Virtualisation

Professeur : Florent Gluck

March 7, 2023

Introduction à QEMU

Introduction

Le but de ce travail pratique est de prendre en main QEMU, de comprendre son fonctionnement, de l'utiliser dans diverses situations et de se familiariser avec les images au format qcow2.

Contrairement à VirtualBox, vous allez ici uniquement utiliser la ligne de commande. Afin de plus facilement gérer vos VMs et leurs configurations, je vous conseille fortement de réaliser des scripts pour chaque partie ou encore un `Makefile` (avec des cibles pour chaque partie).

0. Préparation

En permier lieu, installez QEMU via le package Debian `qemu-system-x86`. Les autres distributions Linux devraient avoir un package au nom similaire. Le binaire QEMU pour les architectures Intel et AMD est '`qemu-system-x86_64`.

Vous utiliserez aussi KVM donc assurez-vous que vous avez bien le module `kvm` chargé dans votre noyau Linux et que votre utilisateur peut y accéder. Lisez les slides du cours QEMU, "Can a host run KVM?" et "Using KVM" afin de vérifier que votre système est configuré correctement.

1. Prise en main de QEMU

Le but ici est d'installer une première VM avec QEMU depuis un live CD. Partez depuis l'image ISO Xubuntu 22.04 déjà téléchargée pendant le premier lab sur Virtualbox.

Partie 1

Exécutez une VM avec `qemu-system-x86_64` et l'image ISO ci-dessus. N'installez pas Xubuntu dans la VM guest pour le moment. Cette VM doit posséder un lecteur CD-ROM/DVD-ROM contenant l'image ISO ci-dessus, 4GB de RAM, et 2 CPUs. N'utilisez pas d'options avancées de QEMU, seulement les arguments de base listés ci-dessous :

- `-cdrom x` spécifie que le contenu du lecteur CD-ROM/DVD-ROM est le fichier `x`
- `-m x` spécifie `x` MB de RAM
- `-smp cpus=1` spécifie 1 CPU

Pour plus d'information sur la syntaxe de QEMU, exécutez `man qemu-system`.

Important : n'exécutez **jamais** QEMU en tant que `root` !. En plus d'être inutile, c'est dangereux car si le processus est compromis, alors il aura un accès complet à toute la machine. De manière générale, assurez-vous de ne jamais exécuter de commandes en tant que `root` (ou `sudo`), à moins que cela soit absolument nécessaire.

- Combien de temps s'est écoulé jusqu'à arriver au choix de la langue dans le processus d'installation de Xubuntu ? Vous pouvez déterminer le temps en exécutant la commande `time` et en inspectant la valeur `real`. Par exemple, `time qemu-system-x86_64 ...` puis en pressant CTRL-C dans le terminal une fois le moment attendu arrivé).

Partie 2

Réalisez la même mesure de temps qu'au point précédent, mais cette fois-ci ajoutez les arguments `-enable-kvm` et `-cpu host` à QEMU.

- Quel est le nouveau temps obtenu ?
- Pourquoi l'exécution est-elle beaucoup plus rapide qu'au point précédent ?
- Plus précisément :
  - dans le cas présent (avec `-enable-kvm`), quel type de technique de virtualisation est utilisé par QEMU ?
  - dans le cas précédent (sans `-enable-kvm`), quel type de technique de virtualisation est utilisé par QEMU ?

Commencez l'installation de Xubuntu dans votre VM.

- Est-il possible d'installer Xubuntu ?

Partie 3

A l'aide de l'outil `qemu-img`, créez les deux images disque suivantes :

- une image `disk1.qcow` de 500GB au format qcow2
- une image `disk2.raw` de 500MB au format raw

Inspectez les deux fichiers images que vous venez de créer (l'outil `hexedit` permet d'inspecter et éditer le contenu de fichiers binaires).

- En quoi l'image `qcow2` diffère de l'image `raw` ? Décrivez en les différences majeures.
- Inspectez la taille apparente (via `ls -l`) de chaque image ainsi que la taille réelle. Que pouvez-vous en conclure ?
- Quel est le contenu de `disk2.raw` ?

