Virtualisation

Professeur : Florent Gluck Assistant : Sebastien Chassot

March 11, 2022

Approfondissement de QEMU/KVM & LVM

Introduction

Le but de ce travail pratique est d'approfondir l'utilisation de QEMU/KVM, puis de se familiariser avec la virtualisation de stockage avec LVM.

Préparation

Une grande partie de ce travail pratique sera réalisée sur un hyperviseur (distant) se trouvant au sein de l'infrastructure d'HEPIA. Pour vous y connecter, vous utiliserez le login `student` et le mot de passe `cours_virtu_2202`. A noter que l'utilisateur `student` possède un accès `root` via `sudo`.

Dans le fichier `students_machines.txt` disponible sur le git du cours, vous pouvez trouver l'ip de la machine qui vous est dédiée. En premier lieu, changez le mot de passe de l'utilisateur `student` 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 hyperviseur 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 10.136.156.210
        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 10.194.186.210 (nommée NEXUS6) en faisant un saut sur une machine intermédiaire 10.136.156.150 (nommée HAL9000) :

```
Host HAL9000
    User dave
    HostName 10.136.156.150

Host NEXUS6
    User student
    HostName 10.194.186.210
    ProxyJump HAL9000
```

Ainsi, en tapant simplement `ssh NEXUS6`, on peut ainsi se connecter à la machine distante.

Information utile

Lors d'un connexion à 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 connextion active. Utilisation :

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

Exercice 1

Pour ce premier exercice vous utiliserez QEMU sur votre machine locale. Le but ici est de mettre en place un répertoire partagé entre la machine hôte et l'OS de la machine virtuelle guest.

En suivant les instructions décrites dans le cours, mettez en place un répertoire partagé dans la machine virtuelle Xubuntu 20.04 du labo précédent. Depuis la machine hôte, copiez des fichiers dans le répertoire partagé et vérifiez que vous pouvez accéder à ces fichiers en lecture et écriture depuis l'OS guest.

- A votre avis, est-il plus facile de mettre en place un répertoire partagé avec Virtualbox ou QEMU ?

Exercice 2

Vous allez maintenant explorer les possibilités offertes par QEMU en matière de snapshots grâce à l'outil `qemu-img`. Lisez la syntaxe des snapshots avec `man qemu-img` ou `qemu-img --help`.

Partie 2A

Nous allons commencer par explorer les snapshots de disque internes car ils sont plus simples à gérer.

Faites une copie du disque de votre machine virtuelle hepiadoom du labo précédent. Nommez cette copie `hepiadoom-backup.qcow`. Dans cette image, créez le script `createfile` qui prend en argument un nom de fichier et créé un fichier de 100MB de ce nom là. 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.

Faites une copie de l'image `hepiadoom-backup.qcow` dans `hepiadoom-snaps.qcow`. Cette image sera utilisée ici pour expérimenter avec les snapshots internes.

Réalisez un premier snapshot interne, taggé `snap1`. Exécutez ensuite une VM sur cette image et utilisez votre script `createfile` pour créer 3 fichiers `f1`, `f2`, `f3`.

Fermez la VM et réalisez un deuxième snapshot interne, `snap2`. Utilisez `qemu-img` pour lister les 2 snapshots créés. Rétablissez ensuite `snap1` et vérifiez que les 3 fichier créés précédemment n'existent plus.

- Quelle est la taille de l'image ?

Revenez au snapshot `snap1`, puis supprimez `snap2`. Exécutez une VM pour valider que les fichiers créés précédemment n'existent plus.

- Quelle est la nouvelle taille de l'image ? Celle-ci devrait rester inchangée malgré la suppression de `snap2`.

Partie 2B

Vous allez maintenant vous familiariser avec les snapshots externes.

Afin de faire un peu de place, vous pouvez supprimer l'image `hepiadoom-snaps.qcow`. Copiez ensuite `hepiadoom-backup.qcow` dans `hepiadoom-base.qcow`. Cette image sera l'image *backed* sur laquelle se baseront les snapshots (overlays) de cet exercice.

Soit la séquence de snapshots illustrés 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`.

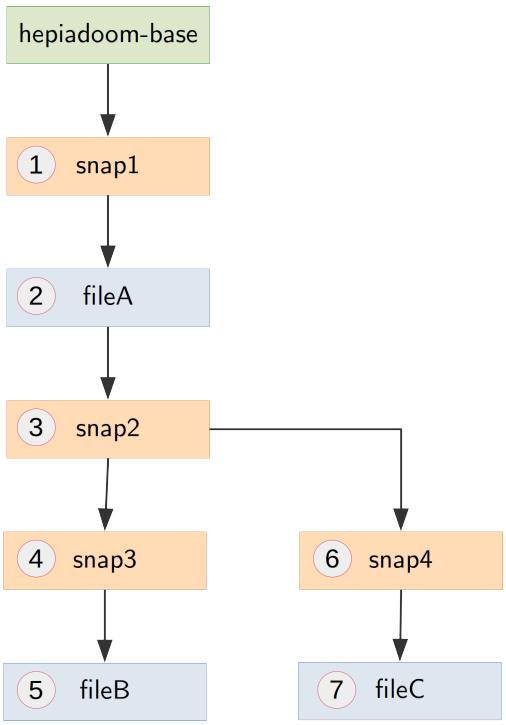

Figure 1: L'image *backed* est en vert, les snapshots (overlays) en orange et les fichiers ajoutés en bleu.

