis l'extérieur est limité, malgré la possibilité d'accéder au serveur DNS primaire via une règle DNAT.

Mise en place sur le même réseau publique que l'hôte

1 Comme pour le serveur, leur accès à Internet est assuré, par le routeur.
2 L'accès aux services qu'ils offrent depuis l'extérieur est assuré grâce à leurs adresses IP



Les problèmes évoqués dans la stratégie n°1 sont résolus ici, cependant chaque machine est exposée à l'extérieur.

Mise en place sur différents réseau

1 un public pour les services devant être accessible de l'extérieur

2 un privé pour les services ne devant pas être accessibles depuis l'extérieur

Les communications entre les deux réseaux sont assurées par routage. \\

TP LXC

Noms des machines durant le TP

host# commande	La commande est tapée sur la machine hôte
nom# commande	La commande est tapée dans le conteneur «nom»

Mise à jour du système:

Premièrement il faut régulièrement mettre à jour les paquets du système.

```
apt-get update
apt-get upgrade
```

Installation de LXC:

Nous allons procédé à installation de LXC avec la commande :

```
apt-get install lxc
```

Vérification de la configuration avec la commande :

```
lxc-checkconfig
```

Configuration du pont réseau:

Installation de bridge-utils avec la commande :

```
apt-get install bridge-utils
```

Création d'un premier pont :

Création du pont br0 avec la commande :

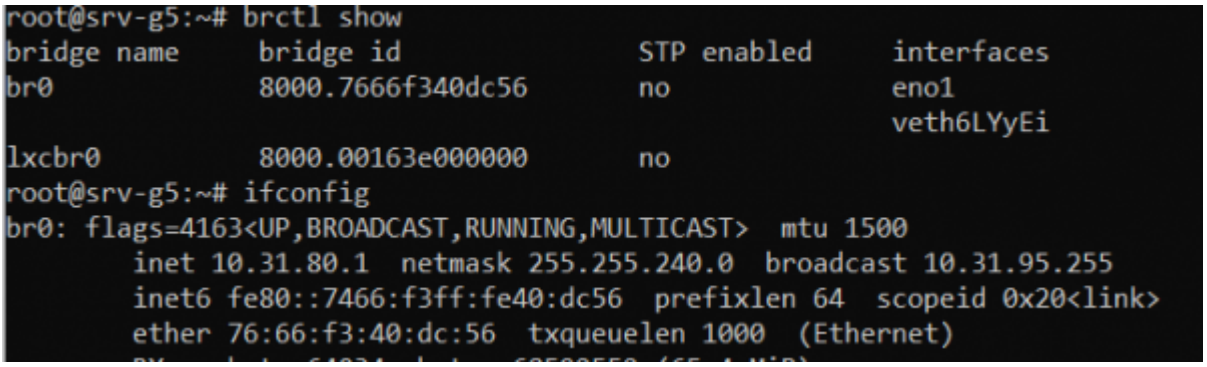
```
brctl addbr br0
```

Vérification :

suite a la création de notre pont nous allons vérifier si la nouvelle carte réseau a bien été crée grâce a la commande :

```
brctl show
ifconfig
```

Image de vérification :



Configuration IP :

1 Se rendre sur le fichier rc.local et remplacer la carte réseau précédente par la nouvelle carte réseau crée :

```
nano /etc/rc.local
```

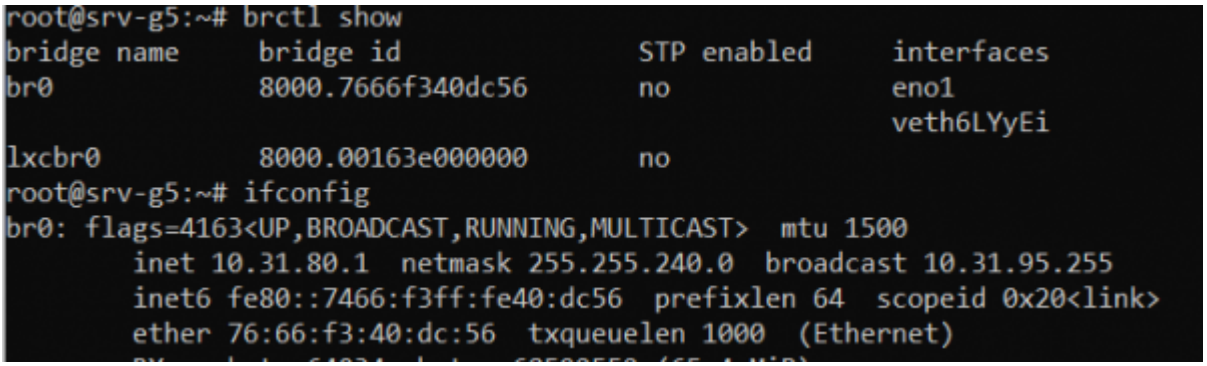
2 Configurer l'ancienne carte réseau sur 0.0.0.0 et la nouvelle carte réseau par l'adresse IP du serveur :