Partie 4

Maintenant que vous avez créé deux disques, configurez votre VM pour qu'elle utilise `disk1.qcow` comme premier disque et `disk2.raw` comme deuxième disque. Un moyen simple pour spécifier le premier disque du contrôleur IDE est avec `-hda`, le deuxième avec `-hdb`, etc. Ceci dit, il est préférable d'utiliser l'argument `-drive` car il permet un contrôle plus fin, notamment en permettant de préciser le type d'image disque (nécessaire pour le format `raw`), le type de media, l'indice du disque, etc. Utilisez donc `-drive` pour spécifier les deux disques ci-dessus, en précisant pour chacun le format de l'image.

Lisez les slides de cours ou le manuel de QEMU (`man qemu-system`) pour déterminer et comprendre la syntaxe à utiliser.

- Quel est la différence entre les arguments `-hdx` (où `x` représente le numéro de disque `a`, `b`, etc.) et `-drive` ?
- Comment peut-on écrire **exactement** l'équivalent de `-hdb` avec la syntaxe `-drive` ?

Démarrez une VM avec la configuration décrite ci-dessus pour les deux disques et installez Xubuntu sur le premier disque. Pour cela, au début de l'installation choisissez de faire un partionnement

manuel des disques. Créez alors une première partition primaire de 1MB de type “Reserved BIOS boot area”. Créez ensuite une deuxième partition primaire de tout l'espace disque restant et créez-y un système de fichiers ext4 dans lequel sera monté la racine du système de fichiers (`/`). Pour le deuxième disque, créez une partition primaire de tout l'espace disque et créez-y un système de fichiers ext4 dans lequel sera monté `/boot`.

Une fois l'installation terminée, rebootez le système et connectez-vous pour vérifier que tout fonctionne correctement (au cas où vous rencontreriez une erreur liée à l'installation de la langue, ignorez la).

Dans l'OS guest, inspectez les périphériques de votre VM avec la commande `lspci`.

- Localisez l'interface réseau. Selon vous, s'agit-t-il d'une interface réseau emulée ou paravirtualisée ?
- Avec VirtualBox la configuration de la VM se trouvait dans un fichier XML. Où se trouve la configuration de la VM dans le cas de QEMU ?
- Finalement, parvenez vous à réaliser un copier/coller de texte entre la VM et votre machine hôte ?

2. Conversion d'images disque

Le but ici est de manipuler la conversion d'images disque et de configurer les drivers de QEMU afin d'en améliorer les performances.

Partie 1

On désire reprendre la VM `hepiadoom` utilisée dans le travail pratique précédent sur VirtualBox.

- Pouvez-vous exécuter directement la VM `hepiadoom` avec QEMU ?

Convertissez le disque VirtualBox de la VM `hepiadoom` en une image au format qcow2, en vous assurant de spécifier une taille de cluster de 2MB (au lieu de la taille par défaut de 64KB).

Exécutez une nouvelle VM en utilisant le disque converti au point précédent et mesurez le temps mis pour parvenir à l'écran de connexion.

- Quel temps avez-vous mesuré ?

Connectez-vous et déterminez quel est le nom du périphérique contenant le système de fichiers monté à la racine (`/`) du système. Pour rappel, sur Linux (et autres UNIXes) les périphériques se trouvent dans le répertoire `/dev`. Aussi, la commande `mount` affiche tous les systèmes de fichiers montés (dont celui monté à la racine). Alternativement, la commande `df -h` liste les systèmes de fichiers montés.

- De quel périphérique s'agit-il ?

Partie 2

On désire améliorer les performances disque de la VM en utilisant des drivers paravirtualisés pour le contrôleur de disque dur. Exécutez la même VM qu'au point précédent, mais ajoutez l'option `if=virtio` à l'argument `-drive`, ce qui a pour effet d'utiliser un contrôleur de disque `virtio` paravirtualisé.

Comme avant, mesurez le temps mis pour parvenir à l'écran de connexion.

- Quel gain de temps avez-vous obtenu par rapport au contrôleur de disque émulé ?

