Docker – TP 2
Système de fichiers et volumes

Objectif

Le but de ce TP est de se familiariser à avec le fonctionnement du système de fichiers docker (UFS) et utiliser des volumes pour améliorer les performances et avoir de la persistance dans la modification de vos données.

Machine virtuelle à configurer

Utiliser la VM VirtualBox que vous avec mise en place lors du TP précèdent avec docker déjà installé. Pour commencer avec un environnement propre, nous allons effacer les configurations précédentes :

1. Arrêtez tous les containers en exécution :
```
$ docker stop $(docker ps –a –q)
```
2. Effacer les containers existants :
```
$ docker rm $(docker ps –a –q)
```
3. Effacer les images créées :
```
$ docker rmi $(docker images –a –q)
```

1. Layers et copie en écriture

Dans cette étape vous allez explorer le fonctionnement des layers et comment les fichiers écrits dans un container sont gérés par le processus *copy on write*.
Vous pouvez regarder la documentation officielle docker pour avoir plus de détails sur son fonctionnement :
https://docs.docker.com/storage/

Exercice

1. Téléchargez l'image de debian stretch minimale :
```
$ docker image pull debian:stretch–slim
```
   - Combien de layers ont été téléchargés ?
   - Quel est l'ID de ces layers ?
   - Quelle est la taille des layers ?

2. Téléchargez l'image de mysql :
```
$ docker image pull mysql
```
   - Quelles différences trouvez-vous par rapport à la sortie du téléchargement de l'image debian ?
   - Quel est l'ID de ces layers ?
   - Quelle est la taille des layers ?
   - Le premier layer a le même ID que le layer de l'image debian téléchargée précédemment, pourquoi ?
   - Regardez le Dockerfile de l'image mysql :
     https://github.com/docker-library/mysql/tree/master/5.7
     Quelles lignes de ce fichier ont générées les différents layers de l'image mysql ?

3. Instanciez un container de l'image debian et démarrez-y un shell :

```
$ docker run --tty --interactive --name debian debian:stretch-slim
```

4. Créez un fichier et listez le contenu du dossier pour vérifier la création du fichier :

```
# touch mon-fichier
# ls -l
```

   - Le fichier a été bien crée ?
   - Quel est le chemin d'accès à votre fichier ?

   Le fichier est copié dans le système de fichiers du container. Ce système de fichiers est géré par le contrôleur docker et il utilise la fonction *copy on write*, dès qu'une opération d'écriture est détectée, les changements sont écrits dans le layer de lecture-écriture du container. Ce layer correspond à un dossier du système de fichiers de la machine hôte.

5. Sortez du container mais laissez-le en exécution avec la combinaison de touches `ctrl+p` et `ctrl+q`.
   La machine virtuelle ubuntu que vous avez démarrée utilise OverlayFS avec le contrôleur de stockage `overlay2`.

   OverlayFS superpose deux dossiers dans l'hôte Linux et il les présente comme un seul dossier. Ces dossiers sont appelés `layers` et le processus d'unification est appelé `union mount`. OverlayFS appele la couche d'en bas « lowerdir » et la couche d'en haut « upperdir ». *Upper* et *Lower* font référence au moment d'ajout du layer à l'image. Dans notre exemple, la couche d'écriture, c'est la couche supérieure. La vue unifiée est montrée par son propre dossier appelé « merged ».

   Vous pouvez utiliser la commande `docker inspect` pour vérifier où sont situés les dossiers des layers. Une guide pour cette commande est ici :
   http://container-solutions.com/docker-inspect-template-magic/

   Le système de fichiers du container debian lancé précédemment peut être trouvé avec cette commande :

```
$ docker inspect -f '{{json .GraphDriver.Data}}' debian | jq
```

   Le programme `jq` n'est pas installé par défaut, il faut l'installer avec `apt-get`.

6. Listez les contenus du dossier *upper*, *lower* et *merge*:

```
$ cd $(docker inspect -f '{{.GraphDriver.Data.UpperDir}} debian)
$ ls
```

   - Quel est le contenu de chaque dossier ?

7. Créer un nouveau fichier dans le *UpperDir* et lister son contenu dans le hôte et dans le container.
   Pour se reconnecter à votre instance debian, vous utilisez cette commande :

```
$ docker attach debian
```

8. Pour sortir du container et l'arrêter :

```
# exit
```

9. Le container est arrêté mais l'instance est toujours dans votre hôte, vérifiez qu'elle y est toujours :

```
$ docker container ls --all
```

10. Listez le contenu du dossier *UpperDir*, les contenus sont toujours là ?

```
$ ls
```

11. Tant que le container est dans la machine, tous les fichiers seront sauvegardés. Effacer le container est listez encore une fois le contenu du dossier :

```
$ docker container rm debian
```

- Les fichiers sont toujours dans le dossier *UpperDir* ?
- Le dossier *UpperDir* existe toujours ?

2. Volumes anonymes docker

Les volumes docker sont des dossiers dans le système hôte qui ne sont pas gérés par le contrôleur de stockage docker. Vous pouvez vous renseigner plus en détails dans la documentation docker : https://docs.docker.com/storage/volumes/

Si vous regardez de nouveau le Dockerfile pour mysql, vous allez trouver la ligne :

```
VOLUME /var/lib/mysql
```

Cette ligne met en place un volume anonyme pour incrémenter la performance de la base de données, de cette façon toutes les opérations d'écriture des données ne vont pas être gérées par le contrôleur de stockage docker. Étant un volume anonyme, les données ne sont pas persistantes.

