TP : Création d'une machine virtuelle Debian en console, puis test via KVM

Table des matières

A) Introduction

Sous *GNU/Linux*, on utilise généralement **virt-manager** en mode graphique pour créer et gérer des machines virtuelles, via la virtualisation *KVM* intégrée nativement au noyau *Linux*.

Le but de ce TP sera de « créer » une VM depuis la console (donc sans mode graphique), comme si on devait, par exemple, installer une distribution *GNU/Linux* minimaliste sur un matériel embarqué, et que la seule chose dont on dispose est le dossier de stockage de l'appareil, « monté » dans un répertoire de notre machine hôte.

Ce TP démontre donc qu'on peut complètement installer & configurer une distribution *GNU/Linux* en console.

La configuration suivante supposera de travailler sur une machine *GNU/Linux*, préinstallée avec une distribution de type *Debian* / *Ubuntu* / *Mint* / etc.

Nous supposons ici travailler sous l'utilisateur « stagiaire », lequel dispose des droits **sudo** pour exécuter des commandes administrateur sur le système.

B) Avant de commencer…

1) Ancien schéma MBR

Historiquement, les anciens *BIOS* utilisent un schéma de partitionnement de type **MBR** (*Master Boot Record*), limité à 4 partitions primaires, ou 3 partitions primaires + 1 étendue, laquelle peut elle-même contenir d'autres lecteurs.

Une de ces partitions doit ainsi recevoir le drapeau `boot`. Le *BIOS* la repère au démarrage et lance le système présent par défaut.

Sous *GNU/Linux*, il faut utiliser `fdisk` pour utiliser le schéma *MBR*.

2) Nouveau schéma UEFI

Les nouveaux *BIOS UEFI* ont introduit un nouveau schéma de partitionnement nommé **GPT** (*GUID Partition Table*, avec *GUID* = Globally Unique Identifier), qui ne fonctionne plus avec des partitions primaires/étendue, mais avec des identifiants uniques de partitions - comprenez : une chaîne de caractère unique dans le système, qui va identifier chaque partition présente.

Par défaut, le *BIOS* au démarrage recherche une partition de type *EFI/ESP* (obligatoire), généralement formatée en fat32 sur 100 Mo. Cette partition contient un dossier *EFI* (en majuscules) et un sous-

dossier par système d'exploitation installé (« **debian** » par exemple).

Dans chaque sous-dossier, on trouve obligatoirement un fichier avec l'extension « **.efi** » qui constitue le point d'entrée (exemple grubx64.efi pour le chargeur de démarrage sous *GNU/Linux*).

Ainsi le *BIOS* au démarrage scanne la partition *EFI*, repère les systèmes installés sur le disque, et se fabrique un menu de démarrage dont la présentation varie suivant les constructeurs.

N.B. : pour installer plusieurs systèmes d'exploitation sur le même compatible PC physique, il est recommandé de désactiver l'option **`Secure Boot`** dans le *BIOS*.

Certains fabricants, pour compliquer les choses, mettent par défaut cette option en grisé. L'astuce consiste alors à enregistrer un mot de passe administrateur *BIOS*, redémarrer, et revenir dans le *BIOS* pour la débloquer...

Sous *GNU/Linux*, il faut utiliser **`gdisk`** pour utiliser le schéma GPT.

C) Mise en œuvre

1. Installation des paquets pour le TP :

```
apt install virt-manager qemu-kvm
```

Petit bug (?) : l'activation manuelle du démon **libvirtd** n'est parfois pas prise en compte tout de suite… Un petit redémarrage peut donc être nécessaire la première fois

```
reboot
```

D) Après redémarrage - sous « stagiaire »

1. Passage en administrateur :

```
whoami
sudo -s
whoami
```

2. Ajout de l'usager *stagiaire* au groupe **kvm** :

```
groups stagiaire
adduser stagiaire kvm
groups stagiaire
newgrp kvm
```

N.B. : on suppose ici que le groupe *stagiaire* existe et a déjà été créé avec l'usager *stagiaire*...

3. Création du dossier de travail :

```
mkdir /home/stagiaire/kvm
cd /home/stagiaire/kvm
```

4. Création du conteneur principal :

La commande suivante va créer un fichier de 12 Go en mode dynamique (*copy on write* ou *cow*). Cela signifie concrètement que les 12 Go ne seront pas immédiatement consommés sur le disque dur cible. En fait, le conteneur se remplira petit à petit, jusqu'à une taille maxi de 12 Go. Vous aurez compris que ce mode particulier permet d'économiser de l'espace disque non consommé, et donc de créer plus de machines sur un même disque physique.

```
qemu-img create -f qcow2 debian.qcow2 12G
```

