

Virtualisation des systèmes

Florent Glück

March 09, 2025

Introduction à QEMU

Objectif

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. Assurez-vous d'utiliser un système natif GNU Linux, sinon vous ne pourrez pas réaliser toutes les étapes du labo et observer les résultats attendus.

A défaut d'utiliser une interface graphique, toutes les opérations ici seront uniquement réalisées via la ligne de commande. Afin de plus facilement gérer vos VMs et leurs configurations, vous terminerez par développer un script bash permettant de créer une VM avec des arguments configurables selon le scénario visé.

(0) Préparation

En premier 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 de cours sur QEMU traitant de 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.

Depuis le site <https://lubuntu.me/downloads/>, téléchargez l'image ISO de la distribution lubuntu Desktop 24.04.2 LTS (Noble Numbat) pour l'architecture AMD64. Vérifiez que l'image ISO téléchargée possède le bon checksum en suivant les instructions décrite ici <https://help.ubuntu.com/community/UbuntuHashes>.

Partie 1

Exécutez une VM avec `qemu-system-x86_64` et l'image ISO téléchargée. N'installez pas lubuntu dans la VM guest pour le moment. Cette VM doit posséder un lecteur CD-ROM/DVD-ROM contenant l'image ISO, 4GB de RAM, et 2 CPUs. N'utilisez pas d'options avancées de QEMU pour l'instant, 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=n` spécifie `n` CPU

Pour plus d'information sur la syntaxe de QEMU, exécutez `man qemu-system-x86_64`.

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 **réellement** nécessaire.

- Combien de temps s'est écoulé jusqu'à arriver dans le bureau (*desktop*) de lubuntu ? 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 passez l'argument `-machine accel=kvm` à QEMU.

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

Commencez l'installation de lubuntu dans votre VM.

- Est-ce que l'installation est possible ? Dans la négative, expliquez pourquoi.

Partie 3

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

- Une image `lubuntu.qcow` de 100GB au format `qcow2`
- Une image `disk.raw` de 100GB au format `raw`

Inspectez les deux fichiers image que vous venez de créer (utilisez `hexedit` pour inspecter ou éditer le contenu de fichiers binaires).

- En quoi l'image `qcow2` diffère-t-elle de l'image `raw` ? Décrivez-en les différences majeures.
- Inspectez la taille apparente (`ls -h`) de chaque image ainsi que la taille réelle (`ls -s`). Que constatez-vous ? Expliquez la raison des différences observées.
- Quel est le contenu de `disk.raw` ?

Partie 4

Créez maintenant une VM pour qu'elle utilise `lubuntu.qcow` comme premier disque. Un moyen simple pour spécifier le premier disque du contrôleur de disque est avec `-hda`, le deuxième avec `-hdb`, etc. Ceci dit, il est préférable d'utiliser l'argument `-drive` car il permet un meilleur contrôle, 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 le premier disque ci-dessus, en précisant le format de l'image.

Lisez les slides de cours ou le manuel de QEMU (`man qemu-system-x86_64`) pour déterminer et comprendre la syntaxe à utiliser.

- Comment peut-on écrire **exactement** l'équivalent de `-hdb` avec la syntaxe `-drive` ?

Installez alors `ubuntu` sur l'image disque `ubuntu.qcow`. Au moment de la sélection du support de stockage, sélectionnez "Effacer le disque" ("*Erase disk*"). Un *swap* file n'est pas nécessaire.

- Une fois l'installation terminée, inspectez sur la machine hôte la taille réelle de l'image disque sur laquelle vous venez d'installer `ubuntu`. Quelle est sa taille ?

Rebootez votre VM et connectez-vous pour vérifier que tout fonctionne correctement.

Dans l'OS guest, inspectez le ou les CPUs détectés avec la commande `lscpu`.

- Inspectez le nombre de CPU dans l'OS guest ainsi que le modèle de CPU. Quelles différences remarquez-vous avec ce que vous observez pour la machine hôte ? Pouvez-vous expliquer ces différences ?
- Inspectez la section "Virtualization features". Quelle différence remarquez-vous avec ce que vous observez pour la machine hôte ? Quelle est la raison de cette différence ?
- D'après ce que vous observez, est-ce que l'OS guest peut être utilisé comme hyperviseur en utilisant la technique "hardware-assisted virtualization" ? Justifiez votre réponse. Dans la négative, recherchez un moyen pour y parvenir malgré tout (cf. cours). Expliquez ensuite comment vous y êtes parvenu et pourquoi cela fonctionne.