Exercice

1. Démarrez un container mysql :

```
$ docker run --name mysqldb -e MYSQL_USER=mysql -e MYSQL_PASSWORD=mysql -e
MYSQL_DATABASE=exemple -e MYSQL_ROOT_PASSWORD=toto -d mysql
```

   Au démarrage du container, le volume anonyme sera crée.

2. Utilisez docker inspect pour voir les détails du volume anonyme :

```
$ docker inspect -f 'in the {{.Name}} container {{(index .Mounts
0).Destination}} is mapped to {{(index .Mounts 0).Source}}' mysqldb
```

3. Listez le contenu du dossier associé au volume:

```
$ cd $(docker inspect -f '{{(index .Mounts 0).Source}}' mysqldb)
$ ls
```

   - Quelle différence remarquez-vous par rapport aux dossiers créés pour les layers ?
   - Quels sont les contenus du volume ?

4. Ouvrez un shell dans le container mysql et faites login dans MySQL :

```
$ docker exec --tty --interactive mysqldb bash

//dans votre container:
# mysql --user=mysql --password=mysql
```

5. Créer une nouvelle table :

```
mysql> show databases;
mysql> connect exemple;
mysql> show tables;
mysql> create table user(name varchar(50));
mysql> show tables;
```

6. Sortez de mysql et du container :

```
mysql> exit
$ exit
```

7. Arrêtez le container et redémarrez-le :

```
$ docker container stop mysqldb
$ docker container start mysqldb
```

8. Faites login de nouveau dans MySQL, la table existe encore ?

9. Sortez de mysql et du container.

10. Vérifiez le chemin du volume anonyme créé pour le container mysql.
    - Est-ce que l'opération d'arrêter et démarrer le container a créé un nouveau volume ?

11. Supprimez le container mysql :
```
$ docker container rm --force mysqldb
```

12. Démarrez un nouveau container MySQL avec les mêmes paramètres d'avant :

13. Listez les détails du volume pour le nouveau container
    - Le dossier de l'hôte c'est le même que pour le container précèdent ?
    - Le dossier des volumes créés sont-ils conservés ?

14. Listez les contenus de la base de données exemple.
    - La table user existe toujours ?

15. Sortez de mysql et du container et supprimer le container.
    - Les dossiers des volumes existent toujours ?
    - Quelles solutions donnerez-vous pour sauvegarder les données d'un volume ?

16. Notez vos conclusions par rapport aux volumes anonymes.

3. Volumes nommés docker

Un volume nommé est un volume qu'a été explicitement nommé et que peut être facilement référencé. Il peut être créé à partir de la ligne des commandes, dans un fichier docker compose ou au démarrage d'un container. Les volumes ne peuvent pas être créés dans un Dokerfile.

Exercice

1. Démarrez un container MySQL avec un volume nommé dbdata :
```
$ docker run --name mysqldb \
  -e MYSQL_USER=mysql \
  -e MYSQL_PASSWORD=mysql \
  -e MYSQL_DATABASE=exemple \
  -e MYSQL_ROOT_PASSWORD=toto \
  --detach \
  --mount type=volume,source=mydbdata,target=/var/lib/mysql \
  mysql
```

   Le nouveau volume sera initialisé avec les données trouvées dans le dossier `/var/lib/mysql` de l'image.

2. Les volumes docker peuvent être listés et supprimés comme les images et les containers. Pour lister les volumes la commande est :
```
$ docker volume ls
```

3. Regardez les détails du volume :
```
$ docker inspect mydbdata
```

   Toute donnée écrite dans /var/lib/mysql sera par la suite enregistré dans le volume qui vient d'être créé.

4. Ouvrez un shell dans le container mysql et faites login dans MySQL.

5. Créez une nouvelle table `user(name varchar(50))`.

6. Sortez de mysql et du container.

7. Supprimez le container.

   - Le volume existe toujours ?
   - Quels sont les contenus du dossier ?

8. Démarrez un nouveau container MySQL avec le même container nommé, faites login dans le shell mysql et vérifiez le contenu de la base de données.
   - La table user a été supprimée ?

9. Sortez du container et supprimez-le, supprimez aussi le volume avec la commande :

```
$ docker volume rm mydbdata
```

4. Application dans le projet

- Quels avantages des volumes pourriez-vous donner à votre projet web?
- Quels volumes serez nécessaires pour votre projet web ?

En cas de pannes

- Vérifier toujours que vous avez les permissions pour utiliser les fichiers, au cas où l'utilisateur n'a pas les permissions, vous pouvez toujours changer les permissions à l'aide de la commande :

```
$ sudo chown [–R] $(utilisateur):$(groupe) $(fichier ou dossier)
```

Ou

```
$ sudo chmod [–R] $(permissions) $(fichier ou dossier)
```

- Si un container est démarré avec les mauvais paramètres, vous pouvez l'arrêter et l'effacer avec les commandes :

```
$ docker stop [id ou nom container]
$ docker rm [id ou nom container]
```