Faire connaissance avec les conteneurs

Premier conteneur

Lancer un conteneur simple:

- Pour chaque commande suivante, essayer de trouver une explication sur ce qu'il se passe vis à vis du résultat de la commande.

```
docker run -i alpine
```

- Que se passe-t'il ?
- Exécuter une commande (par exemple, **ls**) ; Que se passe-t-il ?
- Quitter le conteneur en tapant "**exit**".

```
docker run -i -t alpine
```

- Exécuter une commande (par exemple, **ls**). Quelle est la différence par rapport à la commande précédente ?
- Quitter le conteneur en tapant "**exit**".

Dans deux terminaux différents, lancer les commandes suivantes :

```
docker run -it --name alp1 alpine
docker run -it --name alp1 alpine
```

- Que se passe-t-il lors de l'exécution des deux commande précédentes ? En déduire l'utilité du paramètre `--name`.

En se basant sur les résultats précédents et la documentation de docker, expliquer à quoi correspondent ces options :

- -i ?
- -t ?
- --rm ?
- --name ?

Lister les conteneurs de la machine

La commande permettant de lister les conteneurs est la suivante :

```
docker ps
```

Par défaut, elle liste les conteneurs actuellement en cours de fonctionnement. Pour lister les conteneurs qui ne sont pas en cours de fonctionnement, il faut utiliser l'option `-a` (pour **all**).

```
docker ps -a
```

- Quelles informations sont disponibles ?

```
docker run -it -d --name ng2 nginx
```

- Observer les conteneurs présents sur la machine grâce à `docker ps`. En déduire l'utilité de l'option `-d`.

Stopper un conteneur

La commande est la suivante :

```
docker stop <container>
```

- Exécuter la commande avec l'un des conteneurs créé précédemment. Quelle est la sortie de la commande ?

Supprimer un conteneur

La commande est la suivante :

```
docker rm <container>
```

- Exécuter la commande avec l'un des conteneurs créé précédemment. Quelle est la sortie de la commande ?

Inspection des conteneurs

Les conteneurs possèdent beaucoup d'informations utilisées de façon interne à docker, qui peuvent se révéler très utile en cas de débogage.
Elles peuvent être récupérées grâce à la commande suivante :

```
docker inspect <container-name>
```

- Exécuter cette commande sur l'un de vos conteneur, et citer quelques informations qui semblent pertinentes.

Exécuter une commande arbitraire dans un conteneur.

La commande *run* peut prendre en paramètre une commande à exécuter dans le conteneur :

```
docker run -it --rm alpine echo 'hello world'
```

Par défaut, une image possède une commande lancée à son démarrage. Elle peut être changée au moment du lancement du conteneur.
Certaines images ne démarrons pas correctement si elle est modifiée, d'autres si elle n'est pas spécifiée.

- Lancer un conteneur basé sur l'image "alpine" et passer en paramètre la commande permettant d'afficher le nom de la machine (**hostname**).

- Pourquoi les conteneurs sont-ils stoppés lorsqu'on ne passe qu'une commande, ou que l'on quitte le shell (avec **exit** par exemple) ?

Se "connecter" à un conteneur actif

Pouvoir se connecter à un conteneur actif peut être très utile, notamment s'il ne fonctionne pas comme prévu, pour pouvoir le déboguer.

La commande à exécuter se présente sous la forme suivante:

```
docker exec -it <container-name> <command>
```

- Lancer un conteneur **apache** (le nom de l'image est **httpd**) en mode "détaché" ;
- S'y connecter en utilisant soit son identifiant, soit son nom, en utilisant la commande **bash** dans le conteneur ;
- Installer la commande ps dans le conteneur:

```
apt update
apt install -y procps
```

- Lister les processus s'y exécutant (commande **ps aux**) ; Quels sont-ils ?
- Tuer le processus ayant le plus petit PID (process identifier) grâce à la commande **kill <pid>** ; Que se passe-t-il ?

Afficher les logs d'un conteneur

Lorsqu'un conteneur fonctionne en tâche de fond, il sera très utile de pouvoir afficher sa sortie standard, pour par exemple déboguer.

- lancer un conteneur **nginx** en tâche de fond, puis afficher ses logs grâce à la commande suivante :

```
docker logs <container>
```

- trouver et exécuter l'argument permettant d'afficher les logs en temps réel ;