SAE 23 Mettre en place une solution informatique pour l'entreprise

TP n°2 Créer son Image « Portfolio » avec Docker

Vous effacerez les conteneurs que vous avez créé ainsi que les images que vous avez téléchargées avant de partir.

Objectif :

Après vous être familiarisé avec la création de conteneur dans le TP précédent, vous allez maintenant apprendre à créer vos propres images. Vous verrez également stocker ces images dans votre dépôt, automatiser la création de conteneur et gérer des volumes pour stocker vos données en dehors des conteneurs.

Compte rendu :

Vous ferez état de votre travail dans cette SAé dans le blog WordPress.

Vous ferez figurer notamment :

- Le tableau de synthèse de l'avant dernière question à remplir au fur et à mesure,
- le dockerfile créant l'image Wordpress,
- les parties bonus si vous les avez faites.

N.B : Une partie bonus est disponible dans un autre fichier si vous êtes en avance.

Apprentissages critiques ciblés par ce TP :

- RT3 :
  - AC0311 | Utiliser un système informatique et ses outils
  - AC0312 | Lire, exécuter, corriger et modifier un programme
  - AC0313 | Traduire un algorithme, dans un langage et pour un environnement donné
  - AC0314 |Connaître l'architecture et les technologies d'un site Web
  - AC0315 |Choisir les mécanismes de gestion de données adaptés au développement de l'outil
  - AC0316 |S'intégrer dans un environnement propice au développement et au travail collaboratif

Sommaire

1 Introduction

Dans le précédent TP vous avez utilisé des images pour créer des conteneurs basés sur ces images. Par exemple vous avez utilisés deux conteneurs nginx pour stocker 2 sites web différents (1 par conteneur). Docker ne stockera que la différence entre l'image de base et le conteneur soit le site web dans notre exemple.

Vous avez pu voir qu'il existait beaucoup d'images sur le site docker hub qui permet de stocker, distribuer et partager des images (base de données mongodb, mariadb, mysql, rocketchat, …). Nous allons maintenant voire comment vous pouvez créer et redistribuer vos propres images. Pour créer ces images, on part d'une image de base linux où l'on va installer le ou les services que l'on veut fournir. On sauvegarde ensuite l'image ainsi crée. L'image pourra alors être utilisée pour créer un ou des conteneurs que l'on pourra personnaliser.

Dans ce TP, nous allons construire une image contenant votre portfolio. Nous mettrons à disposition l'image du site à votre voisin.

Certains passages sont guidés, dans d'autre c'est à vous de faire le travail. Seuls les passages où vous devrez faire le travail sont évalués.

2 Synthèse

Ce tableau est à remplir au fur et à mesure de votre avancement et à mettre dans votre blog :

Action	Commande
Créer une image d'après un conteneur	
Créer un dépôt pour stocker vos images	
Stocker une image sur un dépôt privé	
Lister les images disponibles sur un dépôt	
Construire une image d'après un Dockerfile	
Dockerfile : bâtir d'après image xxxx	
Dockerfile : copier un fichier	
Dockerfile : Exécuter une commande	
Dockerfile : Exécuter une commande après avoir crée le conteneur	
Dockerfile : Expose un port	
Créer les conteneurs d'après un fichier docker-compose	
Dockercompose : même chose et revenir à la ligne de commande	
Stopper les conteneurs crée par dockercompose	

3 Création d'une image docker

Nous allons voir deux méthodes pour créer cette image : en convertissant un conteneur en image ou en créant un fichier Dockerfile décrivant comment construire cette image à partir d'une série d'instructions. La première méthode est plus simple à mettre en œuvre mais la deuxième est la plus recommandée.

3.1 Création d'une image à partir d'un conteneur

Comme dans le 1er TP créez un conteneur mon_web basé sur l'image webdevops/php-nginx et reliez le port 80 du conteneur au port **90** de votre ordinateur (le 80 est déjà utilisé par Wordpress).

