

# Debian Lenny - Virtualisation avec Libvirt/KVM

Debian GNU/Linux



Matthieu Vogelweith  
13 janvier 2009

# Résumé

L'objectif de ce document est de détailler l'installation d'un serveur KVM [1] (**K**ernel based **V**irtual **M**achine) complet ainsi que les bases de son administration.

Ce document a été rédigé en LaTeX en utilisant l'excellent Vim sous Debian GNU/Linux. Il est disponible aux formats XHTML et PDF. Les sources LaTeX sont disponibles ici : [L<sup>A</sup>T<sub>E</sub>X](#)

# Licence

Copyright ©2009 Matthieu VOGELWEITH <matthieu@vogelweith.com>.

Vous avez le droit de copier, distribuer et/ou modifier ce document selon les termes de la GNU Free Documentation License, Version 1.3 ou ultérieure publiée par la Free Software Foundation ; avec aucune section inaltérable, aucun texte de première page de couverture, et aucun texte de dernière page de couverture. Une copie de la licence est disponible dans la page [GNU Free Documentation License](#).

# Table des matières

<b>Table des matières</b>	<b>3</b>
<b>1 Préparation</b>	<b>4</b>
1.1 Pré requis . . . . .	4
1.2 Installation des paquets . . . . .	4
<b>2 Gestion des images disques</b>	<b>6</b>
2.1 Présentation . . . . .	6
2.2 Création d'une image disque . . . . .	6
2.3 Création d'une image disque relative . . . . .	6
2.4 Conversion d'images . . . . .	6
<b>3 Gestion des snapshots</b>	<b>7</b>
3.1 Mode snapshot . . . . .	7
3.2 Création d'un snapshot . . . . .	7
3.3 Affichage de la liste des snapshots . . . . .	7
3.4 Chargement d'un snapshot . . . . .	7
<b>4 Utilisation de libvirt pour la gestion des VM</b>	<b>8</b>
4.1 Création d'une VM . . . . .	8
4.2 VIRSH : Gestion des VM en console . . . . .	8
4.3 Sauvegarde/Restauration automatique . . . . .	8
<b>5 Serveur de virtualisation : problématique réseau</b>	<b>10</b>
5.1 Présentation . . . . .	10
5.2 Pré requis . . . . .	10
5.3 Configuration réseau de l'hôte . . . . .	11
5.4 Le proxy ARP avec Shorewall . . . . .	11
5.5 Configuration de libvirt . . . . .	12
5.6 Configuration réseau des clients . . . . .	13
<b>6 Références</b>	<b>14</b>

# Chapitre 1

## Préparation

### 1.1 Pré requis

Pour obtenir des performances correcte et pouvoir installer des OS non modifié dans les machines virtuelle, le support des extensions de virtualisation du processeur est indispensable. Il est possible de vérifier l'existence de cette extension sur le processeur à l'aide de la commande suivante :

```
# egrep '^flags.*(vmx|svm)' /proc/cpuinfo >/dev/null && echo OK || echo KO
```

Si la commande retourne "OK", le processeur possède les extensions de virtualisation.

Ce document suppose également que le serveur est déjà installé avec une Debian 5.0 (Lenny) [2] propre. L'installation et la configuration du système de base sont présentée en détail dans un document dédié à cet effet : [3].

### 1.2 Installation des paquets

Comme indiqué précédemment, cette documentation propose d'utiliser la LibVirt pour gérer des machines virtuelles KVM. Dans Lenny, tout s'installe très simplement avec la commande suivante :

```
# aptitude install libvirt-bin kvm qemu virtinst
```

Si les extensions de virtualisation sont bien gérées par le processeur, le script d'init de kvm a dû charger automatiquement les modules noyaux. Il est possible de le vérifier avec la commande suivante :

```
# lsmod | grep kvm
kvm_intel          39776  1
kvm                127464  1 kvm_intel
```

Par défaut, libvirt ne se lance pas automatiquement. Pour ce faire, il suffit d'éditer le fichier /etc/default/libvirt-bin et de modifier la variable suivante :

```
start_libvirtd="yes"
```

Le démon sera alors actif au démarrage ou après avoir exécuté la commande suivante :

```
# /etc/init.d/libvirt-bin restart
```

## Chapitre 2

# Gestion des images disques

### 2.1 Présentation

- différents format disponibles

### 2.2 Création d'une image disque

```
$ qemu-img create -f qcow2 disk.qcow2 100G
```

### 2.3 Création d'une image disque relative

- Création d'une image disque en utilisant une autre image comme base. Cette nouvelle image ne contiendra que les diff par rapport à l'image de base.
- Utile par exemple si on dispose d'une image clean d'un OS et que l'on souhaite décliner plusieurs machines à partir de cet OS.

```
$ qemu-img create -b disk.qcow2 -f qcow2 reldisk.qcow2
```

Il suffit ensuite d'utiliser reldisk.qcow2 dans les nouvelles VM.

