TP Noté Docker (13 Novembre 2020)

Le rendu attendu est une archive (zip ou tar) contenant les sous-dossiers suivants, ainsi que tout fichiers que vous jugerez utile :

- `mediawiki` ;
- `url-shortener` ;
- `questions.pdf` ;

```
.
├── mediawiki
│   ├── LocalSettings.php
│   └── start.sh
├── url-shortener
│   ├── Dockerfile
│   ├── src
│   └── startup.sh
└── questions.pdf
```

Questions

1. Donner la commande permettant de lancer un conteneur :
    - en mode interactif (permettant d'entrer des commandes)
    - basé sur l'image `debian`
    - portant le nom `debian-interactif`
2. Donner la commande permettant de démarrer un conteneur temporaire (qui se supprime tout seul), basé sur l'image `alpine` et exécutant la commande `hostname` ;
3. Donner la commande permettant de lister les conteneurs en cours de fonctionnement ;
    1. Quelle est l'option à rajouter à la commande de listing permettant de lister TOUS les conteneurs (y compris ceux stoppés) ;
4. Donner la commande permettant d'identifier les réseaux (entre autres informations) auxquels un conteneur appartient.
5. Lorsqu'on ouvre un port lors du lancement d'un conteneur, mais qu'on ne spécifie pas de port sur la machine hôte, que se passe-t-il ?
6. Quels sont les processus en fonctionnement dans un conteneur (de façon générale) et que se passe-t-il si on "tue" le processus ayant le plus petit PID (PID 1) ?

MediaWiki

MediaWiki est un projet open source permettant de faire fonctionner un wiki. Il est notamment utilisé par Wikipédia.

Nous allons dans cet exercice configurer l'image Docker **mediawiki** existante officielle pour la faire fonctionner avec une base de donnée **MySql**.

Note La commande `docker create` permet de créer un conteneur sans le démarrer. Elle s'utilise de la même façon que la commande `docker run`.

1. Créer un fichier Bash `start.sh` permettant d'exécuter les commandes ci-dessous en une seule fois ;

1. Ajouter au fichier fichier `start.sh` la commande permettant de créer un conteneur `MySql:5.7` (ne pas le démarrer, car nous l'ajouterons plus tard à un réseau privé) :
    - nommé `mediawiki_db` ;
    - utilisant la base de donnée `mediawiki_db_name` ;
    - utilisant le nom d'utilisateur `mediawiki_db_user` avec son mot de passe associé `mediawiki_db_password` ;
    - utilisant un mot de passe pour `root` **aléatoire** ;
2. Ajouter au fichier `start.sh` la commande pour créer le conteneur Media Wiki :
    - nommé `mediawiki` ;
    - basé sur l'image `mediawiki` officielle https://hub.docker.com/_/mediawiki ;
    - ayant le **port** `8888` ouvert (sur la machine **hôte**). En interne, MediaWiki utilise le **port** `80` ;
3. Ajouter au fichier `start.sh` la commande pour créer le réseau leur permettant de communiquer
    - nommé `mediawiki_network` ;
    - étant **privé**, de façon à ce que la base de donnée ne soit accessible que depuis le conteneur `mediawiki` ;
4. Ajouter au fichier `start.sh` la commande pour **connecter** les deux conteneurs précédemment créés à ce réseau ;
5. Ajouter au fichier `start.sh` la commande pour démarrer les deux conteneurs ;

2. Exécuter le fichier `start.sh` puis accéder à la page `localhost:8888`
    - Remplir les champs demandés :
        - Sélectionner la langue Française ;
        - La page suivante indique que toutes les dépendances sont vérifiées ;
        - La page de configuration de la base de données :
            - sélectionner le type de base de données : Mysql
        - sélectionner le **nom d'utilisateur**, le **mot de passe** ainsi que le **nom de la base de données** en fonction des noms des conteneurs définis précédemment :
            - type : `Mysql` ;
            - nom d'hôte : `mediawiki_db` ;
            - nom de la base de données : `mediawiki_db_name` ;
            - utilisateur : `mediawiki_db_user` ;
            - mot de passe : `mediawiki_db_password` ;
        - sur la page suivante, utiliser le compte et le moteur de stockage proposé par défaut (InnoDB) ;
        - Configurer le nom du wiki, l'espace de nom du projet ainsi que l'utilisateur administrateur. Il n'y a pas de valeurs prédéfinies, à vous de rentrer celles qui vous conviennent.

