Virtualisation

Professeur : Florent Gluck Assistant : Sebastien Chassot

April 13, 2022

Docker basics

Introduction

Le but de ce travail pratique est de vous familiariser avec les manipulations de base des containers Docker.

Préparation

Tous les travaux pratiques jusqu'à la fin du semestre sont à réaliser sur la machine distante qui vous a été fournie.

Ceux-ci nécessitent un volume de stockage important. Vos machines disposent de nombreux disques totalisant un espace de stockage de 110GB. Vous allez commencer ici par reconfigurer votre support de stockage avec LVM afin d'utiliser cet espace à bon escient.

Le système de fichiers racine de 20GB doit rester inchangé. Par contre, supprimez les contenus des 3 disques supplémentaires et créez un volume groupe de 90GB avec ces 3 disques. Créez-y un volume logique de la capacité totale du volume groupe. Créez alors un système de fichiers de type ext4 au sein de ce volume groupe et montez le dans `/mnt/data`. Créez ensuite le répertoire vide `/mnt/data/docker`, puis créez le lien symbolique `/var/lib/docker` afin qu'il pointe dessus.

Une fois ces opérations réalisées, procédez sur votre machine distante à l'installation de Docker. Pour rappel `apt-cache search` permet de réaliser une recherche par mot-clés dans les repositories Ubuntu/Debian. Faites attention à installer le bon package (lisez...).

Exercice 0

Nous désirons découpler le client Docker du daemon `dockerd`, c'est à dire que nous ne voulons pas qu'ils s'exécutent tous deux sur la même machine.

Sur le serveur, vérifiez que le service `dockerd` est bien actif et assurez-vous que celui-ci soit toujours démarré au démarrage de la machine.

Sur le client, configurez votre système afin que vous puissiez correctement exécuter le client Docker. Pour des raisons de sécurité, n'exécutez **jamais** le client Docker avec `sudo` ou en tant que `root`. Ce n'est pas nécessaire et c'est un trou de sécurité potentielle.

Finalement, configurez serveur et client afin que les deux puissent communiquer alors qu'ils se trouvent sur des machines physiques différentes. Un simple test comme `docker info` est suffisant pour vérifier que le client Docker est capable d'atteindre le serveur correctement et que tout fonctionne comme il se doit.

Exercice 1

Une série d'exercices se trouvent sur Le Docker hub. Commencez par les trouver à l'aide de la commande `docker search` (indice : hepia...).

Sans exécuter de shell interactif dans le container, déterminez l'image du premier exercice et affichez le contenu du fichier se trouvant à la racine de l'image. Procédez en deux étapes: listez d'abord les fichiers se trouvant à la racine, puis ensuite affichez le contenu du fichier. A noter qu'au lancement d'un container vous pouvez spécifier une commande à executer (p.ex `cat` ou `ls`).

- Combien de containers ont été créés et quels sont leurs noms et IDs ?
- Quel est l'état des containers ?
- Quel est l'ID de l'image utilisée par les containers que vous venez d'exécuter ?
- Est-ce que l'ID de l'image sera identique chez vos camarades de classe ?
- Est-ce que les noms et IDs des containers seront identiques chez vos camarades ?
- Sur quelle machine et à quel chemin absolu se trouvent les images Docker ?

Exercice 2

Exécutez le container `hepia/docker_ex02` mais cette fois-ci avec la commande `ls -l /ex02` et l'option `--rm`

- Que réalise l'argument `--rm` ?
- Quel est le nom du fichier se trouvant dans le répertoire `ex02` du container ?
- Quel est l'ID de ce container ?

Exercice 3

Exécutez le container `hepia/docker_ex03` avec un shell (`sh`) en mode interractif. Executez dans ce terminal la commande `touch /ex03/new_file`.

- Que se passe-t-il quand vous quittez le shell avec `ctrl+d` ?
- Si vous exécutez à nouveau le container avec `docker run`, est-ce que le fichier `/ex03/new_file` existe toujours ?

Exécutez un nouveau container basé sur la même image. Celui-ci s'appellera `solution_ex03`, puis dans celui-ci executez `touch /ex03/new_file` et pressez la combinaison de touches `ctrl+p ctrl+q`.

- Quel est le status de ce container ?
- Quel est donc le rôle de la combinaison de touches `ctrl+p ctrl+q` ?

Attachez-vous alors au container `solution_ex03`.

- Comment vous êtes-vous attaché au container ? Est-ce que le fichier `/ex03/new_file` existe encore ?
- Si vous exécutez la commande `docker exec -it solution_03 sh`, le fichier `/ex03/new_file` existe-t-il ? Combien de shells sont en cours d'execution (aide: `ps`) ?
- Quelle est la différence entre les commandes `attach` et `exec` ?

Redémarrez maintenant le container `solution_03` avec la commande `restart`.

- Le fichier `/ex03/new_file` existe-t-il toujours ?

- Comment stopper le container `solution_03` ?

Exercice 4

Exécutez un container basé sur l'image Alpine 3.15. Vérifiez que vous utilisez la bonne distribution en inspectant le contenu du fichier `/etc/alpine-release`.

La commande `ps aux` permet de lister les processus en cours d'exécution

- Est-ce qu'un processus appartenant à l'utilisateur root dans le container appartient également à root en dehors du container ? Décrivez une méthode permettant de valider votre affirmation.
- Pourquoi est-il dangereux qu'un processus UID 0 dans le container soit également UID 0 sur la machine hôte ?
- Quelle différence remarquez vous lorsque vous exécutez `ps aux` dans le container et sur la machine hôte ?
- A votre avis, quelle est la raison de cette différence ?

A l'intérieur du container installez et exécutez le programme `htop`.

- Quel est le PID de ce processus dans le container ? Quel est le PID de ce même processus sur la machine hôte et pourquoi est-il différent ?

Vous allez maintenant investiguer empiriquement le comportement des containers.

- Quel est le processus portant le numéro de PID 1 dans le container ? Quel est le processus portant le même numéro de PID sur la machine hôte ?
- Pour généraliser, quel est le processus portant le PID 1 dans le container (càd quel que soit le container exécuté) ?
- Comparer la commande `uname -rv` à l'intérieur et dehors du container. Que remarquez-vous ?
- Comparer le contenu du répertoire `/dev/` à l'intérieur et dehors du container. Que remarquez-vous ? A votre avis quelle est la raison de cette différence ?
- Est-il possible de donner à un container accès à un ou plusieurs périphériques de la machine hôte ? Comment ? Trouver un moyen permettant de vérifier votre méthode.
- Exécutez la commande `mount` sur la machine hôte et dans le container. Quelles différences remarquez vous, notamment en ce qui concerne le système de fichiers racine ?
- Retrouvez-vous l'entrée définissant le système de fichiers racine du container sur la machine hôte ? Que pouvez-vous donc conclure du système de fichiers racine du container ?

Sortez du container, puis réaliser 5 fois l'opération suivante : exécutez un container Alpine:3.15, puis terminez-le avec ctrl+d.

Listez enfin tous les containers avec la commande `ps -a`.

- Que remarquez-vous ?