### 2.4 Conversion d'images

Conversion d'une image VMWare :

```
$ qemu-img convert -f vmdk vdisk.vmdk -O qcow2 vdisk.qcow2
```

Réduction de la taille d'une image qcow2 :

```
$ qemu-img convert vdisk.qcow2 -O qcow2 vdisk-clean.qcow2
```

## Chapitre 3

# Gestion des snapshots

### 3.1 Mode snapshot

Mode snapshot : disk en RO, toute les modifs sont faites dans des fichiers temp dans /tmp. Possibilité d'écriture sur le disk avec "commit" en console.

### 3.2 Création d'un snapshot

```
qemu> savevm
```

### 3.3 Affichage de la liste des snapshots

```
qemu> info snapshots
```

### 3.4 Chargement d'un snapshot

```
qemu> loadvm
```

## Chapitre 4

# Utilisation de libvirt pour la gestion des VM

### 4.1 Création d'une VM

```
# qemu-img create -f qcow2 lenny.qcow2 100G
# virt-install --ram=1024 --name=lenny \
  --file=/var/lib/libvirt/images/lenny.qcow2 \
  --cdrom=/tmp/debian-LennyBeta2-amd64-netinst.iso \
  --hvm --vnc --noautoconsole --accelerate --network=bridge:br0
```

### 4.2 VIRSH : Gestion des VM en console

### 4.3 Sauvegarde/Restauration automatique

VIRSH apporte également des commandes simples permettant de suspendre restaurer les machines virtuelles. Par exemple, il est possible de mettre une machine en hibernation avec la commande suivante :

```
# virsh save vm_name vm_name.dump
```

La machine sera alors stoppée et les données nécessaires à la restauration de la machine seront stockées dans le dump `vm_name.dump`. Pour restaurer la VM, il suffit ensuite d'exécuter la commande ci-dessous :

```
# virsh restore vm_name.dump
```

Une utilisation courante de ce type de commandes et la suspension/restauration des VMs lors de l'arrêt/démarrage de la machine hôte. En effet, par défaut, les machines virtuelles sont stoppées brutalement lors de l'arrêt du démon `libvirt-bin`. Pour corriger ce problème, le paquet disponible fournit depuis la version 0.4.6-4 un script d'init **libvirt-suspendonreboot** permettant de suspendre les VMs lors de l'arrêt de la machine hôte et de la restaurer lors du démarrage. C'est une solution temporaire mais qui permet d'arrêter correctement les VMs lors d'un reboot de la machine hôte par exemple. Ce script peut être installé en utilisant les commandes ci-dessous :

```
# cp /usr/share/doc/libvirt-bin/examples/libvirt-suspendonreboot /etc/init.d/  
# chmod 755 /etc/init.d/libvirt-suspendonreboot  
# mkdir -p /var/lib/libvirt/autosuspend/  
# update-rc.d libvirt-suspendonreboot defaults 21 19
```

## Chapitre 5

# Serveur de virtualisation : problématique réseau

### 5.1 Présentation

Lors de l'utilisation d'une machine hébergée comme serveur de virtualisation, on peut être amené à rencontrer des problèmes réseaux un peu spécifique. Par machine hébergée, entendez machine qui n'est pas accessible physiquement et ou on ne gère pas les équipements réseaux associés".

Par exemple sur les Dedibox XL [4], les extensions de virtualisations sont disponibles et il est possible d'avoir plusieurs IP publiques routées sur la même machine. Cependant, il n'est pas possible d'affecter simplement les autres IP publiques à des VMs bridées sur le réseau en raison des systèmes de sécurités mis en place sur les switches. En effet, lorsqu'on bridge une VM directement sur l'interface physique de la machine hôte, les paquets sortant de la VM sont émis avec une adresse MAC virtuelle différente de l'adresse MAC de la machine hôte. Ce comportement sera immédiatement détecté par le switch qui coupera le port pour des raisons de sécurité parce qu'il n'est pas censé recevoir des paquets avec des adresses MAC différentes.

