Formation Kubernetes

TP1 - Création d'images personnalisées.

Nous commençons par créer un dossier et télécharger les sources de l'image, en cliquant
ici .

Désarchiver le fichier zip, et mettez les dossiers suivants dans votre nouveau dossier :

- db : contient un fichier articles.sql, qui renferme toute l'architecture de la base de données.
- app : comporte les sources php de notre application web.

Création du dockerfile

```
# ---------- DÉBUT COUCHE OS ----------
FROM debian:stable-slim
# ---------- FIN COUCHE OS ----------


# MÉTADONNÉES DE L'IMAGE
LABEL version="1.0" maintainer="AJDAINI Hatim <ajdaini.hatim@gmail.com>"


# VARIABLES TEMPORAIRES
ARG APT_FLAGS="-q -y"
ARG DOCUMENTROOT="/var/www/html"



# ---------- DÉBUT COUCHE APACHE ----------
RUN apt-get update -y && \
        apt-get install ${APT_FLAGS} apache2
# ---------- FIN COUCHE APACHE ----------



# ---------- DÉBUT COUCHE MYSQL ----------
RUN apt-get install ${APT_FLAGS} mariadb-server
```

```
COPY db/articles.sql /
# ---------------- FIN COUCHE MYSQL ------------------



# ---------------- DÉBUT COUCHE PHP ------------------
RUN apt-get install ${APT_FLAGS} \
        php-mysql \
        php && \
        rm -f ${DOCUMENTROOT}/index.html && \
        apt-get autoclean -y

COPY app ${DOCUMENTROOT}
# ---------------- FIN COUCHE PHP --------------------


# OUVERTURE DU PORT HTTP
EXPOSE 80


# RÉPERTOIRE DE TRAVAIL
WORKDIR  ${DOCUMENTROOT}


# DÉMARRAGE DES SERVICES LORS DE L'EXÉCUTION DE L'IMAGE
ENTRYPOINT service mariadb start && mariadb < /articles.sql && apache2ctl -D
FOREGROUND
```

Explication du Dockerfile

Création de la couche OS basée sur l'image **debian-slim**.

```
FROM debian:stable-slim
```

Rajout des métadonnées spécifiques à l'image à créer.

```
LABEL version="1.0" maintainer="Dmalembe<dmalembe@gmail.com>"
```

docker inspect <IMAGE_NAME>

La première variable sert pour la commande apt .

La seconde variable est le répertoire de travail de apache.

```
ARG APT_FLAGS="-q -y"
ARG DOCUMENTROOT="/var/www/html"
```

Construction de la couche Apache.

```
RUN apt-get update -y && apt-get install ${APT_FLAGS} apache2
```

Téléchargement du service Mysql et rajout du fichier :

```
RUN  apt-get install ${APT_FLAGS} mariadb-server
COPY  db/articles.sql
```

Installation de l'interpréteur php et du module php-mysql.
Suppression du fichier index.html du DocumentRoot d'Apache.
Nettoyage du cache.

```
RUN  apt-get install ${APT_FLAGS} \
         php-mysql \
         php && \
         rm -f ${DOCUMENTROOT}/index.html && \
         apt-get autoclean -y

COPY  app ${DOCUMENTROOT}
```

Ouverture du port HTTP.

```
EXPOSE 80
```

Démarrage de la base de données grâce au fichier articles.sql

```
ENTRYPOINT service mariadb start && mariadb < /articles.sql && apache2ctl -D
FOREGROUND
```

Construction et Exécution de l'image.

Construction d'une image docker depuis un Dockerfile.

Syntaxe :

```
docker build -t <IMAGE_NAME> .
```

Dans notre contexte :

```
docker build -t my_lamp .
```

Exécution de l'image personnalisée.

```
docker run -d --name my_lamp_c -p 8080:80 my_lamp
```

On oeuvre ensuite l'application à l'adresse : http://localhost:8080/

Publier son image dans le Docker Hub.

Pour partager une image avec d'autre utilisateurs on peut utiliser le docker hub.

On s'inscrit sur la plateforme.

Docker Hub

Après la création du compte, on se connecte à partir de la ligne de commande :

```
docker login
```

Renseigner nom d’utilisateur et mot de passe :

On récupère l’id ou le nom de l’image :

```
docker images
```

On ajoute ensuite un tag à l’id ou au nom de l’image.

```
docker tag <IMAGENAME OU ID> <HUB-USER>/<REPONAME>[:<TAG>]
```

ou

```
docker tag my_lamp dmalembe/firstrepo_1:first_tag
```

On peut pusher l’image vers de Docker Hub :

```
docker push <HUB-USER>/<REPONAME>[:<TAG>]
```

ou

```
docker push dmalembe/firstrepo_1:first_tag
```