Gestion des données

Introduction

1. Lancer un conteneur basé sur l'image **alpine** ayant pour nom "**data_test**".

2. Dans le conteneur, écrire un fichier dans **/data** :

```
$ mkdir /data
$ echo "my data" > /data/hello.txt
```

3. Redémarrer le conteneur :

Entrer **exit** dans le conteneur, puis :

```
$ docker start data_test
```

- Se connecter au conteneur et afficher le contenu du répertoire **/data**. Que peut-on observer ?

4. Stopper et supprimer le conteneur **data_test**, en créer un nouveau identique et étudier le contenu du dossier **/data**.

Bind mounts

Le paramètre permettant de monter des répertoires dans des conteneurs est `-v`. Il s'utilise en prenant un argument décrivant les deux répertoires à utiliser :

```
-v <absolute-file-path-host>:<absolute-file-path-container>
```

1. Créer un répertoire local nommé **nginx-sources** contenant un fichier HTML (**index.html**). Écrire dans ce fichier quelques lignes HTML affichant une page basique.

2. Chercher dans la documentation de l'image **nginx** le dossier de destination à utiliser avec le paramètre `-v` dans lequel placer les sources applicatives.

3. Démarrer un conteneur en montant ce répertoire (attention, le répertoire source doit être spécifié de façon absolue) :
   Se documenter sur la commande pwd si vous ne la connaissez pas.

```
docker run -it --name bm -v
```

```
$(pwd)/nginx-sources:<dossier-de-destination> -p 8080:80 nginx
```

4. Vérifier en accédant à **localhost:8080** que ce dossier est bien utilisé et affiche votre page HTML.

5. Essayez de changer le contenu du fichier HTML sur le système hôte, que constatez-vous ?
6. Renommer le dossier sur le système hôte et vérifier ce que nginx au navigateur. Comment pourriez-vous expliquer ce comportement ?
7. Redémarrer le conteneur précédant (**docker restart**). Que constatez-vous ?
8. Inspecter le conteneur et chercher si une information nous permet de savoir si un dossier est monté et si oui, quelles informations sont disponibles.

Volumes

La syntaxe de la commande pour monter un volume est légèrement identique à celle utilisée pour les bind-mounts, à la différence que l'on spécifie le nom du volume à la place d'un dossier source :

```
-v <volume-name>:<absolute-file-path-container>
```

1. En utilisant cette syntaxe, créer un conteneur nommé **vol_test** basé sur l'image **alpine** utilisant le volume **vol_demo**, monté dans le répertoire de destination **/data**.

2. Lister les volumes présents sur le système :

```
docker volume ls
```

- Que remarquez-vous ?

3. Se connecter au conteneur si ce n'est pas déjà le cas, et créer un fichier dans le répertoire **/data** :

```
docker exec -it vol_test sh
touch /data/fichier_de_test.txt
```

4. Supprimer le conteneur précédemment créé et en créer un nouveau utilisant les mêmes paramètres.
    - Le répertoire **/data** est-il présent ?
    - Contient-il quelque chose ?

5. En laissant fonctionner le conteneur précédemment créé, en créer un nouveau nommé **vol_test_2** utilisant le même volume **vol_demo**.
    - Créer un nouveau fichier dans ce conteneur à partir de ce second conteneur :

```
docker exec -it vol_test_2 sh
touch /data/fichier_from_vol_test_2.txt
```

6. On a maintenant deux conteneurs **vol_test** et **vol_test_2** fonctionnant tous les deux. Le fichier **/data/fichier_from_vol_test_2.txt** est-il visible depuis le premier conteneur ?

7. Supprimer tous les conteneurs utilisant ce volume, et lister les volumes sur le système hôte. Que remarquez-vous ?

8. Trouver quelques cas d'usages où le fait de monter un volume partagé par plusieurs conteneurs serait intéressant.

Syntaxe alternative

Il existe une option alternative à l'option `-v` pour monter les répertoires et les volumes.

- Quelle est-elle et chercher dans quels cas elle est plus utile que la première (`-v`) ?