Dans ce cas de figure, qui correspond à la plupart des plates-formes d'hébergement, il n'est donc pas possible de brider les machines virtuelles directement sur le réseau physique. Pour contourner le problème, il y a principalement deux solutions :

- Faire du NAT pour "cacher" les machines virtuelles derrière la machine hôte. Dans ce cas les machines virtuelles ont une IP privée mais sont accessible depuis l'extérieur avec leur IP publique ;
- Mettre en place un proxy ARP pour "cacher" uniquement les adresses MAC des machines virtuelles. Dans ce cas les machines virtuelles utilisent directement leurs IP publiques.

La solution la plus souple et la plus élégante est bien entendu la solution du Proxy ARP. Ce paragraphe propose donc de mettre en place un proxy ARP afin de gérer une "DMZ de machines virtuelles".

### 5.2 Pré requis

Ce paragraphe suppose bien évidemment qu'une machine hébergée est disponible avec les extensions de virtualisation et plusieurs IP publiques (au moins 2) sont disponibles, toutes routées vers la machine hôte.

Coté logiciel, on suppose que shorewall [5] est déjà installé et correctement configuré sur la machine hôte.

### 5.3 Configuration réseau de l'hôte

La toute première étape dans la mise en place de cette DMZ virtuelle est la création d'un bridge sur la machine hôte qui sera le point d'entrée de la DMZ. Pour faire le parallèle avec une installation physique, on peut comparer ce bridge à un switch sur lequel serait branché une interface du firewall et une interface de chaque machine virtuelle.

La création de ce bridge doit être faite en même temps que le montage de l'interface réseau publique de la machine hôte. Pour cela, modifier la configuration réseau de l'hôte dans le fichier `/etc/network/interfaces` comme indiqué ci-dessous :

```
auto eth0
iface eth0 inet static
    address IP_HOTE
    netmask MASK_HOTE
    gateway IP_GATEWAY
    up brctl addbr dmz0
    up brctl setfd dmz0 0
    up brctl stp dmz0 on
    up ifconfig dmz0 up
```

La configuration ci-dessus permet donc d'ajouter une interface **dmz0** de type bridge pour accueillir les interfaces virtuelles des VMs. Notons que libvirt est capable de gérer ce type de bridge de manière automatique mais dans le cas présent il est indispensable de la gérer manuellement pour les raisons suivantes :

- Le bridge doit être nommé de la même façon parce que son nom est utilisé dans la configuration de shorewall ;
- Le bridge doit impérativement être monté AVANT le démarrage de Shorewall pour que celui-ci démarre correctement et puisse mettre en place le proxy ARP.

**Attention**, il ne faut surtout pas faire l'erreur d'ajouter `eth0` dans les interfaces associées au bridge `dmz0`, dans ce cas les VMs seraient directement bridgées sur le réseau physique et le switch bloquerait le port. Libvirt ajoutera dynamiquement les interfaces virtuelles (`vnet*`) au bridge lors du démarrage des VMs.

### 5.4 Le proxy ARP avec Shorewall

Shorewall permet de configurer un proxy ARP de manière relativement simple. Pour des besoins plus génériques, la documentation officielle [6] est très bien faite.

La première étape dans la mise en place de cette DMZ virtuelle est la création d'une nouvelle zone Shorewall de type IPv4. Cette zone permettra de gérer les règles d'accès à toutes les machines virtuelles présentes dans la DMZ. Ceci se fait en ajoutant la ligne suivante dans `/etc/shorewall/zones` :

```
# ZONE TYPE
dmz     ipv4
```

Il faut ensuite associer cette nouvelle zone à une interface disponible sur le machine hôte. Dans le cas présent, il s'agit du bridge qui a été créé précédemment dmz0. Pour cela, ajouter la ligne suivante dans le fichier /etc/shorewall/interfaces :

```
# ZONE  INTERFACE
dmz     dmz0
```

L'étape suivante est la gestion des règles de filtrage pour cette zone. Si l'on considère que cette zone est réellement une DMZ, on peut ajouter les lignes suivantes dans /etc/shorewall/policy pour autoriser tout le trafic entrant et sortant de la DMZ :

```
# SOURCE  DEST  POLICY
dmz       all   ACCEPT
all       dmz   ACCEPT
```

**Attention**, la configuration décrite ci-dessus autorise tout le trafic vers les machines de la DMZ. Ces machines ne seront donc pas du tout protégées par le firewall de la machine hôte et il est indispensable d'installer un firewall sur chaque machine virtuelle. Néanmoins, il est tout à fait possible, pour des raisons de performance notamment, de modifier ces règles et de gérer toute la sécurité réseau sur la machine hôte.