Personnalisez votre page d'accueil à l'aide de la commande (mettre votre nom au lieu de NOM) :

```
docker exec -ti mon_web sh -c 'echo \"Bienvenue chez NOM\" > /app/index.php'
```

Il est important pour la suite de bien comprendre cette commande :

- `docker exec` : exécute un commande dans le conteneur
- `-ti` : mode terminal interactif
- `mon_web` : nom du conteneur
- `sh -c 'echo \"Bienvenue chez NOM\" > /app/index.php'` : exécute la commande en orange avec le shell Bourne (option -c pour exécuter la commande entre apostrophes)
- `echo \"Bienvenue chez NOM\" > /app/index.php` : on écrit Bienvenu chez NOM dans le fichier /app/index.php (redirection). Notez \" pour indiquer que c'est le caractère " que l'on utilise et que ce n'est pas le symbole délimitant une chaine de caractère.

Tester le site web (port 90) – bien indiquer le fichier index.php sinon cela ne marche pas.

Nous allons transformer ce conteneur en image pour pouvoir le réutiliser indéfiniment pour cela on utilise la commande **`docker commit`** :

```
docker commit mon_web im_web
```

Lister les images avec docker images et vous verrez alors apparaître im_web.

Supprimez le conteneur et recréez-le à partir de l'image.

```
docker stop mon_web
docker rm mon_web
docker create --name mon_web2 -p 90:80 im_web
docker start mon_web2
```

Vous avez maintenant une image permettant de créer votre conteneur personnalisé avec votre propre site web.

Cette méthode est assez pratique pour créer des images mais ne permet pas de conserver dans un fichier la méthode qui nous a conduit à créer cette image. On va donc montrer une deuxième méthode basée sur un fichier « Dockerfile » décrivant les opérations pour créer une image.

3.2 Création d'une image à partir d'un dockerfile

Pour créer une image à partir d'un dockerfile, créez un répertoire rep_web_nom (remplacer nom par votre nom) dans lequel vous mettre un fichier photo.jpg correspondant à votre photo et un fichier nommé **`Dockerfile`** (pas d'extension) que vous éditerez avec le VisualStudio Code, le bloc note ou NotePad++.

Le Dockerfile est un fichier qui décrit comment construire l'image. Il est composé d'une série de commandes parmi lesquelles :

- FROM xxxxx qui indique que l'on construit le docker à partir de l'image xxxxx,
- COPY a b permet de copier le fichier a dans notre répertoire en b dans le conteneur,
- RUN xxxxx qui exécute la commande xxxxx lorsque l'on construit l'image,
- CMD xxxx qui exécute une commande lorsque l'on construit le conteneur (utilisé par exemple pour démarrer des services),
- EXPOSE nn mm qui indique que les ports que le conteneur écoute (utile pour la documentation).

Dans notre exemple, nous allons mettre dans le fichier Dockerfile :

```
FROM webdevops/php-nginx

COPY photo.jpg /app/photo.jpg
RUN echo "<p>Coucou c'est NOM</p>" > /app/index.php
RUN echo "<img src=\"photo.jpg\">" >> /app/index.php

EXPOSE 80
```

Le >> permet d'écrire à la suite d'un fichier (on n'efface pas ce qu'il y avait avant mais on compléte). Le > efface l'ancien contenu.

Placez-vous en dehors du répertoire rep_web_nom et utiliser la commande **`docker build`** :

```
docker build -t img_web_nom rep_web_nom
```

Vous avez maintenant une image img_web_nom permettant de créer un conteneur correspondant à votre site personnalisé. L'option -t signifie tag. Le tag (marquage) est le nom de l'image.

Créez un conteneur correspondant à cette image (port 80 -> 90) et vérifiez que cela marche avec votre navigateur.

Stoppez puis supprimer le conteneur. Conservez votre image.

