TP #2

Etape 0 :

- **Stopper et supprimer tous les containers existants**
  - *docker stop $(docker ps -a -q); docker rm $(docker ps -a -q)*
- **Créer un répertoire docker-tp2**
  - *mkdir docker-tp2; cd docker-tp2*
- **Initialiser un repository git dans ce répertoire**
  - *git init*

Chaque étape doit être dans un sous-répertoire nommé “etapeX” (X étant le numéro de l’étape) Il peut être utile au fur et à mesure des étapes d’organiser les fichiers dans des sous-répertoires (config, src, … par exemple). Le TP sera à rendre sous la forme d’un repository github dont vous me donnerez le lien (et les droits de consultation ;)

Etape 1 :

2 containers nommés comme suit :
- HTTP : 1 container avec un serveur HTTP qui écoute sur le port 8080
- SCRIPT : 1 container avec un interpréteur PHP (plus le protocole FPM pour NGINX)
Une page index.php qui lorsqu'elle est appelée exécute la fonction php_info() et qui sera située dans les containers dans le répertoire /app.
Décommenter et remplacer les lignes 30 à 36 par les suivantes :
location ~ \.php$ {
root /app; fastcgi_pass script:9000;
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
include fastcgi_params; }

-Nous créons d'abord un fichier index.php :

```
<?php phpinfo(); ?>
```

-Nous créons un fichier default.conf : server {

```
    listen 8080;
    server_name localhost;

    location / {
        root /app;
        index index.php;
    }

    location ~ \.php$ {
        root /app;
        fastcgi_pass script:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

-Nous créons le container script : docker run -d --name script -v "${PWD}:/app" php:7.4-fpm
-Nous créons le container http : PS C:\Users\gucav\OneDrive\Bureau\EFREI\Mastere_DE2\DevOps_MlOps\Cours2\docker-tp2\etape1> docker run -d --name http -p 8080:8080 -v "C:/Users/gucav/OneDrive/Bureau/EFREI/Mastere_DE2/DevOps_MlOps/Cours2/docker-tp2/etape1/default.conf:/etc/nginx/conf.d/default.conf" -v "C:/Users/gucav/OneDrive/Bureau/EFREI/Mastere_DE2/DevOps_MlOps/Cours2/docker-tp2/etape1:/app" --link script nginx

Test de validité de l'exercice : avec un navigateur voir le résultat de l'exécution du php_info() : http://localhost:8080/index.php
Nous avons la page par défaut php info.

Etape 2 :
3 containers nommés comme suit :
- HTTP : 1 container avec un serveur HTTP qui écoute sur le port 8080
- SCRIPT : 1 container avec un interpréteur PHP (plus le protocole FPM pour NGINX)
- DATA : 1 container avec un serveur de base données SQL (MariaDB, MySQL, PostgreSQL, ...)

Une page test_bdd.php qui lorsqu'elle est appelée va executer 2 requêtes CRUD (Request : lecture, Create Update Delete : écriture) au minimum sur le serveur SQL : 1 lecture et 1 écriture Test de validité de l'exercice : avec un navigateur voir le résultat de l'exécution de la page en retournant un résultat différent et dépendant du contenu de la base de données à chaque refresh de la page

Mise en place d'un espace de travail :
C:\Users\gucav\OneDrive\Bureau\EFREI\Mastere_DE2\DevOps_MlOps\Cours2\docker-tp2\etape2

1. **Nous allons dans cette étape créer 2 fichiers:**
- **default.conf**
- **test_bdd**

2. **Nous allons ensuite créer les 3 containers :**
   - **script : docker run -d --name script --link data -v ${PWD}:/app -w /app php:7.4-fpm bash -c "apt-get update && apt-get install -y libpng-dev && docker-php-ext-install mysqli && php-fpm" : il faut faire attention à bien installer l'extension mysqli**

   - **http : docker run -d --name http -p 8080:8080 -v ${PWD}/nginx.conf:/etc/nginx/conf.d/default.conf -v ${PWD}:/app --link script nginx**

   - **data : docker run -d --name data -e MYSQL_ROOT_PASSWORD=cavdar58 -e MYSQL_DATABASE=mydb -e MYSQL_USER=gurcu -e MYSQL_PASSWORD=cavdar58 mysql:latest**

3. **Puis nous allons créer une table test_table :**
   CREATE TABLE test_table (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(50), created_at DATETIME DEFAULT CURRENT_TIMESTAMP);

   Lorsque je teste dans le navigateur "http://localhost:8080/test_bdd.php", nous avons l'erreur suivante : Error: INSERT INTO test_table (data) VALUES ('Hello World') Unknown column 'data' in 'field list'0 results.
   → Solution ajouter la colonne data dans la table : ALTER TABLE test_table ADD COLUMN data VARCHAR(255) NOT NULL;
   Lorsque je teste dans le navigateur "http://localhost:8080/test_bdd.php" : New record created successfully
   id: 1 - Data: Hello World

Le test fonctionne bien, nous vérifions également dans la base de donnée :