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

- Localisez l'interface réseau. Selon vous, s'agit-il d'une interface réseau émulée ou paravirtualisée ?
- Où se trouve la configuration de la VM et comment est-elle définie ?

(2) Optimisation de performances

Le but ici est d'optimiser les performances de votre VM en remplaçant le contrôleur de disque émulé (*full virtualization*) par un contrôleur de disque paravirtualisé.

Redémarrez votre VM fraîchement créée en utilisant la même configuration que précédemment et mesurez le temps écoulé entre le démarrage de la VM jusqu'à sa terminaison après un shutdown depuis l'interface graphique (il s'agit donc d'un shutdown "gracieux" ici). Mesurez le temps comme à l'exercice précédent avec la commande `time`.

- 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. Enfin, un autre moyen de lister les systèmes de fichiers montés est en inspectant le fichier virtuel `/proc/mounts`.

- De quel périphérique s'agit-il ?

On désire améliorer les performances disque de la VM en utilisant un contrôleur de disque paravirtualisé. Pour cela, ajoutez l'option `if=virtio` à l'argument `-drive`, ce qui a pour effet d'utiliser un contrôleur de disque paravirtualisé `virtio`.

Comme avant, mesurez le temps écoulé pour arriver à l'écran de connexion.

- Quel est le nouveau temps obtenu ? Quel est le facteur de gain obtenu par rapport à l'utilisation d'un contrôleur de disque émulé ?

Tout comme pour la partie précédente, déterminez quel est le nom du périphérique contenant le système de fichiers monté à la racine (`/`) du système.

- De quel périphérique s'agit-il ?

Afin d'investiguer un peu plus ce que signifie ce nom de périphérique, ouvrez le journal des messages du noyau avec la commande :

```
journalctl -ka
```

Il est possible de faire des recherches textuelles dans le journal avec la touche `/` (tout comme avec l'éditeur de texte `vim` ou le manuel `man`).

- Que trouvez-vous si vous cherchez le mot-clé `virtio` ?
- Que voyez-vous comme contrôleur de stockage si vous listez les périphériques PCI avec la commande `lspci` ?
- Que voyez-vous également comme carte graphique ? Est-ce une carte graphique emulée ou paravirtualisée ?
- Qu'observez-vous lorsque vous redimensionnez la fenêtre d'affichage de QEMU ?

Redémarrez la VM, mais cette fois-ci en spécifiant la carte graphique `-vga virtio`.

- Que voyez-vous comme carte graphique ? Est-ce une carte graphique emulée ou paravirtualisée ?
- Qu'observez-vous lorsque vous redimensionnez la fenêtre d'affichage de QEMU ?

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

- Pour chaque périphérique PCI, indiquez si il s'agit d'un périphérique émulé ou paravirtualisé, en justifiant chacune de vos réponse.

Observez les modules chargés dans le noyau de l'OS guest grâce à la commande `lsmod`.

- Observez-vous des modules liés à la paravirtualisation ? Si oui, quels sont leurs noms ? Pour chacun, essayez d'en déterminer le rôle.

Enfin, réalisez les mêmes observation sur la machine hôte.

- Que remarquez-vous ? Expliquez le comportement observé.

(3) Émulation

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` (package que vous devez par ailleurs installer) : <https://github.com/wimvanderbauwhede/limited-systems/wiki/Raspbian-%22stretch%22-for-Raspberry-Pi-3-on-QEMU>. Attention : le lien indiqué dans le blog télécharge l'image Raspbian Buster et non Stretch. Téléchargez donc le noyau et device tree correspondants, à savoir `kernel-qemu-4.19.50-buster` et `versatile-pb-buster.dtb`.

En vous aidant de ce qui est décrit dans le blog, utilisez QEMU pour booter sur l'image Raspbian que vous avez téléchargée. Si tout c'est bien passé, l'OS devrait booter avec succès et vous devriez arriver à un login. Spécifiez alors `pi` comme username et `raspberry` comme mot de passe.

Inspectez le CPU avec `lscpu`

- Quelle est l'architecture CPU ?
- En partant du principe que l'architecture de votre machine physique est x86 (AMD64 pour être précis), est-ce que QEMU joue le rôle d'hyperviseur ou pas ? Dans la négative, de quel mécanisme s'agit-il ?
- Peut-on ajouter le paramètre `-machine accel=kvm` à QEMU ? Est-ce que cela a un intérêt ou pas ? Faites le test, puis développez votre réponse.

(4) Interaction avec l'OS guest

Le but de cette partie est d'interagir avec l'OS guest, en commençant par la redirection de port, puis en vous familiarisant avec le QEMU Guest Agent (QGA).