4 Publication d'une image docker sur un dépôt privé

Vous avez téléchargé vos images docker à partir du docker hub avec la commande ***`docker pull`****. Malheureusement, vous ne pouvez pas télécharger (pour l'instant) l'image qu'à créer votre voisin car elle n'est pas stockée dans un dépôt (le docker hub est un dépôt d'image). Afin de partager l'image que vous avez créé avec votre voisin, nous allons créer notre propre dépôt privé pour que vous puissiez y publier votre image et la rendre accessible à votre voisin.*

4.1 Installation du dépôt privé

Récupérez l'image registry à partir du docker hub. Regardez dans la documentation du docker hub comment démarrer le conteneur registry et l'associer au port 5000 (1 ligne).

4.2 Stockage de l'image

Pour stocker l'image, vous devez lui ajouter une marque (tag) afin de l'associer à votre registry privée :

```
docker tag img_web_nom localhost:5000/img_web_nom
```

Si vous listez les images, vous verrez qu'une nouvelle image est apparue.

On demande ensuite à docker de sauvegarder l'image dans le dépôt grâce à la commande push :

```
docker push localhost:5000/img_web_nom
```

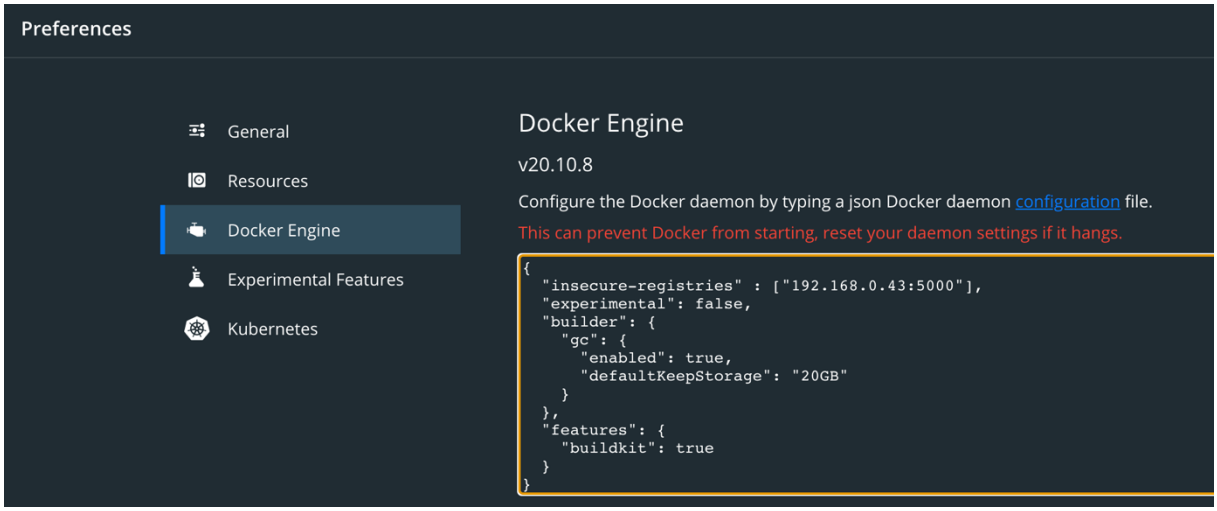
4.3 Récupération d'une image sur un registre privé

Nous allons maintenant essayer de récupérer une image stockée sur le dépôt de votre voisin. Dans un premier temps, il faut autoriser l'utilisation de l'adresse IP de votre voisin (sinon vous aurez un message `Error response from daemon: Get "https://votreIP:5000/v2/": http: server gave HTTP response to HTTPS client`).

Dans le docker hub allez dans le menu préférence puis docker engine.

Ajoutez les lignes suivante en remplaçant myregistrydomain.com par l'adresse IP de votre voisin:

```
{
  "insecure-registries" : ["myregistrydomain.com:5000"]
}
```



Cliquez sur Apply and Restart.