5. Bouclage du conteneur :

```
modprobe nbd max_part=4
qemu-nbd -c /dev/nbd0 debian.qcow2
```

6. Création des partitions en mode *NON UEFI* (anciens *BIOS*) :

Ici, le conteneur sera initialisé avec 2 partitions :

- ➔ **une partition d'échange (swap) de type 82**, et de taille 2Go. Cette partition était historiquement utile sur les système disposant de peu de mémoire vive (RAM). Quand le noyau n'avait plus de place mémoire, il en écrivait une partie sur le disque pour pouvoir continuer ses opérations. Avec les machines modernes, cette partition d'échange n'est plus vraiment nécessaire. Ici on l'a conservée pour l'exemple.
- ➔ **une partition en ext4 de type 83**, le système de fichiers par défaut de *GNU/Linux*, qui occupera le reste du conteneur.

Attention : après le **`fdisk`**, ce sont des combinaisons de touches, qu'il faut taper les unes après les autres !

`n` = new, **`p`**=partition, **`t`**=type, **`p`**=print (1er niveau), **`w`**=write, **`a`**=flag

```
fdisk /dev/nbd0
n p 1 ENTER +2G t 82 p
n p ENTER ENTER ENTER p
a 2 p
w
```

7. Relecture des partitions du conteneur par le noyau :

```
kpartx -av /dev/nbd0
```

8. Formatage des partitions :

```
mkswap /dev/nbd0p1
mkfs.ext4 /dev/nbd0p2
```

9. Montage de la partition *ext4* du conteneur dans ***/mnt/*** :

```
mount /dev/nbd0p2 /mnt
```

10. Installation de la *Debian* en version *stable* dans ***/mnt*** :

Pour cela, il faut d'abord installer le paquet **debootstrap**, puis l'utiliser en lui indiquant les paquets additionnels que l'on veut voir installés dans le conteneur (option *--include*), la version souhaitée de la *Debian* (ici *stable*), et surtout le dossier dans lequel on veut installer la distribution (ici ***/mnt***).

```
df
apt-get install debootstrap

# Attention : la commande suivante doit tenir sur une seule ligne ! Vérifiez bien vos arguments avant de la lancer !

debootstrap –include=less,locales-all,vim,gpm,sudo,openssh-server,grub-pc,manpages-fr stable /mnt

df
```

11. Translation dans le conteneur :

Il s'agit de lier 3 dossiers virtuels clés (***/dev***, ***/proc*** et ***/sys***), depuis la distribution en cours d'utilisation, vers le dossier contenant le nouveau système, puis d'utiliser la commande **`chroot`** qui va permettre de se placer dans ce nouveau système exactement comme si on l'avait déjà démarré.

C'est la même technique que l'on utilise quand on veut récupérer un système *GNU/Linux* planté : on démarre sur une clé USB de secours, avec un autre système *GNU/Linux* léger. On se connecte en administrateur (**`su -l`** ou **`sudo -s`** suivant la distribution). Avec **`lsblk -f`**, on repère la partition principale du système à réparer. On monte cette partition à réparer avec la commande **`mount`**, généralement dans ***/mnt***, puis on effectue ces mêmes 4

opérations, et on arrive sur la partition principale à réparer comme si on avait démarré le système planté.

```
mount --bind /dev /mnt/dev
mount --bind /proc /mnt/proc
mount --bind /sys /mnt/sys
chroot /mnt
```

12. Configuration des partitions à monter au démarrage dans ***/etc/fstab*** :

```
cat /etc/fstab

# Attention : la commande suivante est sur une ligne !
echo "/dev/sda1 none swap defaults 0 0" >> /etc/fstab

# Attention : la commande suivante est sur une ligne !
echo "/dev/sda2 / ext4 errors=remount-ro 0 1" >> /etc/fstab

cat /etc/fstab
```

13. Attribution d'un mot de passe à **root** (ne jamais faire ça en production : normalement, il faudrait créer un autre usager, et lui donner des droit administrateurs temporaires, via **`sudo`** ou autres techniques…) :

```
passwd root
```

14. Configuration du chargeur de démarrage (**grub**) et des options noyau au démarrage (ici **`net.ifnames=0`** pour renommer les interfaces réseaux en **`eth0`**, **`eth1`**, **`eth2`**, etc, et **`ipv6.disable=1`** pour désactiver l'IPv6 par défaut, et s'éviter des attaques et autres exploits) :

```
# Attention : la commande suivante est sur une ligne !
sed -i
's/^GRUB_CMDLINE_LINUX_DEFAULT="quiet"/GRUB_CMDLINE_LINUX_DEFAULT="net.ifnames=0 ipv6.disable=1"/' /etc/default/grub

update-grub
grub-install /dev/nbd0
```