Démarrez votre VM, mais en ajoutant la redirection de port de sorte à ce que depuis la machine hôte il soit possible de vous connecter en ssh à l'OS guest (cf. slides de cours). Faites en sorte que le trafic vers le port 4000 sur l'interface localhost de la machine hôte soit redirigé sur le port 22 de l'OS guest. Bien évidemment, n'oubliez pas d'installer un serveur ssh dans le guest.

Vérifiez que vous obtenez bien le résultat attendu en vous connectant à votre OS guest via ssh depuis votre machine hôte, puis éteignez votre VM depuis votre session ssh.

Vous allez maintenant aller plus loin en matière d'interaction grâce à l'utilisation du QEMU Guest Agent.

Démarrez à nouveau votre VM, mais en ajoutant les options nécessaires à l'utilisation du QEMU Guest Agent. Une fois le guest booté, vérifiez que le QEMU Guest Agent est en exécution (cf. slides de cours).

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. Écrivez ensuite un programme dans le script/langage de votre choix qui réalise les opérations suivantes :

- 1) Récupère le premier utilisateur du système
- 2) Crée le fichier `busted` dans le répertoire de l'utilisateur précédent, contenant un message de votre choix

3) Éteint gracieusement la VM

(5) Augmentation de la capacité de stockage

Vous réalisez que le disque de votre VM sera insuffisant pour vos besoins et qu'il vous faut plus d'espace. Vous désirez donc augmenter la capacité du disque avec 500GB supplémentaires. Deux méthodes s'offrent à vous pour réaliser ceci.

Avant de commencer par la première méthode, réalisez un backup (p.ex. `lubuntu.ori.qcow`) de l'image disque `lubuntu.qcow` car celle-ci sera modifiée.

Réalisez donc la première méthode qui consiste à :

- 1) Agrandir l'image disque avec l'outil `qemu-img resize` (la nouvelle taille peut être vérifiée avec la commande `qemu-img info`)
- 2) Booter sur une distribution live (p.ex. l'image ISO `lubuntu`) dans une VM contenant le disque à agrandir, puis une fois le système live prêt, utiliser l'outil `gparted` pour agrandir la partition du disque et le système de fichiers s'y trouvant.

Une fois la procédure terminée, rebootez et vérifiez avec `df -h` que le système de fichiers de votre OS guest fait bien une taille de 600GB.

Procédez maintenant avec la deuxième méthode qui implique les étapes suivantes :

- 1) Créer un nouveau disque de destination de la taille souhaitée avec l'outil `qemu-img create`
- 2) Utiliser l'outil `virt-resize` afin d'étendre le système de fichiers souhaité de l'image source en spécifiant l'image disque juste créée comme destination ; le système de fichiers sera alors étendu à la taille totale de l'image de destination
 - l'outil `virt-filesystems` permet de lister les partitions et systèmes de fichiers au sein d'une image (cf. `man virt-filesystems`)

Comme auparavant, bootez sur la nouvelle image et vérifiez avec `df -h` que le système de fichiers de votre OS guest fait bien une taille de 600GB.

(6) Script frontend à QEMU

À ce stade, vous avez une idée plutôt claire de l'utilisation de QEMU et de son fonctionnement. Nous sommes donc intéressés à développer le script bash `vm_run` permettant d'exécuter une VM via QEMU. Une partie de la configuration de la VM doit être configurable à l'aide d'arguments potentiellement optionnels.

La syntaxe du script `vm_run` doit être la suivante :

```
vm_run - launch a VM on the specified disk image(s).
The VM is configured with 2 CPUs and 4G of RAM.

Usage: vm_run -d disk -i iso
disk    the disk image to use (qcow file)
iso     the CD/DVD-ROM image to use (iso file)

At least one image must be specified.
```

La VM créée par votre script doit avoir le comportement suivant :

- Au minimum une image disque ou une image ISO doivent être spécifiées
- Si une image disque et une image ISO sont spécifiées, alors le système doit booter en priorité sur l'image ISO (investiguez l'argument `-boot`)
- La VM doit être créée et configurée avec :
 - KVM actif
 - 2 CPUs
 - 4 GB de RAM
 - une carte graphique paravirtualisée
 - une carte réseau paravirtualisée, configurée en NAT
 - un contrôleur de disque paravirtualisé
 - Le port 4000 de l'interface localhost sur l'hôte est redirigé vers le port 22 dans le guest
 - QEMU Guest Agent actif

Pour faciliter la maintenance et les changements dans votre script, déclarez des constantes pour la quantité de RAM et le nombre de CPUs.