P.S. : Sur Linux, il faut éditer/créer le fichier **`/etc/docker/daemon.json`** puis taper les commandes `systemctl daemon-reload` suivie de `systemctl restart docker` pour redémarrer docker et prendre en compte les changements.

Vous pouvez ensuite utiliser la commande pull pour récupérer l'image de votre voisin sur son registry en replaçant IPvoisin par l'adresse IP de votre voisin :

```
docker pull IPvoisin:5000/img_web_nomvoisin
```

Créez un conteneur correspondant à l'image que vous avez récupéré (n'oubliez pas de publier le port 80) et visualisez le site web associé. Vous devez voir l'image de votre voisin.

Pour consulter le contenu de votre registry, vous pouvez ouvrir un navigateur à l'adresse http://127.0.0.1:5000/v2/_catalog

5 Création d'une image Portfolio

C'est à vous : Vous nous rendrez le dockerfile que vous créerez dans cette partie.

En parcourant le Docker Hub, vous vous rendrez compte qu'il manque une image essentielle : une image permettant de voir votre site web votre portfolio ! Vous allez donc créer un dockerfile pour y remédier.

5.1 Recensement des commandes

Pour créer votre image portfolio, nous devons tout d'abord répertorier les commandes à rentrer dans le Dockerfile. C'est pourquoi nous allons dans un premier temps créer un conteneur debian où symfony tournera sur le port 80 dans le conteneur. Le port 80 du conteneur sera associé au port 82 sur le PC (le 80 étant déjà pris par WordPress) :

```
docker run -ti --name test -p 82:80 debian /bin/bash
```

On voit ici que l'on peut demander d'éxécuter la commande /bin/bash avec la commande run comme on fait avec la commande exec.

Nous sommes dans un terminal debian. Vous noterez dans un fichier toutes les commandes que vous taperez pour installer symfony (rassurez-vous les étapes sont indiqués ci-dessous).

Sauf pour l'étape 1, nous utiliserons l'option y :

```
apt install xxx -y (où xxx est le paquet à installer)
```

Cette option permet de ne rien demander à l'utilisateur. Ce sera important lorsque ces commandes seront exécutées automatiquement dans un dockerfile où vous ne pouvez rien taper.

1. Mettre à jour la base de données apt : apt update

Installation de apache php mysql

2. Installer le paquet apache2 (c'est celui qui s'appelait httpd en centos)
3. Installer le paquet php
4. Installer les paquets fournissant des fonctions optionnelles à php : php-xml php-mbstring php-intl php-mysql

Installation de composer

5. Installer le paquet zip
6. Aller sur le site composer cliquer sur download : vous verrez les commandes pour installer composer. Je vous donne une plus version adaptée ci-dessous :

```
php -r "copy('https://getcomposer.org/installer', '/root/composer-setup.php');"
php /root/composer-setup.php --install-dir="/usr/bin"
```

Installation de git et récupération de votre site

7. Installer git
8. Clonez votre dépôt git portfolio dans le répertoire /var/www/html/site à l'aide de la commande :

   `git clone https://github.com/xxxx/yyy.git /var/www/html/site` (à adapter)
9. Placez-vous dans le répertoire /var/www/html/site

Installation des dépendances

10. Demander à composer de télécharger les dépendances de votre site (les dépendances sont les fichiers nécessaires pour faire tourner votre site) :

    `composer.phar -n update && composer.phar -n install`

Configuration de apache

Nous n'avons pas installé l'utilitaire symfony qui vous permettait de lancer votre site web : c'est normal, nous allons utiliser apache comme serveur web et pas symfony.

