Admin Infra : Exercices

1 Préliminaires

Pour cette séance, nous continuons d'utiliser notre Debian 10 sur Azure.

Lorsqu'un mot de passe est demandé, nous vous conseillons de mettre « azerty1.» par facilité.

2 Exercice : Manipuler Docker

Référez-vous à la section 10.9 Docker- conteneurs du syllabus. Lisez-bien les indications utiles ci-dessous avant de commencer l'exercice.

Nous continuons à construire l'infrastructure de notre société ITDEV. Celle-ci souhaite que ses développeurs utilisent des images docker pour leur développement afin de faciliter les déploiement, l'intégration continue et les tests.

1. Installer Docker sur votre machine Linux (Voir Syllabus section 10.9 Docker)
2. Créer une image Docker avec Apache et le syllabus HTML (HTTP) via un Dockerfile
    1. Cette image sera basée sur debian(latest)
    2. Cette image utilisera apache2
    3. Remplacer le contenu du fichier 000-default.conf présent dans /etc/apache2/sites-available par votre propre VirtualHost. Dans un conteneur nous ne déployons qu'un seul site donc autant que ce soit le site par défaut et de plus vous ne devrez pas l'activer (a2ensite).
    4. Démarrer apache2
        1. Attention par défaut un conteneur s'arrête lorsqu'il a effectué tous ses actions
        2. Regarder sur Internet pour que votre conteneur tourne en permanence. Ainsi Apache pourra répondre aux requêtes HTTP. Rechercher «How to start apache2 automatically in docker »
3. Lancer votre conteneur
    1. Vous ferez un mapping de port (port forwarding) entre votre machine hôte(8090) et le conteneur (80). Ainsi le site HTML sera accessible depuis votre machine hôte sur le port 8090
    2. Tester le résultat sur votre machine hôte : « lynx localhost:8090 »
    3. Tester le résultat sur votre machine physique depuis un navigateur. Vous avez normalement maintenant toutes les connaissances nécessaires pour y arriver !

2.1 Indications utiles

Il n'est pas nécessaire d'utiliser les volumes pour cet exercice.

Les étapes pour cet exercice sont les suivantes : création du Dockerfile, création de l'image (docker build), création du conteneur (docker run).

Si vous avez des problèmes lors du « docker build », cela signifie que des erreurs sont présentes dans le Dockerfile. Vérifiez bien votre fichier (path, noms, …)

Si vous avez des problèmes lors du « docker run » ou lors des tests, n'oubliez pas que vous pouvez vous connecter à l'intérieur du conteneur pour vérifier/tester différentes choses.

Il est utile de supprimer les conteneurs et images problématiques pour plus de lisibilité.

Vous pouvez vous connecter plusieurs fois en SSH pour avoir plusieurs terminaux à votre disposition.

3 Exercice : 2 conteneurs

Pour ce deuxième exercice, nous allons nous rapprocher d'une architecture micro-service. L'idée est de déployer le site des bonnes nouvelles(site PHP-MySQL) via 2 conteneurs. Le premier conteneur contiendra la base de données tandis que le deuxième conteneur contiendra le serveur Web (apache) et le code source. Nous allons également tirer profit du Docker Hub en utilisant des images déjà optimisées et prêtes à l'emploi. **Lisez-bien les indications utiles ci-dessous avant de commencer l'exercice.**

1. Créer l'arborescence suivante :
    - sitePHPDocker
        - contdbphp
            - Dockerfile
            - bdbn.sql (fichier de création de la DB disponible sur Moodle)
        - contsitephp
            - Dockerfile
            - siteBonneNouvelles (site php disponible sur Moodle)
            - sitePHP.conf (vhost disponible sur Moodle)

2. Pour le Dockerfile « contdbphp »
    1. Utiliser l'image « mysql »
    2. Afin de créer automatiquement la DB à partir du fichier bdbn.sql, renseignez-vous sur le Docker Hub (image mysql) ou éventuellement sur Internet

3. Pour le Dockerfile « contcodephp »
    1. Utiliser l'image « php:7.2.8-apache-stretch »
    2. Renseignez-vous sur l'image pour intégrer le driver pdo_mysql
    3. Copier le vhost (sitePHP.conf)
    4. Copier les sources du site
4. Lancer vos 2 conteneurs
    1. Vérifier que le sitePHP fonctionne (menu « La genèse »)
        1. Si ceci fonctionne, cela veut dire que votre 1re conteneur (contsitephp) est opérationnel
        2. Vérifier que le sitePHP fonctionne (menu « Les livres»)
            1. Ceci utilise la connexion avec la base de données située sur un autre conteneur
            2. Il sera donc nécessaire de créer un réseau entre les 2 conteneurs
            3. la connexion avec la base de données s'effectue dans le fichier models/Db.class.php, il sera nécessaire de le modifier
            4. Regardez bien le message d'erreur de cette page
                1. erreur driver → problème intégration PDO
                2. erreur de connexion → problème réseau

3.1 Indications utiles

Il n'est pas nécessaire d'utiliser les volumes pour cet exercice.

Nommer clairement vos images (docker build -t) et conteneurs (docker run --name) !

Suggestion :

- imgdbphp → contdbphp
- imgsitephp → contsitephp

La difficulté de cet exercice réside essentiellement dans la communication entre les 2 conteneurs. Pour déboguer, n'oubliez pas que vous pouvez vous connecter aux conteneurs en fonctionnement afin de les modifier (fichiers, …) afin de valider votre solution. Lorsque votre solution est validée, vous pouvez réfléchir à comment intégrer le fruit de vos résultats au Dockerfile.