Vous remarquerez qu'avec ce nouveau contrôleur, le nom du périphérique a changé. Il s'appelle `/dev/vda` où le `v` signifie probablement *virtual*.

- Que voyez-vous également comme contrôleur de stockage si vous listez les périphériques PCI avec la commande `lspci` ?

3. Inspection d'images

A l'aide de `wget` ou `curl`, téléchargez le fichier `vm2fix.vmdk` se trouvant à cette URL. Convertissez ensuite cette image au format `qcow2` à l'aide de `qemu-img`.

Cette image contient un OS guest Linux, mais vous ne connaissez pas les credentials de l'utilisateur. Pour réussir à vous y connecter, il vous faut donc *hacker* cette image de disque comme au labo précédent sur VirtualBox. Le but ici est d'accéder au système de fichiers de l'image disque en lecture/écriture afin de modifier le contenu du fichier `/etc/shadow` ou `/etc/passwd` pour y supprimer le mot de passe de l'utilisateur.

En tant qu'hackeur en herbe, vous devez déterminer les partitions que comporte l'image de disque à hacker. C'est exactement le but du programme `virt-filesystems`. Relisez le cours qui explique l'inspection/modification de fichiers image.

- Combien de partitions comporte l'image `vm2fix.qcow` ?

Ensuite, vous devez déterminer ce que contient chaque partition. Pour cela, l'outil `guestmount` est d'une aide précieuse : il permet de monter une partition de l'image disque dans le système de fichiers de la machine hôte. Montez donc chaque partition pour en inspecter le contenu, ce qui vous permettra de trouver quel(s) fichier(s) modifier pour réussir à vous logger correctement dans la VM. N'oubliez pas de démonter les partitions montées une fois que vous aurez terminé (avec `guestunmount` ou `umount`).

Vous aurez la confirmation d'avoir réussi à *hacker* la VM lorsque vous aurez abouti à un shell :-)

Remarque

Dans le cas d'une image disque *raw* (brute -image exacte d'un disque physique), il est possible de la monter avec la commande `mount` et le périphérique "loopback", comme illustré dans l'exemple ci-dessous :

```
mount disk.raw -o loop,offset=1048576 dir
```

L'argument `offset` (en bytes) indique où se situe le début du système de fichiers dans l'image. Cet offset peut être déterminé en listant les partitions du disque avec `fdisk -l disk.raw` (attention : l'unité affichée par `fdisk` pour les valeurs de début et fin de partition est le secteur).

4. QEMU Guest Agent

Le but de cette partie est de vous familiariser avec le QEMU Guest Agent (QGA).

Re-démarrez la VM précédente mais avec les options nécessaires à l'utilisation du QEMU Guest Agent et vérifiez que celui-ci s'exécute dans la VM.

Comme le QEMU Guest Agent renvoie du JSON, il est recommandé d'installer et utiliser un formateur de JSON comme `aeson-pretty`. Celui-ci lit du JSON sur l'entrée standard (stdin) et le formate joliment sur la sortie standard (stdout).

En guise de premier essai, exécutez la commande QGA `guest-info` afin de déterminer quelles sont les commandes supportées. Ecrivez ensuite un programme dans le script/langage de votre choix qui réalise les opérations suivantes :

1. Récupère le premier utilisateur du système

2. Créé un fichier “busted” dans le répertoire de l’utilisateur précédent, contenant un message de votre choix
3. Eteint gracieusement la VM

5. Virtualisation de *desktop* (VDI)

Jusqu’à maintenant, l’hyperviseur et l’utilisateur de la VM utilisent et interagissent avec la même machine physique. On s’intéresse ici à découpler l’exécution de la VM des périphériques nécessaires à son interaction, c’est à dire l’affichage du *desktop*, la clavier, la souris et l’audio. Ainsi, la VM s’exécutera sur un serveur distant et votre PC (local) sera utilisé comme système d’interaction uniquement (affichage, clavier, etc.). La communication entre hyperviseur/VM et périphériques d’interaction (affichage, etc.) utilisera le protocole Spice au-dessus de tcp/ip.