On part du principe que les 6 étapes ont été réalisées. Vous exécutez une VM avec l'image `snap3`.

- Quels fichiers `fileX` seront présents ?

Reproduisez alors 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 ?

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.

Effectuez maintenant un snapshot interne (de disque) de l'image `snap3`.

- Est-ce que cela fonctionne ?

Démarrez une VM avec l'image `snap3` et effacez-y tous les `fileX` qui s'y trouvent. Effectuez alors un autre snapshot interne de l'image `snap3`.

A l'aide de `qemu-img info`, affichez la chaîne d'images ayant mené à `snap3`.

- Qu'observez-vous en plus, par rapport à la chaîne que vous aviez affichée précédemment ?

Rétablissez le premier snapshot interne réalisé au point précédent afin que les fichiers `fileX` soient à nouveau présents.

Réalisez enfin un *merge* de type "commit" de l'image `snap4` dans l'image `snap1`, puis exécutez une VM utilisant `snap1` afin de vérifier que les `fileX` sont toujours présents.

- Est-ce que l'image `snap1` contient toujours les snapshots internes ?

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

- Quelle(s) image(s) peut/peuvent potentiellement être incohérente(s) ?

Au début de la partie 2A, vous aviez réalisé une copie de l'image `hepiadoom-backup.qcow` dans `hepiadoom-snaps.qcow` afin de pas modifier la première image.

- Qu'auriez-vous pu faire pour éviter de réaliser cette copie de plus de 9GB tout en évitant de modifier le fichier d'origine (`hepiadoom-backup.qcow`) ?

Partie 2C

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. Vous pouvez remarquer que celui-ci n'est pas instantané, contrairement aux snapshots 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.

Eteignez votre VM et listez les snapshots présents dans le fichier qcow2 avec `qemu-img`.

- Voyez-vous tous les snapshots réalisés ?

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`.

Exercice 3

A partir de maintenant, vous allez utiliser votre hyperviseur distant. Le disque principal de celui-ci (i.e. celui où est monté le système Linux, cf. `df -h`) se trouvera rapidement à court d'espace libre. Pour palier à ceci et fournir suffisamment d'espace pour les machines virtuelles à venir, vous allez créer un espace de stockage logique unifié grâce à LVM.

- Quel est l'espace disponible sur la partition où est monté le système de fichiers racine (`/`) ?

Inspectez la configuration des disques avec la commande `lsblk` et inspectez les partitions de chaque disque avec la commande `fdisk -l`. Identifiez le nombre de disques, leurs noms, leurs partitionnements (nombre de partitions, capacité et rôle de chaque partition), et la capacité de stockage totale de la machine.

Basé sur les résultats de cette investigation, mettez en place la structure de stockage décrite ci-dessous à l'aide de LVM :

- Un nouveau volume groupe constitué des deux disques qui se trouvent après le premier disque.
- Un nouveau volume logique de la capacité totale du volume groupe.
- Le volume logique est formatté en `ext4` (commande `mkfs.ext4`).
- Le volume logique est monté dans le répertoire de votre choix (choisissez un nom adéquat) ;
    - vous placerez les VMs et fichiers associés des exercices suivants dans ce volume logique.

N'oubliez pas de valider chaque étape réalisée !

- Quelle est la taille d'un physical extent ?
- Quelle est la capacité du volume groupe créé ci-dessus ?
- Peut-on agrandir le volume groupe en lui ajoutant le disque sur lequel se trouve le système d'exploitation ?
- Aurait-il été possible d'étendre le volume logique `ubuntu-lv` avec le contenu des disques `sdb`, `sdc` and `sdd` ?

Exercice 4

Dans cet exercice, vous utiliserez le volume logique créé à l'exercice précédent pour y stocker vos images de VMs.

A l'aide de `wget` ou `curl`, téléchargez le fichier `vm2fix.vmdk` se trouvant à cette URL dans le système de fichiers se trouvant sur le volume groupe `vg1`. Convertissez ensuite cette image au format `qcow2`.

Démarrez une VM intégrant un serveur Spice avec cette image disque et connectez-y vous avec un client Spice.

Sans connaître le login et mot de passe (qui sont bien sûre secrets) vous ne pourrez pas faire grand chose avec cette image. Pour vous connecter avec succès, il vous faut donc *hacker* cette image de disque !

