

LE LAB

Ansible

Les VMS

Debian 13 netinstall.iso

Configurations Minimales

Le contrôleur Ansible :

- 4G de RAM
- 2 processeurs
- 60 G DD

Installation :

- Hostname : debian-ansible
- Pas de mot de passe root (sudo)
- LVM (tout dans une seule partition)
- Installation Gnome FlashBack à la place de Gnome

La plate-forme LXC

- 2G de RAM
- 1 proc
- 20G DD

Installation :

- Hostname : debian-lxc
- Pas de mot de passe root (sudo)
- LVM (tout dans une seule partition)
- Installation : pas de bureau, serveur SSH

Post-installation debian-lxc

Installation de lxc :

apt install lxc

Lxc vient avec un bridge par défaut, mais nous allons configurer nous-même nos bridges :

vi /etc/default/lxc-net

USE_LXC_BRIDGE="false"

vi /etc/lxc/default.conf

On remplace la ligne correspondante à lxcbr0 par br0

en y rajoutant les lignes :

lxc.net.0.hwaddr = xx:xx:xx:xx:xx:xx (des x minuscules...)

Ce qui fixera les adresses MAC de nos containers...

Et lxc.start.auto = 1

Pour démarrer automatiquement nos containers au démarrage de l'hôte.

Installation du bridge

N.B. :

Pour VirtualBox : Dans la configuration réseau de VirtualBox → avancé → mode promiscuité → tout autoriser

Pour HyperV : Carte réseau → Fonctionnalités avancées → Activer l'usurpation d'adresse MAC

Configuration de l'adresse statique:

Avec l'adresse attribuée par le serveur dhcp ou avec une adresse en dehors de la plage dhcp...

N.B. Jusqu'à Debian 12, lxc venait avec le paquet bridge-utils qui permettait de configurer les bridges dans le fichier /etc/network/interfaces, ce qui n'est plus le cas... Nous allons donc devoir utiliser systemd-networkd...

Pour trouver les informations dont nous avons besoin :

ip a # adresse et masque

ip route # gateway (route par défaut)

cat /etc/resolv.conf # DNS resolver

N.B. Ici notre interface s'appelle enp1s0 : à adapter à votre cas.

```
vi /etc/network/interface
```

```
iface enp1s0 inet manual
```

```
vi /etc/systemd/network/br0.netdev
```

```
[NetDev]
```

```
Name=br0
```

```
Kind=bridge
```

```
vi /etc/systemd/network/br0.network
```

```
[Match]
```

```
Name=br0
```

```
[Network]
```

```
Address=192.168.122.x/24
```

```
Gateway=192.168.122.1
```

```
DNS=192.168.122.1
```

```
vi /etc/systemd/network/enp1s0.network
```

```
[Match]
```

```
Name=enp1s0
```

```
[Network]
```

```
Bridge=br0
```

```
sudo apt install systemd-resolved
```

Ce n'est plus le serveur dhcp qui nous indique notre resolver !

```
sudo networkctl
```

```
sudo systemctl enable systemd-networkd
```

```
sudo reboot
```

Contrôler avec :

```
ip a
```

```
ip route
```

```
cat /etc/resolv.conf
```

Création de nos containers :

sudo su -

lxc-create -n dmz -t download

distribution: debian

release : trixie

architecture : amd64

N.B. En une seule ligne de commande :

lxc-create -t download -n dmz -- -d debian -r trixie -a amd64

Le container est créé dans /var/lib/lxc/dmz/

Et son système de fichier dans /var/lib/lxc/dmz/rootfs/

Pour démarrer/stopper le container :

lxc-start dmz

lxc-stop dmz

Pour s'y connecter :

lxc-attach dmz

Pour lister les différents containers :

lxc-ls -f

On peut créer ainsi différents containers rockylinux, debian, ubuntu pour notre lab

Post-installation debian-ansible

Installation de git et d'un utilisateur dédié

sudo apt install git

sudo adduser git

su – git

git init --bare test.git

exit

ssh-keygen

ssh-copy-id [git@localhost](#)

git clone git@localhost:test.git

cd test/

...