Machine distante

A partir de maintenant, vous réaliserez la suite du labo sur un hyperviseur se trouvant sur une machine distante localisée au sein de l’infrastructure d’HEPIA. Dans le fichier `students_machines.txt` disponible sur le git du cours, vous pouvez trouver l’ip de la machine qui vous est attribuée ainsi que les crédentials pour vous y connecter. A noter que votre utilisateur possède un accès `root` via `sudo`.

En premier lieu, changez le mot de passe de votre utilisateur (`passwd`) afin que personne d’autre ne puisse accéder à votre machine.

Ensuite, mettez en place une paire de clés ssh pour vous connecter à votre machine sans avoir à entrer de mot de passe. La clé privée doit résider sur la machine source et la clé publique sur la machine distante. Un moyen simple pour mettre en place vos clés est d’utiliser la commande `ssh-keygen` qui génère une paire de clés ssh. Une clé RSA de 2048 bits fait très bien l’affaire. La commande `ssh-copy-id` copie une clé publique sur une machine de destination (celle-ci est copiée dans le répertoire `.ssh/` de l’utilisateur spécifié). Pensez à copier votre clé publique pour les utilisateurs `student` et `root`.

Enfin, il est possible de configurer, via le fichier `~/.ssh/config`, les crédentials que le client ssh utilise lors de l’établissement d’une connexion. Voici un exemple de configuration où `id_rsa_virt` dénote la clé privée (la clé publique porte l’extension `.pub`) :

```
Host some_host_name 192.168.100.100
        User student
        Port 22
        IdentityFile ~/.ssh/id_rsa_virt
```

Aussi, ssh permet de se connecter à une machine distante en passant par d’autres machines intermédiaires. Cela s’appelle un saut ssh (*jump* ou *hop*). Voici un exemple qui permet de se connecter sur la machine distante 192.168.100.10 (nommée “hyperion”) en faisant un saut sur une machine intermédiaire 192.168.100.20 (nommée “endimyon”) :

```
Host endimyon
    User dave
    HostName 192.168.100.20

Host hyperion
    User student
    HostName 192.168.100.10
    ProxyJump endimyon
```

Ainsi, en tapant simplement `ssh hyperion`, on peut se connecter à la machine distante.

Pour info, lors d’une connexion à une machine distante via ssh, il est possible de se faire déconnecter après un certain temps d’inactivité. Une solution à ce problème est d’utiliser un outil comme

`screen` (disponible dans toutes les distributions Linux), qui s'assure de maintenir la connexion active. Utilisation :

```
screen ssh <hostname/ip>
```

Vous utiliserez votre machine distante à partir de la Partie 2 ci-dessous.

Partie 1

Sur votre machine locale, basez-vous sur la VM `hepiadoom` précédente et assurez-vous qu'elle présente un contrôleur de disque paravirtualisé. Démarrez la et observez le type de carte graphique via la commande `lspci`.

- Quelle est le nom de la carte graphique installée ? Pensez-vous qu'il s'agit d'une carte graphique émulée ou paravirtualisée ?

Changez la config de la VM en précisant le contrôleur graphique paravirtualisé `virtio`, puis redémarrez votre VM. Pour info, l'argument `-vga` permet de spécifier le type de contrôleur graphique que QEMU doit utiliser.

- Quelle est le nom de la carte graphique installée ? Pensez-vous qu'il s'agit d'une carte graphique émulée ou paravirtualisée ?
- En inspectant les modules noyau, en voyez-vous liés à l'affichage paravirtualisé ?

Partie 2

Vous allez maintenant vous familiariser avec la virtualisation de *desktop* grâce à votre machine distante et au protocole Spice.

Copiez l'image disque de votre VM `hepiadoom` et basez vous sur les slides de cours pour ajouter un serveur Spice à votre VM. Il est impératif que cette VM s'exécute sur votre machine distante ! On désire également améliorer les performances réseau de la VM et on désire aussi du son ce qui implique l'ajout d'une carte audio.