11. On peut installer les paquets lynx (navigateur web en mode console) et nano (éditeur).

12. On donne le contenu du fichier site.conf à mettre dans le répertoire `/etc/apache2/sites-available/site.conf`

    Rajouter des commentaires dans ce fichier avec # pour expliquer chacune des lignes en vert

```
<VirtualHost *:80>
      ServerAdmin mettre.votre.email@univ-st-etienne.fr
      ServerName localhost
      DocumentRoot /var/www/html/site/public

      <Directory /var/www/html/site/public>
          AllowOverride None
```

```
            Order Allow,Deny
            Allow from All

            <IfModule mod_rewrite.c>
                RewriteEngine On

                RewriteCond %{REQUEST_URI} \..+$
                RewriteCond %{REQUEST_URI} !\.html$
                RewriteRule .* - [L]

                RewriteRule ^(.*)$ index.php [QSA,L]
            </IfModule>
        </Directory>
    </VirtualHost>
```

L'utilisation des RewriteRule pour symfony est expliqué sur le site : https://symfony.com/legacy/doc/cookbook/1_2/fr/web_server

13. Contrairement à Centos où il suffisait de redémarrer le démon httpd dans debian il faut activer site.conf et désactiver le site par défaut. On active ensuite les rewrite rules.

```
a2ensite site.conf
a2dissite 000-default.conf
a2enmod rewrite
```

14. On utilise `/usr/sbin/service apache2 start` pour démarrer apache (systemctl n'est pas installé)

5.2 Ajustement du site web

Vérifiez que vous voyez bien votre site sur le port 82.

Si vous voyez l'interface de symfony c'est que vous n'avez pas définit de route pour « / ». Dans ce cas vous pouvez, dans le contrôleur rajouter à l'intérieur de la classe (en remplaçant xxxx par le nom de la route associée à l'accueil) :

```
/**
 * @Route("/", name="root")
 */
public function root(): Response
{
        return $this->redirectToRoute("xxxx");
}
```

Poussez votre modification éventuelle dans git.

5.3 Réalisation du Dockerfile

Créez un Dockerfile qui va réaliser les opérations que nous venons de réaliser à la main automatiquement.

Par rapport à l'exemple, vous utiliserez les commandes supplémentaires ci-dessous :

- WORKDIR qui permet de définir changer le répertoire où l'on travaille (permet de réaliser l'étape « Placez-vous dans le répertoire »),

Remplacer **`apt install xxx -y`** par la combinaison **`apt update && apt install xxx -y`** pour faire une mise à jour de la base apt avant chaque installation (*La commande && permet de faire 2 actions sur la même ligne. Lorsque l'on crée une image à partir d'un dockerfile, il existe un mécanisme de cache (mémorisation des résultats) qui évite de refaire une commande déjà rentrée. Dès lors, si l'on écrit apt-update au début, il ne sera plus jamais exécuté si l'on modifie le DockerFile et que l'on refait un build plus tard car il se trouve dans le cache. Ici, on impose*

de la faire pour chaque nouvelle installation de paquet que l'on viendrait ajouter dans le dockerfile car la ligne est différente pour chaque paquet (update + install avec le nom du paquet).),

- CMD /usr/sbin/service apache2 start & bash à la fin pour exécuter ces commandes lorsqu'un conteneur aura été crée d'après l'image.

Vous mettrez un fichier site.conf dans le même répertoire que le dockerfile. Vous recopierez ce fichier dans l'image.

Construisez une image à partir du DockerFile.
Construisez un contenir à partir de cette image.
Testez.
Publiez l'image sur votre dépôt privé créer un conteneur à partir de l'image portfolio de votre voisin.

Créer un fichier portfolio.ps1 qui construit l'image et crée un conteneur qui vous placerez sur le port 82 et sur le réseau sae23_votrenom crée dans le précédent TD.

Attention : le port exposé dans l'image sera bien le 80 (commande EXPOSE du Dockerfile). C'est avec la commande -p que vous ferez correspondre le port 80 du conteneur crée d'après l'image au port 82 du PC.

Pour lancer le conteneur utiliser -tid pour empêcher le conteneur de s'arrêter après son lancement.

Vous avez fini ce TP, mais si vous êtes en avance vous pouvez faire la partie en bonus sur la thématique de la cybersécurité et qui vous montrera surtout comment utiliser docker compose.