```
mysql> SELECT * FROM test_data;
ERROR 1146 (42S02): Table 'mydb.test_data' doesn't exist
mysql> SELECT * FROM test_table;
+----+------+---------------------+-------------+
| id | name | created_at          | data        |
+----+------+---------------------+-------------+
|  1 | NULL | 2024-10-15 20:33:50 | Hello World |
+----+------+---------------------+-------------+
```

Etape 3 :
3 containers nommés comme suit :
- HTTP : 1 container avec un serveur HTTP qui écoute sur le port 8080
- SCRIPT : 1 container avec un interpréteur PHP (plus le protocole FPM pour NGINX)
- DATA : 1 container avec un serveur de base données SQL (MariaDB, MySQL, PostgreSQL, ...)

Remplacer la/les pages PHP simples par un package Wordpress complet.

Test de validité de l'exercice : avec un navigateur voir l'interface d'admin/installation de Wordpress afin de finaliser l'installation de celui-ci

Création des répertoires:
mkdir config
mkdir src
mkdir src/wordpress

Création des fichiers :
notepad config/Dockerfile-http

```
FROM nginx:latest

COPY ./config/nginx.conf /etc/nginx/nginx.conf
COPY ./src/wordpress /var/www/html
COPY ./config/fastcgi-php.conf /etc/nginx/snippets/fastcgi-php.conf

EXPOSE 8080
```

notepad config/nginx.conf

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    server {
        listen 8080;
        server_name localhost;

        root /var/www/html;
        index index.php index.html index.htm;

        # Gestion des fichiers PHP
        location ~ \.php$ {
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
                fastcgi_pass script-container:9000;
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            include fastcgi_params;
        }

        # Gestion des fichiers statiques
        location / {
            try_files $uri $uri/ =404;
        }
    }
}
```

notepad config/Dockerfile-script

```
FROM php:fpm

# Installer les extensions PHP nécessaires pour WordPress, y compris MySQLi
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli

WORKDIR /var/www/html
COPY ./src/wordpress /var/www/html
```

notepad config/Dockerfile-data

```
FROM mariadb:latest

ENV MYSQL_ROOT_PASSWORD=rootpassword
ENV MYSQL_DATABASE=wordpress
ENV MYSQL_USER=wpuser
ENV MYSQL_PASSWORD=wppassword

EXPOSE 3306
```

notepad config/fastcgi-php.conf

```
fastcgi_split_path_info ^(.+\.php)(/.+)$;
fastcgi_pass script:9000;
fastcgi_index index.php;
include fastcgi_params;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
fastcgi_param PATH_INFO $fastcgi_path_info;
```

Installation de WordPress :

Invoke-WebRequest -Uri https://wordpress.org/latest.tar.gz -OutFile wordpress.tar.gz
tar -xvzf wordpress.tar.gz -C ./src
rm wordpress.tar.gz

Création du réseau Docker

docker network create wp-network

Création des images & containers :

Container HTTP (NGINX)

docker build -t nginx-container -f ./config/Dockerfile-http .
docker run -d --name http-container -p 8080:8080 --network wp-network nginx-container

Container SCRIPT (PHP-FPM)

docker build -t php-container -f ./config/Dockerfile-script .
docker run -d --name script-container --network wp-network php-container

Container DATA (MariaDB)

docker build -t db-container -f ./config/Dockerfile-data .
docker run -d --name data-container --network wp-network -e MYSQL_ROOT_PASSWORD=rootpassword -e MYSQL_DATABASE=wordpress -e MYSQL_USER=wpuser -e MYSQL_PASSWORD=wppassword db-container

WordPress

Set up your database connection

Below you should enter your database connection details. If you are not sure about these, contact your host.

Database Name

wordpress

The name of the database you want to use with WordPress.

Username

wpuser

Your database username.

Password

•••••••••• Show

Your database password.

Database Host

localhost

You should be able to get this info from your web host, if `localhost` does not work.

Table Prefix

wp_

If you want to run multiple WordPress installations in a single database, change this.

Submit

Etape 4 :
Convertir la configuration de l'étape 3 en Docker Compose Test de validité de l'exercice : identique à l'étape 3

docker-compose.yml

```
version: '3'

services:
  http:
    image: nginx:latest
    container_name: http
    ports:
      - "8080:80"
    volumes:
      - ./src/wordpress:/app
      - ./config/nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - script

  script:
    build: .  # Utilise le Dockerfile pour construire l'image PHP
    container_name: script
    volumes:
      - ./src/wordpress:/app

  data:
    image: mysql:latest
    container_name: data
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: wordpress_db
      MYSQL_USER: wp_user
      MYSQL_PASSWORD: wp_password
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql

volumes:
  db_data:
```

dockerfile

```
FROM php:fpm

# Installer l'extension mysqli
RUN docker-php-ext-install mysqli
```

nginx.conf

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name localhost;

        root /app;  # Le dossier WordPress monté
        index index.php index.html index.htm;

        # Gestion des fichiers PHP
        location ~ \.php$ {
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_pass script:9000;
```

```
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            include fastcgi_params;
        }

        # Gestion des fichiers statiques
        location / {
            try_files $uri $uri/ =404;
        }
    }
}
```

Le dossier wordpress est également récupéré.