Voici les arguments pour ajouter une interface réseau paravirtualisée :

```
-nic user,model=virtio-net-pci
```

Voici les arguments pour ajouter une carte son Intel High Definition :

```
-device intel-hda
```

Démarrez votre nouvelle VM distante intégrant un serveur Spice, une interface réseau paravirtualisée et une carte audio sur l'image disque `hepiadoom`.

Partie 3

Sur votre machine locale, utilisez le client `remote-viewer` ou `spicy` pour vous connecter à votre VM distante `hepiadoom`.

Exécutez `chocolate-doom` afin d'observer les performance de l'affichage distant. Normalement le jeu devrait être relativement jouable (fortement dépendant du trafic réseau). Dans le cas où votre machine locale se trouverait à l'extérieur du réseau HES, vous devriez pouvoir y accéder via le VPN de l'école.

- Si vous fermez la fenêtre du client Spice durant l'utilisation de la VM, est-ce que la VM termine son exécution ? Que se passe-t-il exactement ?
- Parvenez vous à réaliser un copier/coller de texte entre la VM et votre machine hôte ?
- Le jeu Doom est-il jouable ?

Partie 4

Vous allez ici expérimenter avec le *monitor* de QEMU.

Exécutez à nouveau votre VM `hepiadoom` mais en ajoutant un *monitor* accessible via le protocole telnet.

Vérifiez que vous avez accès au monitor QEMU et parcourez les différentes commandes disponibles dans le *monitor* avec la commande `help`.

- Utilisez alors le *monitor* pour inspecter diverses informations liées à votre VM, comme par exemple :
  - que est l'état de votre VM ?
  - combien de CPUs comporte-elle ?
  - KVM est-il actif ?
  - afficher les informations liées à Spice
- Comment pouvez-vous, 1) éteindre (brutalement) votre VM et, 2) faire un *reset* (forcé) de de cell-ci sans accéder à celle-ci via Spice ? Testez pratiquement chacune de vos réponse.

6. Snapshots

Vous allez ici explorer les possibilités offertes par QEMU en matière de snapshots via l'utilisation de `qemu-img`. Lisez la syntaxe des snapshots avec `man qemu-img` ou `qemu-img --help`.

Tout comme pour la partie précédente, utilisez l'hyperviseur sur votre machine distante.

Partie 1

En premier lieu, vous allez manipuler les snapshots externes. Commencez par faire une copie de l'image du disque de votre VM `hepiadoom`. Nommez cette copie `hepiadoom-base.qcow`. Cette image sera l'image *backed* sur laquelle se baseront les snapshots (overlays).

Démarrez une VM utilisant le disque `hepiadoom-base.qcow`. Dans celle-ci, créez le script `createfile` qui prend en argument un nom de fichier et y créé un fichier de 100MB. Pour rappel, `dd` permet de créer un fichier à partir d'une source : `dd if=source_file of=dest_file bs=block_size count=block_count`. Dans votre script, utilisez `dd` avec la source `/dev/urandom` pour remplir les 100MB du fichier.

Soit la séquence de snapshots illustrée en Figure 1 où l'ordre des opérations réalisées est indiqué par les disques numérotés. Le script `createfile` précédent a été utilisé pour ajouter les fichiers `fileA, fileB, fileC`.

On part du principe que les 6 étapes illustrées en Figure 1 ont été réalisées. Vous exécutez une VM utilisant l'image `snap3`.

- Quels fichiers `fileX` seront présents dans cette VM ?

Reproduisez le scénario de la Figure 1 afin de confirmer votre réponse.

Utilisez `qemu-img` pour afficher la chaîne d'images ayant mené à `snap3` et vérifiez qu'il s'agit de la même séquence qu'en Figure 1. Réalisez pareil pour `snap4`.

- Est-il possible de réaliser un snapshot de disque d'une VM en cours d'utilisation ? Confirmez votre hypothèse de manière empirique.

Exécutez une première VM avec le disque `snap3.qcow`. Exécutez ensuite une deuxième VM avec le disque `snap1.qcow`.

- Que va-t-il se passer à votre avis ? Vérifiez votre réponse empiriquement et déduisez-en ce que réalise QEMU.