15. Ajout du noyau par défaut :

```
apt install linux-image-amd64
```

16. Configuration du nom d'hôte dans ***/etc/hostname*** :

```
echo "debian" > /etc/hostname
```

17. Configuration réseau :

```
echo "
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet dhcp
" >> /etc/network/interfaces
```

18. Configuration de la localisation :

```
apt install locales console-data
dpkg-reconfigure locales
```

```
debconf-show locales
```

19. Configuration de la disposition du clavier :

```
apt install keyboard-configuration
debconf-show keyboard-configuration
```

20. Configuration de la langue en console :

```
apt install console-setup
debconf-show console-setup
```

21. Configuration du fuseau horaire :

```
dpkg-reconfigure tzdata
debconf-show tzdata
```

22. Ajouts de 6 terminaux virtuels :

    En général dans les distribution, pour basculer du mode graphique au mode texte/console, il faut utiliser les raccourcis **`CTRL+ALT+F1`**, **`CTRL+ALT+F2`**, **`CTRL+ALT+F3`**, etc.

    Une fois en mode texte/console, on peut aussi utiliser les raccourcis **`ALT+flèche gauche/droite`** pour changer de terminal virtuel.

    Par défaut, un seul terminal est disponible. Pour en activer d'autres, dans les distributions utilisant *systemd*, il faut modifier ***/etc/systemd/logind.conf***.

    Ici on utilise une fonctionnalité de rechercher/remplacer du texte directement dans le fichier source, via l'option **`-i`** (in-place) de la commande **`sed`** (stream editor) :

```
# Attention : la commande suivante est sur une ligne !
sed -i 's/^#NAutoVTs=6/NAutoVTs=6/' /etc/systemd/logind.conf
```

.....

Petite note personnelle : M. Torvalds, le créateur historique du noyau *Linux* en 1991, a jugé bon, en 2020, de retirer le défilement en console (scrolling) des paquets **`vgacon/fbcon`**, jugeant que ce code n'était pas propre, et surtout : ne servait à rien…

On passera la bêtise du propos, en espérant qu'un développeur un jour réécrira cette fonctionnalité si pratique, que l'on utilise au quotidien en mode graphique !

En attendant, on peut aussi utiliser l'outil **`screen`** pour tenter de sauver les meubles. Mais les raccourcis clavier de cet outil demandent un apprentissage certain…

- Fin de la parenthèse -

.....

23. Sortie du conteneur :

```
exit
```

24. Démontage des partitions :

```
umount /mnt/dev /mnt/proc /mnt/sys
umount /mnt
```

25. Retrait du périphérique et de son module noyau :

```
qemu-nbd -d /dev/nbd0
rmmod nbd
```

E) Test de la VM sur l'hôte via KVM

```
kvm --name debian -k fr -m 2048 -hda /home/stagiaire/kvm/debian.qcow2 -boot c -soundhw hda,ac97 -smp cpus=2 -net nic,model=virtio -net user
```

F) Optionnel dans la VM après démarrage

1. Utilisation de **`tasksel`** pour installer rapidement d'autres paquets « par tâches » :

```
tasksel –list-tasks
```

2. Exemple pour une installation « standard » :

```
tasksel install standard
```

3. Gestion des alternatives :

```
update-alternatives --get-selections
update-alternatives --list editor
update-alternatives --config editor
```

G) Votre nouvelle mission (si vous l'acceptez...)

Refaire tout ce TP MAIS avec un schéma UEFI/*GPT* bien sûr !

Dans cette nouvelle version, on ignorera la partition d'échange (swap) considérant que les machines actuelles ont largement assez de mémoire vive en standard, mais il faudra créer la partition *EFI* de 100Mo en fat32 à la place, en tête du conteneur pour rester cohérent, sans oublier le drapeau *efi/esp*...

L'amorçage du système sous *KVM* demandera l'installation du paquet **ovmf** s'il n'est pas déjà présent, et le rajout de l'option :

```
-bios /usr/share/ovmf/OVMF.fd
```

à la commande déjà vue pour démarrer une fois le conteneur prêt.

Attention : il faudra également adapter votre ***/etc/fstab*** pour refléter le nouveau schéma de partitionnement, en vous aidant d'internet…

N'oubliez d'utiliser massivement la commande *GNU/Linux* **`history`**, qui vous permet de repérer des commandes déjà tapées, avec les syntaxes suivantes :

➔ **`!$NUMERO`** : pour rejouer la commande *$NUMERO*.

➔ **`!$DEBUT_DE_COMMANDE`** : pour rejouer la commande qui commençait par *$DEBUT_DE_COMMANDE*.

➔ **`CTRL-R`** : recherche arrière dans l'historique via mots-clés.

➔ **`!!`** :répétition de la dernière commande.

➔ etc