N'oublier pas la Commande “brctl addif br0 enp0s3” à ajouter dans le fichier rc.local

```
#!/bin/sh -e
ifconfig eno1 0.0.0.0
brctl addbr br0
ifconfig br0 10.31.80.1/20
route add default gw 10.31.95.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
brctl addif br0 eno1
```

vérification :

Vérifier si l'interfaces de br0 et bien eno1 et vérification de la configuration IP , images :

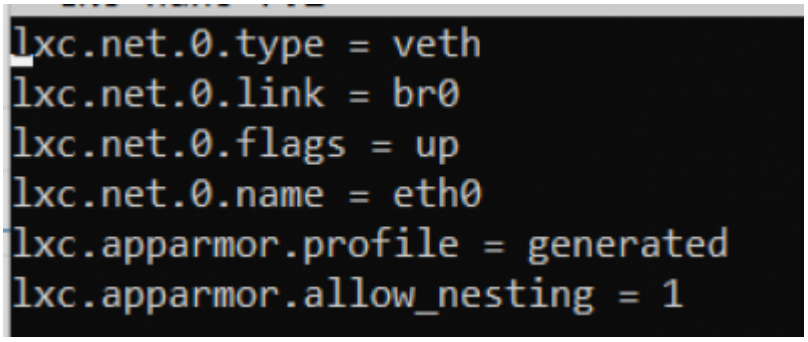


Vérification de la configuration des conteneurs :

Nous allons procédé à la vérification de la configuration des containers si dessous la commande pour se rendre sur le fichiers :

```
host# nano /etc/lxc/default.conf
lxc.net.0.type = veth
lxc.net.0.link = br0
lxc.net.0.flags = up
lxc.net.0.name = eth0
lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1
```

Image de vérification :



Création d’un premier conteneur - template :

Nous allons procédé al la création du premiers containers qui servira de template commande utiliser ci dessous :

```
lxc-create -n template -t debian -- -r bookworm
```

Vérification :

Vérifier si le nouveau containers a bien été crée :

```
lxc-ls
```

Affichage des infos :

pour afficher les infos du containers template nous utiliserons la commande :

```
lxc-info template
```

Démarrage du conteneur template :

1 mettre le template en “on”
2 vérifier si la templete est en “on”

```
lxc-start template
lxc-info template
```

Démarrage automatique d’un container :

Pour que la template démarre automatiquement il faut éditer le fichier config de template, commande ci dessous :

```
nano /var/lib/lxc/template/config
# ajouter la commande ci dessous dans ce fichier
lxc.start.auto = 1
```

Configuration IP temporaire du conteneur pour installation d’outils :

Pour configurer les adresse IP de manière définitive nous allons procéder à la modification du fichier '/etc/network/interfaces'.

Configuration : nous allons donner l'adresse IP 10.31.80.2 a la template

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 10.31.80.2/20
        gateway 10.31.95.254
        dns-nameservers 8.8.8.8
```

Installation d'outils :

nous venons de donner accès a internet au container grâce a la configuration IP suite à ça nous allons installer les outils nécessaire :

```
apt update
apt upgrade
apt install sudo net-tools tcpdump nano iputils-ping dbus
```

Configuration de la timezone :

Grâce a la timezone nous allons configurer le container sur la timezone de sa zone géographique :

```
ln -fs /usr/share/zoneinfo/Europe/Paris /etc/localtime
dpkg-reconfigure -f noninteractive tzdata
```

Facultatif : Configuration du sudo

premièrement il faut crée un nouveau utilisateur, puis nous lui configurerons ces permission d'utilisation de certain droit root :

```
adduser std
usermod -a -G sudo std
```

Arrêt du conteneur template et copie

Pour stopper le container le container : commande

```
lxc-stop template
```

Copie du containers

ce containers que nous venons de crée servira de template pour le dupliquer :

```
lxc-copy -n template -N  web
```

Renommer le container web :

Suite a la copie on renomme le container en web :

```
hostnamectl set-hostname web
```

Configuration ip du container web :

configurer l'adresse IP du web car elle est encore configurer sur l'IP de la template :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 10.31.80.80/20
        gateway 10.31.95.254
        dns-nameservers 8.8.8.8
```