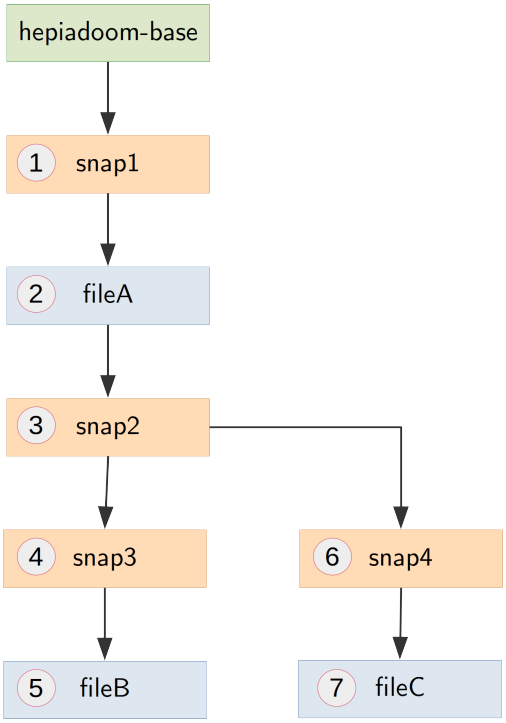

Figure 1: L'image *backed* est en vert, les snapshots (overlays) en orange et les fichiers ajoutés en bleu. L'ordre des opérations réalisées est indiqué par les disques numérotés.

Réalisez un *merge* de type "commit" de l'image `snap4` dans l'image `snap1`.

- Quels fichiers `fileX` seront présents dans `snap1` ? Confirmez votre hypothèse empiriquement.

Comme indiqué dans la documentation de QEMU, il est alors prudent de supprimer les images intermédaires qui pourraient être incohérentes.

- Quelle(s) image(s) pourrait/pourraient être incohérente(s) ?

Au début de la partie 1, vous aviez réalisé une copie de l'image `hepiadoom.qcow` dans `hepiadoom-base.qcow` afin de pas modifier la première image.

- Qu'auriez-vous pu faire pour éviter de réaliser cette copie de plusieurs GB tout en évitant de modifier le fichier d'origine (`hepiadoom.qcow`) ?

Partie 2

Lancez maintenant une partie du jeu Doom en utilisant p.ex. l'image utilisée auparavant (`snap1`), puis en cours de jeu, réalisez un snapshot de VM.

- Pourquoi est-ce que la création de ce snapshot prend significativement plus de temps à se terminer qu'un snapshot de disque ?

Jouez un petit peu, puis réalisez un nouveau snapshot de VM. Listez ensuite les snapshots depuis le monitor.

- Voyez-vous les snapshots réalisés ?

Chargez chaque snapshot de VM afin de vérifier que vous obtenez bien le comportement attendu.

Éteignez votre VM.

- Comment pouvez-vous listez les snapshots de VM que vous venez d'effectuer sans utiliser le QEMU *monitor* ?

Exécutez à nouveau votre VM, mais en indiquant à QEMU de charger le dernier snapshot réalisé depuis la ligne de commande avec l'argument `-loadvm`.

Éteignez votre VM, puis exécutez en une nouvelle avec la même image disque, mais avec des périphériques différents.

- Chargez ensuite un des snapshots effectués précédemment. Que remarquez-vous ?

7. Emulation

Jusqu'à présent vous avez utilisé `qemu-system-x86_64` mais maintenant vous désirez exécuter une VM pour une architecture complètement différente, à savoir ARM et plus exactement une RaspberryPi 3.

Suivez les instructions de ce tutoriel qui détaillent clairement les étapes à effectuer pour démarrer l'image Raspbian Stretch Lite avec `qemu-system-arm` : https://github.com/wimvanderbauwhede/limited-systems/wiki/Raspbian-%22stretch%22-for-Raspberry-Pi-3-on-QEMU

- Quel est le type de l'image disque Raspbian (indice : commande `file`) ?
- Quelle est la raison pour laquelle on passe le kernel en paramètre à QEMU ?
- Peut-on ajouter le paramètre `-enable-kvm` à QEMU ? Est-ce que cela a un intérêt ou pas ?