La zone DMZ étant définie et configurée, il reste à mettre en place le proxy ARP proprement dit, tâche qui est grandement simplifiée par shorewall. Par exemple, si l'on souhaite démarrer deux VM dans la DMZ ayant pour IP publique IP\_DMZ\_1 et IP\_DMZ\_2, il faut ajouter les lignes suivantes dans le fichier /etc/shorewall/proxyarp :

```
# ADDRESS  INTERFACE  EXTERNAL  HAVEROUTE  PERSISTENT
IP_DMZ_1   dmz0       eth0       no          yes
IP_DMZ_2   dmz0       eth0       no          yes
```

Notons que dans la configuration ci-dessus, l'option HAVEROUTE a été définie à "no" pour indiquer à shorewall que les routes nécessaires pour joindre les VMs ne sont renseignées sur la machine hôte. Shorewall ajoutera donc les routes automatiquement.

Enfin, pour permettre à shorewall de transmettre les paquets entre eth0 et dmz0, il faut activer le forwarding en modifiant l'option suivante dans /etc/shorewall/shorewall.conf :

```
IP_FORWARDING=On
```

Pour que les modifications soient prises en compte, il faut bien entendu redémarrer shorewall. **ATTENTION** : pour que shorewall redémarre correctement, le bridge dmz0 doit déjà être actif ! Lorsque tout est vérifié, le redémarrage peut se faire avec la commande ci-dessous :

```
# /etc/init.d/shorewall restart
```

## 5.5 Configuration de libvirt

Lorsque le proxy ARP est correctement configuré, il faut configurer les machines virtuelles pour qu'elles puissent utiliser cette nouvelle configuration. Cette configuration est extrêmement simple

puisqu'il suffit de configurer les VM en mode bridge sur l'interface dmz0. Comme les paquets seront directement envoyés par shorewall sur dmz0, les machines pourront recevoir les paquets sur leur interface réseau.

Cette configuration en mode bridge peut être réalisée avec les différentes interfaces de libvirt (virsh, virt-manager, ...) ou tout simplement en éditant le fichier XML associé à la machine dans /etc/libvirt/qemu/nom\_de\_la\_vm.xml :

```
<interface type='bridge'>
  <mac address='xx:xx:xx:xx:xx:xx' />
  <source bridge='dmz0' />
</interface>
```

## 5.6 Configuration réseau des clients

La machine hôte étant maintenant correctement configurée, il reste à configurer les machines virtuelles pour qu'elles puissent utiliser directement leurs IP publiques. Ceci se fait simplement en utilisant la configuration ci-dessous dans /etc/network/interfaces :

```
auto eth0
iface eth0 inet static
    address IP_DMZ
    netmask 255.255.255.255
    post-up /sbin/ip route add IP_HOTE/32 dev eth0
    post-up /sbin/ip route add default via IP_HOTE
```

Si tout c'est passé correctement, les machines virtuelles bridgées sur dmz0, donc situées dans la DMZ, doit être accessible directement en utilisant leur IP publique.

## Chapitre 6

# Références

- [1] Site officiel de kvm. [kvm.qumranet.com](http://kvm.qumranet.com).
- [2] Site officiel du projet debian. [www.debian.org](http://www.debian.org).
- [3] Installation et configuration de debian etch. [www.vogelweith.com/debian\\_server/00\\_installation.php](http://www.vogelweith.com/debian_server/00_installation.php).
- [4] Le site dedibox. [www.dedibox.fr](http://www.dedibox.fr).
- [5] Shoreline firewall. [www.shorewall.net](http://www.shorewall.net).
- [6] Configuration d'un proxyarp avec shorewall. [www.shorewall.net/ProxyARP.htm](http://www.shorewall.net/ProxyARP.htm).
- [7] Utilisation de virtio avec kvm. [kvm.qumranet.com/kvmwiki/Virtio](http://kvm.qumranet.com/kvmwiki/Virtio).
- [8] Site officiel de libvirt. [libvirt.org](http://libvirt.org).
- [9] Éléments de sécurisation de debian lenny. [www.vogelweith.com/debian\\_server/01\\_security.php](http://www.vogelweith.com/debian_server/01_security.php).