Virtualisation avec qemu/KVM et libvirt

- Objet : Création de machines virtuelles avec libvirt (qemu/kvm) en mode bridge
- Niveau requis :
  débutant, avisé
- Commentaires : La virtu c'est cool, sans interface graphique encore plus...
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  à-placer
    - Création par framend 08/05/2020
    - Testé par <...> le <...>
    - * Commentaires sur le forum : Lien vers le forum concernant ce tuto [1] Fix Me!

Nota :

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

Créer des machines virtuelles c'est super intéressant pour expérimenter sans tout casser.

Voici donc une méthode utilisant l'hyperviseur KVM/qemu. J'ai choisi la techno libvirt/KVM, pour son coté «full-virtualization» apportant, en principe, de meilleures performances.

En effet, lorsqu'un CPU emule un CPU virtuel (vCPU) l'hyperviseur doit «traduire» les instructions de l'un vers l'autre. Ce qui implique un impact conséquent en terme de performances. Pour éviter ceci les technologies du type Intel VT-x et AMD-V permettent un transfert plus direct des instructions. Ce qui signifie que les instructions données au vCPU peuvent-être directement exécutés directement sur une partie du CPU physique.

Considérons donc ici qemu comme l'hyperviseur des machines virtuelles, KVM comme un agent accélérateur d'instructions et libvirt (et tous ses composants), comme un gestionnaire de VM.

Il faut, avant toute installation vérifier que le CPU de la machine hôte supporte la virtualisation.

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

Doit renvoyer autre chose que «0», pour permettre la virtualisation sur ce processeur. Sinon il faut changer de machine (ou de processeur).

Installation

Pour installer le minimum requis de dépendances:

```
apt install --no-install-recommends libvirt-clients libvirt-daemon bridge-
```

```
utils virt-manager
```

Puis vérifier l'état du système libvirtd (le daemon de libvirt):

```
systemctl status libvirtd.service
```

Qui doit renvoyer quelque chose dans le genre de:

systemctl status libvirtd.service

```
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Thu 2020-05-07 13:54:29 CEST; 1 day
1h ago
     Docs: man:libvirtd(8)
           https://libvirt.org
 Main PID: 645 (libvirtd)
    Tasks: 19 (limit: 32768)
   Memory: 32.6M
   CGroup: /system.slice/libvirtd.service
           ├─645 /usr/sbin/libvirtd
           ├─782 /usr/sbin/dnsmasq --conf-
file=/var/lib/libvirt/dnsmasq/default.conf --lease
           └─783 /usr/sbin/dnsmasq --conf-
file=/var/lib/libvirt/dnsmasq/default.conf --lease
```

À partir de là le démon libvirtd est bien lancé.

Création des utilisateurs

Pour pouvoir utiliser des VM via virsh (l'outil en ligne de commande permettant la gestion de libvirt) en user non privilégié (pas root, donc) il va falloir ajouter cet user dans les groupes libvirt. Il n’est pas nécessaire d’ajouter l’utilisateur au groupe libvirt-qemu, seul l’utilisateur libvirt-qemu a besoin d’y être (ce qui est fait à l’installation).

```
adduser <nom_d'user> libvirt
```

Une fois cela réalisé, si vous souhaitez pouvoir créer et gérer des VM en simple user, il est nécessaire d’éditer le fichier /etc/libvirt/libvirtd.conf et de décommenter la ligne suivante:

/etc/libvirt/libvirtd.conf

```
unix_sock_group = "libvirt"
```

Préparation du profil réseau

Il va maintenant s'agir de lancer le profil réseau de libvirt. Pour des VM créées en user non privilégié, il faudra utiliser sudo, sinon passer root via:

```
su -l
```

. Libvirt utilise un profil déja enregistré sous le nom de «default». On va donc le lancer et le passer en auto-start afin d'éviter d'avoir à le faire à chaque utilisation:

```
sudo virsh net-start default
sudo virsh net-autostart default
```

Attention: Avant d'éditer un profil réseau il est impératif de le stopper!

Vous pouvez stopper le profil réseau via:

```
sudo virsh net-stop default
```

La liste des profil réseau existants s'obtient via:

```
sudo virsh net-list --all
```

Ce qui doit renvoyer, pour un service nommé «default» lancé:

virsh net-list --all

```
 Name      State    Autostart   Persistent
----------------------------------------------
 default   active   yes         yes
```

Création du bridge réseau

Pour lier la VM à l'interface réseau de la machine hôte (host) il faut créer un bridge, ici appellé br0.

Plusieurs possiblités, via network-manager ou via le fichier /etc/network/interfaces.

