TD1
Rappels Docker

Documentation utiles

Docker compose file reference : `https://docs.docker.com/compose/compose-file/`
Dockerfile reference : `https://docs.docker.com/engine/reference/builder/`
Docker cli reference : `https://docs.docker.com/engine/reference/run/`
Docker compose cli reference : `https://docs.docker.com/compose/reference/`
Docker php : `https://hub.docker.com/_/php`
Docker mariadb : `https://hub.docker.com/_/mariadb`

1 Etape 1 : Mise en place de l'environnement de travail :

Depuis un terminal

a. Vérifier que docker est installé sur votre machine en lançant un docker simpliste.

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

Relancez une seconde fois le même container. Quelle différence observez vous dans les logs ? Que pouvez vous en déduire lorsqu'un container n'est pas dispo en local ?

b. A l'aide de la commande *git* clonez le dépôt `https://gitlabinfo.iutmontp.univ-montp2.fr/coletta/r5.a.09-virtualisation-avancee`

c. Placez vous dans le répertoire $tp1$

d. Ouvrez le fichier `docker-compose.yml` . Combien de containers différents seront lancés ? quels seront les ports exposés ?

e. Lancez le compose à l'aide de la commande :

```
$ docker compose up
```

Vous pouvez utiliser l'option -d pour le mode détaché.

f. Vérifiez que les conteneurs sont bien déployés. Dans un second terminal et dans le même répertoire :

```
$ docker compose ps
```

g. Commentez le `docker-compose.yml` pour en expliquer chaque ligne

2 Etape 2 : Test de l'application

Vous l'avez compris, ce `docker-compose.yml` se comporte comme "Wamp", il est composé d'un serveur web avec interpréteur php7.2 et d'une base mysql.

h. Où se trouvent les pages php de notre application (volontairement simpliste) ? les parcourir et déduire le rôle de chacune d'elle.

i. A l'aide de la commande curl (et/ou de votre navigateur) vous allez successivement :
— appeler la page hello world
— appeler la page de création de la base + insertion de quelques tuples
— consulter le contenu de la base

j. Eteignez les conteneurs de votre projet avec le commande :

```
$ docker compose down
```

et relancer les.

k. Consultez à nouveau la base, qu'en déduisez vous en terme de persistance de données ?

l. En utilisant la documentation de mariadb déterminer le répertoire dans lequel le serveur stoque ses données. Puis en vous basant sur l'exemple du conteneur *app* dans le *compose* rendre la base persistante.

m. Tester à nouveau l'interruption / re-lancement pour vérifier.

3 Etape 3 : Développement / Mise au point

n. L'équipe de dev souhaite pouvoir accéder au shell du conteneur app.

o. Depuis le conteneur app, à l'aide de l'utilitaire $nmap$ (que vous installerez si nécessaire) scannez les ports ouverts $database$?

p. L'équipe de dev souhaite pouvoir accéder à la base de données depuis leur IDE.
Avec $telnet$ ($nmap$ n'est pas installé sur les machines de l'IUT) tester si vous avez accès au port de $MariaDB$ depuis la machine hôte.

Si ce n'est pas le cas, changez le `docker-compose.yml` pour exposer ce port.

q. Configurer votre IDE pour extraire en quelques clics, un diagramme du schémas de la base de données. Ajouter vous comme un tuple dans la table $users$.

4 Etape 4 : Montée de version et rétro-compatibilité

r. La DSI vous impose de monter de version de PHP de 7.2 à 8.2

s. Un client vous demande si votre application serait compatible avec son serveur actuel (environnement Php 5.4 et MariaDB 10), modifiez le `docker-compose.yml` et testez.

5 Etape 5 : Changer de moteur de DB

Le Product Owner vous impose de migrer de MySQL à PostgresSQL.

t. Dans le `docker-compose.yml`, modifier la configuration du container database pour passer de mariadb à postgres

u. Modifier le code de l'application pour migrer à postgresSQL puis tester :
— Testez votre application code (Etape 2),
— Vérifiez la persistance des données (question l)
— Vérifiez les accès des dévellopeurs demandés à la question p.