TP6 - Docker
ING1-GI-GM – Systèmes d'exploitation
Année 2018–2019

1 Introduction

L'objectif de ce TP est de vous apprendre à exécuter, gérer et installer des conteneurs Docker. Par défaut, vous avez Docker déjà installé sur votre ordinateur EISTI.
Pour tester votre installation de Docker, démarrez une terminal et exécutez la commande :

```
$ docker run hello-world
```

Cela téléchargera et exécutera votre premier conteneur. Vous devrez voir un message de bienvenue et une explication de la démarche suivie pour l'y exécuter.

2 Démarrage et gestion d'un conteneur

Un conteneur est créé à partir d'une image de base qui utilise le kernel du host. Sur cette image de base on ajoutera des couches logicielles additionnelles. Docker s'occupe de gérer tout le processus. Pour démarrer notre premier conteneur "utile", exécutez la commande :

```
$ docker run -it ubuntu /bin/bash
```

où `ubuntu` est une image existant dans le Docker Hub, et les options `-it` servent à exécuter la commande /bin/bash de façon interactive dans le terminal. Ensuite, réalisez les actions suivantes :

1. Comparez la version du kernel du conteneur et du host (avec `uname -a`).
2. Consultez l'image téléchargée par Docker avec la commande "`docker images`" sur l'host.
3. Comparez le système de fichiers du conteneur et du host avec `df -h`. Essayez de créer un fichier dans le conteneur en exécution. Existe-t-il dans la machine host ?
4. Depuis une ligne de commande du host, arrêtez le conteneur avec
   `docker stop conteneur_id`
   On peut obtenir le `conteneur_id` avec la commande
   `docker ps -a`
   Redémarrez le conteneur avec `docker start conteneur_id`, connectez-vous avec `docker attach conteneur_id` et vérifiez que le fichier existe encore.

5. Nous allons installer firefox dans un nouveau conteneur :
   — Pour démarrer une application graphique, il faut permettre la connexion sur le serveur graphique du host, en exécutant '`xhost +`' dans une ligne de commandes du host. Ensuite, on devra exporter le display au moment du démarrage du conteneur :

     ```
     $ docker run -it --rm -e DISPLAY=$DISPLAY \
                    -v /tmp/.X11-unix:/tmp/.X11-unix ubuntu /bin/bash
     ```

   — Une fois sur la ligne de commande du conteneur, installez firefox avec apt-get (en exécutant `apt-get update` avant).
   — Vérifiez que vous pouvez exécuter votre version containerisée de firefox.
6. Ensuite, vous pouvez sauvegarder localement votre conteneur avec firefox dans une nouvelle image avec `docker commit container_id image_name`
7. Arrêtez le conteneur et effacez le avec `docker rm conteneur_id`. Démarrez un nouveau conteneur à partir de votre nouvelle image et vérifiez que vous pouvez toujours exécuter firefox.
8. Finalement, vous verrez qu'il existent beaucoup d'applications déjà containerisées qu'on peut exécutez directement. Téléchargez une image de l'éditeur *sublime-text-3* avec `docker pull jess/sublime-text-3` et l'exécutez. Vérifiez la différence entre exécuter l'image directement avec et sans `/bin/bash`.

3 Docker Hub et création d'images

Lorsqu'on commence à créer plusieurs conteneurs, leur gestion devient plus complexe. Pour cela on peut utiliser des fichiers appelés '*Dockerfiles*' qui, de la même façon qu'un script en bash, permettent l'exécution de tâches associées aux conteneurs. On va voir un exemple :

— Exécutez l'image `docker/whalesay` avec les options `'cowsay bonjour'`.

— Pour faire parler la baleine plus couramment, on va ajouter le programme `fortunes` et, ensuite, on va créer une nouvelle image avec ces modifications. Pour ce faire, créez un fichier appelé *Dockerfile* avec le contenu suivant :

```
FROM docker/whalesay:latest
RUN apt-get -y update && apt-get install -y fortunes
CMD /usr/games/fortune -a | cowsay
```

— Pour pouvoir lire le Dockerfile, ce fichier doit être disponible dans le même répertoire où vous allez exécuter la commande de création de l'image. Tapez

```
$ docker build -t docker-baleine .
```

et vérifiez que la nouvelle image a été bien créée. Ensuite, exécutez un conteneur à partir de cette image.

— En considérant que notre nouvelle image est très utile pour la communauté d'utilisateurs de Docker, on va la rendre disponible pour tout le monde dans le Docker Hub. Allez à `https://hub.docker.com/` et créez un compte. Une fois votre compte validé, faites login et créez un nouvel repo appelé `docker-baleine` avec de la visibilité publique.

— De votre terminal, ajoutez un *namespace* (votre login du Docker Hub) à l'image que vous avez créée, avec la syntaxe :

```
$ docker tag image_id dockerhublogin/docker-baleine:latest
```

Vérifiez que votre image a été étiquetée avec le tag `latest`

— Faites login sur votre compte :

```
$ docker login --username=loginhub --email=courriel@mondomain.com
```

— Envoyez l'image au Hub:

```
$ docker push dockerhublogin/docker-baleine
```

À partir de ce moment, votre image sera disponible dans Docker Hub et tout le monde pourra la télécharger.

4 Microservices avec docker-compose

4.1 Microservices "à la main"

Docker permet de relier plusieurs conteneurs avec des fonctions complémentaires. Dans cet exercice, on va démarrer un conteneur avec un serveur Wordpress relié à un autre conteneur avec une base de données. Exécutez la commande suivante pour créer un conteneur avec mysql :

```
$ docker run --name wordpressdb -e MYSQL_ROOT_PASSWORD=password \
        -e MYSQL_DATABASE=wordpress -d mysql:5.7
```

Ensuite, on va télécharger une image wordpress et l'exécuter en la reliant à la base de données

```
$ docker pull wordpress
$ docker run -e WORDPRESS_DB_PASSWORD=password -d --name wordpress \
             --link wordpressdb:mysql -p 127.0.0.2:8080:80 \
             -v "$PWD/":/var/www/html  wordpress
```

Allez sur votre navigateur et tapez `http://127.0.0.2:8080`. Vous devrez voir l'écran d'installation de wordpress. Notez que, dans ce cas, nous avons donné l'adresse IP de notre conteneur Wordpress. Si on ne le fait pas, il faudra utiliser la commande `docker inspect wordpress` pour trouver l'adresse IP donnée automatiquement par Docker.

4.2 Un peu plus professionnel

Il est possible de faire la même démarche de façon automatique avec le logiciel `docker-compose`. Tapez `docker-compose version`. Si aucun message ne se montre, installez `docker-compose` avec la commande :

```
$ sudo apt install docker-compose
```

— Créez un dossier et, à l'intérieur, créez un fichier `docker-compose.yml` avec le contenu suivant :

```
version: '2'
services:
  db:
    image: mariadb:latest
    mem_limit: 256m
    container_name: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: "password"
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: "password"

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    links:
```

```
    - db
  ports:
    - "8080:80"
  restart: always
  environment:
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_NAME: wordpress
    WORDPRESS_DB_PASSWORD: "password"
    WORDPRESS_DB_HOST: mariadb
    MYSQL_PORT_3306_TCP: 3306
```

— Exécutez `docker-compose up -d` (en background) ou `docker-compose -verbose up` (pour afficher les informations au démarrage) dans le dossier qui contient le fichier `yml`.

— Vérifiez la différence entre arrêter les conteneurs (`docker stop conteneur-id`) et les supprimer une fois qu'ils sont arrêtés : `docker rm -v $(docker ps -a -q -f status=exited)`