Ici, j'ai éliminé l'usage de network-manager que je n'utilise que sur des machines disposants d'une connection wifi. Un tuto sur son usage est cependant disponible ici:

- https://debian-facile.org/doc:reseau:network-manager

Donc, pour un bridge réseau, éditer le fichier /etc/network/interfaces de la manière suivante, en remplaçant évidemment le nom des interfaces par les votres:

```
vim /etc/network/interfaces
```

```
source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug enp0s7
iface enp0s7 inet manual

# This is an autoconfigured IPv6 interface
iface enp0s7 inet6 manual

# Bridge interface
auto br0
iface br0 inet dhcp
    bridge_ports enp0s7
    bridge_stp off
    bridge_maxwait 0
    bridge_fd 0
```

Ceci fait, un redémarrage du service networking sera de mise:

```
systemctl restart networking.service
```

Une vérification du fait qu'une adresse IP est obtenue par le bridge (et non plus par l'interface physique), via:

```
ip a
```

Doit renvoyer quelque chose de cet ordre:

```
[…]
2: enp0s7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master br0 state UP group default qlen 1000
    link/ether 00:1c:25:02:43:50 brd ff:ff:ff:ff:ff:ff
3: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether 00:1c:25:02:43:50 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.45/24 brd 192.168.0.255 scope global dynamic br0
       valid_lft 769121sec preferred_lft 769121sec
    inet6 2a01:e0a:37f:93a0:21c:25ff:fe02:4350/64 scope global dynamic
mngtmpaddr
       valid_lft 85982sec preferred_lft 85982sec
    inet6 fe80::21c:25ff:fe02:4350/64 scope link
       valid_lft forever preferred_lft forever
```

Sinon relancer une demande d'ip au DHCP via:

```
dhclient
```

Installation de la machine virtuelle

Je préfére mettre les images des machines virtuelles dans un repertoire que je choisis plutôt que l'espace de stockage par defaut alloué par libvirt. Donc il faut les créer:

```
mkdir -p $HOME/Virtu/vm
```

Ou tout autre chemin que vous souhaitez...

Une dernière étape avant l'installation proprement dite de la VM est de mettre un bit setuid sur un script de qemu dans /usr/lib/qemu/qemu-bridge-helper. Il s'agit de bits de contrôle d'accès appliqués aux fichiers et répertoires d'un système d'exploitation UNIX. Grâce à eux, un processus exécutant un tel fichier peut s'exécuter au nom d'un autre utilisateur (wikipedia). En d'autres terme elle permet la création et l'administration de machines virtuelles par un user non-privilégié:

```
chmod u+s /usr/lib/qemu/qemu-bridge-helper
```

Ne reste plus qu'a installer la VM proprement dite. En l'occurence je prends l'exemple d'un guest (la VM) Debian buster auquel j'alloue 50Go d'espcace disque, 1Go de RAM et un seul vCPU. Remplacez évidemment <nom_VM> par le nom que vous souhaitez lui donner.

```
virt-install \
--name <nom_VM> \
--memory 1024 \
--vcpus 1 \
--os-type linux \
--os-variant debian10 \
--graphics none \
--network bridge=br0 \
--extra-args 'console=ttyS0,115200n8 serial' \
--location http://deb.debian.org/debian/dists/stable/main/installer-amd64/ \
--disk ~/Virtu/vm/<nom_VM>.img,device=disk,size=50,format=qcow2
```

Une fois l'installation terminé (je ne détaille pas, c'est une Debian, donc fastoche), le reboot final se fera seul. Et voilà vous avez une belle Debian dans une Debian !

Utilisation de base

Lister les VM's

```
virsh list --all
```

Lancer une VM

```
virsh start --domain <vm_name>
```

Se connecter à la console d'une VM

```
virsh console <vm_name>
```

Suspendre une VM

```
virsh suspend <vm_name>
```

Reprendre après suspension

```
virsh resume <vm_name>
```

Arrêt gracieux de la VM

```
virsh shutdown --domain <vm_name>
```

Arrêt brutal de la VM (si shutdown ne fonctionne pas)

```
virsh destroy --domain <vm_name>
```

Supprimer la VM

```
virsh undefine --domain <vm_name>
```

Les fichiers de stockage des vm ne sont pas supprimés automatiquement, il faut le faire à la main:

```
rm -rf $HOME/Virtu/vm/<nom_VM>.img
```

Documentation

Les commandes man pour ces outils.

```
man virt-install
```

```
man virsh
```

Documentations utilisées pour écrire ce tuto:

- https://wiki.debian.org/KVM#Managing_VMs_from_the_command-line
- https://wiki.debian.org/fr/BridgeNetworkConnections
- https://wiki.libvirt.org/page/Networking#Altering_the_interface_config
- https://wiki.libvirt.org/page/Libvirtd_and_dnsmasq
- https://wiki.qemu.org/Features/HelperNetworking
- https://www.raymii.org/s/articles/virt-install_introduction_and_copy_paste_distro_install_commands.html

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/atelier:chantier:virtualisation-avec-libvirt

Last update: **22/05/2023 21:50**