Systèmes d'exploitation
TP – Docker
ING1 Informatique - Mathématiques appliquée – MIM
Année 2023–2024

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 CY-Tech.

Pour tester votre installation de Docker, démarrez un 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, des couches logicielles additionnelles seront ajoutées. Docker s'occupe de gérer tout le processus. Pour démarrer notre premier conteneur "utile", exécutez la commande :
$ docker run −it ubuntu:xenial-20181005 /bin/bash

où **ubuntu:xenial-20181005** 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 la commande **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 la commande **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 la commande
$ docker stop conteneur_id

Vous pouvez obtenir le **conteneur_id** avec la commande

$ docker ps -a
Redémarrez le conteneur avec la commande docker start conteneur_id, connectez-vous avec la commande 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 la commande 'xhost +' dans une ligne de commande du host.

Ensuite, on devra exporter le display au moment du démarrage du conteneur :
**$ docker run -it -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 la commande
apt-get update (avant installation)

— 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 la commande docker commit container_id image_name
7. Arrêtez le conteneur et effacez-le avec la commande docker rm container_id. Démarrez un nouveau conteneur à partir de votre nouvelle image et vérifiez que vous pouvez toujours exécuter Firefox.
8. Finalement, vous verrez qu'existent beaucoup d'applications déjà containerisées prêtes à être exécutées directement. Téléchargez une image de l'éditeur *sublime-text-3* avec la commande docker pull jessfraz/sublime-text-3 et l'exécutez.

3 Docker Hub et création d'images

Lorsqu''on commence à créer plusieurs conteneurs, leur gestion devient plus complexe. Pour cela nous pouvons utiliser des fichiers appellés ***Dockerfiles*** qui, de la même façon qu''un script en bash, permettent l'exécution de taches 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, nous allons ajouter le programme fortunes et, ensuite, nous allons créer une nouvelle image avec ces modifications. Pour ce faire, créez un dossier vide (par exemple : mkdir mabaleine) et, à l'intérieur, créez un fichier appelé ***Dockerfile***

avec le contenu suivant :

FROM docker /whalesay : latest
RUN apt – get – y update && apt – get ins tall – y fortunes
CMD /usr /games/ fortune – a | cowsay

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

$ docker build – t docker – baleine .

(Attention au point à la fin de la ligne pour indiquer le dossier courant) 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 sur **https://hub.docker.com/** et créez un compte. Une fois votre compte valide, faites login et créez un nouvel repo appelé **docker-baleine** avec visibilité publique.

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

$ docker tag image_id votrelogin /docker – baleine : latest

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

— Faites login sur votre compte :

$ docker login --username=votrelogin

— Envoyez l'image au Hub:

$ docker push votrelogin/docker – baleine

A partir de ce moment, votre image sera disponible sur Docker Hub pour tout ce qui veut 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, nous allons démarrer un conteneur avec un serveur Wordpress relie à un autre conteneur avec une

base de données. Tout d'abord, créez un dossier vide (par exemple : **mkdir mondossierwp**) et placez-vous là (**cd mondossierwp**). Exécutez la commande suivante pour créer un conteneur avecmysql :

**$ docker run --name db −e MYSQL_DATABASE=wordpressdb −e MYSQL_USER=exampleuser **
−e MYSQL_PASSWORD=examplepass −e MYSQL_RANDOM_ROOT_PASSWORD= '1' mysql :5.7

Ensuite, téléchargez une image Wordpress et l'exécutez en la reliant a la base de données

$ docker pull wordpress
**$ docker run −e WORDPRESS_DB_HOST=db −e WORDPRESS_DB_USER=exampleuser **
**−e WORDPRESS_DB_PASSWORD=examplepass −e WORDPRESS_DB_NAME=wordpressdb **
--name wordpress −p 8080:80 −v wordpress: / var /www/html --link db:mysql wordpress

Allez sur votre navigateur et tapez **http://localhost:8080**. Vous devrez voir l'écran d'installation de Wordpress.

4.2 Un peu plus professionnel

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

$ sudo apt install docker−compose

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

version: '3.1'

services:
wordpress:
image: wordpress
restart: always
ports:
- **8080**:80
environment:
WORDPRESS_DB_HOST: db
WORDPRESS_DB_USER: exampleuser
WORDPRESS_DB_PASSWORD: examplepass
WORDPRESS_DB_NAME: exampledb

```
volumes:
- wordpress:/ var /www/html

db:
image: mysql:5.7
restart : always
environment:
MYSQL_DATABASE: exampledb
MYSQL_USER: exampleuser
MYSQL_PASSWORD: examplepass
MYSQL_RANDOM_ROOT_PASSWORD: '1'
volumes:
- db:/ var / lib /mysql

volumes:
wordpress:
db:
```

— Exécutez la commande **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 (commande **docker stop conteneur-id**) et les supprimer une fois qu'ils sont arrêtés : commande **docker rm -v $(docker ps -a -q -f status=exited)**