          Sélectionner en bas de page "Ne plus poser de questions" et Continuer ;
        - Télécharger le fichier proposé `LocalSettings.php` ; Ce fichier contient les informations rentrées jusqu'à maintenant.
3. Copier le fichier `LocalSettings.php` dans le conteneur grâce à la commande `docker cp` dans le dossier de destination `/var/www/html/` ;
4. En accédant à la page `localhost:8888`, le site MediaWiki devrait être accessible.

URL shortener

Ce projet est un réducteur d'URL, permettant de raccourcir une URL en une URL plus petite. Pour cet exercice, nous utiliserons le projet : **node-url-shortener**, fonctionnant grâce à Node.js.

Base de données

1. Démarrer un conteneur basé sur l'image `redis` :
   - ayant le nom `url-shortener-redis` ;
   - lancé en mode dæmon ;
   - sauvegardant ses données dans un volume nommé `url-shortener-redis-data` (se baser sur la documentation en ligne de l'image `redis` pour trouver le chemin à utiliser dans le conteneur) ;
2. Nous n'avons pas ouverts de port, mais le conteneur l'a fait de façon automatique. Trouver grâce à la commande permettant de lister les conteneurs sur quel port il est joignable.

Applicatif

Note : pour déboguer le conteneur, vous pouvez le démarrer sur la commande bash pour exécuter manuellement des commandes, vous connecter au conteneur déjà en cours de fonctionnement ou encore visualiser les logs depuis la machine hôte.

1. Créer le dossier `url-shortener` ;
2. Télécharger les sources du projet depuis Github (https://github.com/dotzero/node-url-shortener.git) et les déplacer dans le sous-dossier `url-shortener/sources/` ;
3. Créer le fichier `Dockerfile` dans le dossier `url-shortener/`.
   1. Ajouter l'instruction permettant de baser notre image sur l'image ***debian 10***.
   2. Node.js sera nécessaire pour faire fonctionne l'application, mais Debian ne fournissant pas une version suffisamment récente, nous allons donc avoir besoin de l'installer manuellement. Pour ce faire, inclure les instructions provenant du site suivant dans le `Dockerfile` : https://github.com/nodesource/distributions/blob/master/README.md#installation-instructions.

      À noter que le paquet `curl` est nécessaire, l'installer en ajoutant dans le `Dockerfile` l'instruction correspondante.
   3. Ajouter l'instruction permettant de copier les sources applicatives depuis `url-shortener/sources` dans le dossier `/app` de l'image.
   4. Ajouter l'instruction permettant de définir le dossier de travail (voir la documentation Docker). Cette commande permet de définir que toutes les commandes suivantes seront exécutées à partir de ce dossier.
   5. Ajouter l'instruction permettant d'installer les dépendances Node.js (`npm install`) lors de la construction de l'image.
   6. Définir que la commande lancée au démarrage du conteneur sera `bash /scripts/startup.sh`.
4. Créer le fichier `startup.sh` dans le dossier `url-shortener/`.

   La commande permettant de démarrer l'applicatif est `node /app/app.js`. Cette commande peut prendre en argument plusieurs paramètre, dont l'adresse du serveur Redis via l'argument `--redis-host`.

L'objectif du script `startup.sh` est de configurer la manière dont l'applicatif va démarrer. Si la variable d'environnement `REDIS_HOST` est présente, nous lancerons la commande de démarrage de l'application (`node /app/app.js`) avec le paramètre `--redis-host` qui aura pour valeur le contenu de la variable d'environnement `$REDIS_HOST`. Sinon, on lancera cette même commande sans paramètre, ce qui aura pour effet d'utiliser les valeurs par défaut.

5. Ajouter l'instruction au fichier `Dockerfile` permettant de copier le script `startup.sh` dans le dossier `/scripts/startup.sh`.
6. Donner la commande permettant de construire l'image sous le nom `url-shortener` et l'exécuter.
7. Donner la commande permettant de démarrer le conteneur et l'exécuter :
    - basé sur notre image `url-shortener` ;
    - exposant le port **3000** du conteneur sur le port **3000** de l'hôte ;
    - ajoutant la variable d'environnement `REDIS_HOST` avec pour valeur le nom du conteneur de base de données Redis ;

Connexion des deux conteneurs

1. Créer le réseau `url-shortener`, utilisant le sous-réseau `10.0.50.0`.
2. Connecter le conteneur de base de données `url-shortener-redis` à ce réseau.
3. Connecter le conteneur applicatif `url-shortener` à ce réseau.
4. Vérifier grâce à une commande docker que les deux conteneurs sont bien connectés au réseau `url-shortener`.
5. Accéder à la page `http://localhost:3000` et vérifier que tout fonctionne bien.

Lorsqu'une URL est "raccourcie", l'application affiche un message de succès. En accédant à l'URL, vous serez redirigé vers la page originale.