En tant que hackeur en herbe, installez le paquet `libguestfs-tools`. Celui-ci contient toutes sortes d'outils pour manipuler les images disques dans un grand nombre de formats.

Tout d'abord, vous devez déterminer les partitions que comporte l'image de disque à hacker. C'est exactement le but du programme `virt-filesystems` (pensez à l'exécuter en `root` si vous obtenez le message d'erreur `libguestfs: error: /usr/bin/supermin exited with error status 1.`).

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

Ensuite, il faut déterminer ce que contient chaque partition. Pour cela, l'outil `guestmount` est d'une aide précieuse : il permet de monter un 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) changer pour pouvoir finalement vous logger correctement dans la VM. N'oubliez pas de démonter les partitions montées une fois que vous aurez terminé (avec `guestunmount` ou `umount`).

Voici quelques indices pour vous aider à *hacker* le disque de cette VM en vue de vous y connecter :

- Le fichier `/etc/passwd` contient les comptes utilisateurs, un par ligne, avec plusieurs informations importantes...
- Le fichier `/etc/shadow` contient les hash de chaque mot de passe...

Vous aurez la confirmation d'avoir réussi à *hacker* la VM lorsque pourrez vous connecter et aboutir 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).

- Dans cet exercice 4, pourquoi ne pas avoir simplement utilisé la technique décrite ici avec `mount` plutôt que de s'embêter avec `virt-filesystems` et `guestmount` ?

Exercice 5

Il devrait rester un disque inutilisé sur votre hyperviseur, donc étendez le volume groupe créé précédemment avec ce nouveau disque.

Etendez également le volume logique à l'intérieur afin qu'il utilise tout l'espace du volume groupe. Enfin, il est également nécessaire d'étendre le système de fichiers contenu dans ce volume logique avec l'outil `resize2fs`. Rappel : `df -h` liste les tailles des systèmes de fichiers montés sur le système.

Pour éviter toute coupure potentielle de service (dûe à un reboot p.ex.), veillez à réalisez ces opérations à chaud (*online*) !

Avant de poursuivre, assurez-vous de désactiver les services `multipathd.service` et `multipathd.socket` en exécutant les commandes suivantes :

```
sudo systemctl stop multipathd.service
sudo systemctl stop multipathd.socket
```

En effet, il semble que ce service pose problème avec les snapshots LVM.

On désire maintenant expérimenter avec les snapshots offerts par LVM. A savoir que lorsqu'on créé un snapshot d'un volume logique, LVM créé un volume logique pour le contenu du snapshot en question.

- Pourquoi doit-on définir une taille pour un volume de type snapshot ?

Effectuez un snapshot du volume logique "original" créé précédement...

Vous devriez toutefois rencontrer un problème d'espace disque car votre volume groupe n'a plus d'espace disponible !

Essayez alors de réduire la taille de votre volume logique de 50%.

- Que se passe-t-il si vous essayez de le faire à chaud ?

Réduire la taille d'un volume logique contenant un système de fichiers ext4 à chaud n'est malheureusement pas supporté. Pour cela, il faut passer par les étapes suivantes :

- Démonter le système de fichiers
- Réduire (redimensionner) la taille du système de fichiers
- Réduire la taille du volume logique
- Vérifier que le système de fichiers est valide (commande `fsck.ext4`)
- Monter le système de fichiers

Réalisez donc ces opérations et validez que tout c'est bien passé. Essayez alors à nouveau d'effectuer un snapshot du volume "original". Celui-ci devrait maintenant réussir. Vérifiez que vous avez bien un nouveau volume "snapshot" de la capacité souhaitée.

Montez le volume "snapshot".

- Quel est son contenu comparé au contenu du volume "original" ?

Populez maintenant le volume logique "original" avec plusieurs fichiers. Créez-y notamment un gros fichiers d'environ 10% de la taille du volume utilisé pour le snapshot. Inspectez l'espace utilisé dans le volume du snapshot.

- Que remarquez vous ?

Comparez à nouveau les fichiers dans le volume “original” avec ceux du volume “snapshot”.

- Quels sont les fichiers présents dans le volume “original” ?
- Quels sont les fichiers présents dans le volume “snapshot” ?

Vous décidez finalement de revenir à l'état initial (au moment où le snapshot avait été effecuté) en effectuant un *merge* du snapshot.

- Peut-on réaliser cette opération à chaud (*online*) ?
- Quelle a été la durée du **merge** (approximative) ?
- Une fois l'opération terminée, comment pouvez-vous valider que tout s'est bien passé ?
- Quels sont les avantages principaux à utiliser LVM par dessus des disques physiques, plutôt que d'y installer directement un système de fichiers comme `ext4` ?
- Pourquoi est-ce que la commande `lvconvert --merge` prend un certain temps à se terminer ?

Exercice 6

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 Raspbian (tips : 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 ?