Gonzalez Montes Nathan, Pombo Dias Miguel

Descripción breve

[Dibujar su lector con un resumen de la participación. Normalmente es un breve resumen del documento.   
Cuando esté listo para agregar contenido, haga clic aquí y empiece a escribir.]

Projet res http

[Subtítulo del documento]

# Introduction

# Partie 1

D’abord, il faut créer in repo pour pouvoir réaliser le laboratoire. Pour ceci, il y a la possibilité d’en créer un nouveau ou de juste fork celui fournit par le professeur et après cloner le répertoire dans un dossier avec le laboratoire à faire. (On a fait une branche appelée appacheStatic pour continuer le laboratoire, erreur de frappe avec le double p).

Juste après, on crée un dossier qui nous permettra de stocker les dossiers nécessaires pour pouvoir faire notre laboratoire. Dans ce cas on a appelé ce dossier "apache-images", dedans, un dossier "apache-php-image" qui contiendra notre fichier Dockerfile. Pour avoir les documents nécessaire pour créer l’image Docker, on va les chercher sur internet, dans la page web de hub.docker.com et ici d’abord on cherche ceux de httpd (apache) et après ceux de PHP et on utilise une des images déjà existante (php:7.0-apache est celui que l’on utilise) et on copie les ligne suivantes dans notre Dockerfile pour les utiliser :

* FROM php:7.0-apache (de où on prend les fichiers)
* COPY content/ /var/www/html/ (où vont être stockées les fichiers dans "/var/www/html/", "content/" dans notre cas)

Pour tester que l’on a bien pris le bon endroit avec l’image Docker, on va run le server sur notre machine avec la commande docker run -d -p 9090:80 php:7.0-apache (avec 9090:80, on dit de se mettre à l’écoute en mappant les information sur le port 9090 au lieu du port 80).

On teste en essayant de nous connecter sur le serveur avec la commande ‘telnet adresseIP port’, et dans notre cas se serait l’adresse IP de notre docker qui est 192.168.99.100, et le port que l’on a choisi pour mapper, donc 9090. On essaie de lancer une requête http avec un GET / HTTP/1.0 et si on reçoit une réponse, ça veut dire qu’il y a bien un serveur apache qui a été lancé.

Avec la commande "docker ps" on regarde que le serveur a bien été lancé, et on utilise la commande "docker inspect nom\_container" (le nom apparait sous la partie NAMES dans le terminal après avoir fait la commande "docker ps") et on regarde quelle est l’adresse IP de l’image tout en bas. (IP du docker php après le lancer : "172.17.0.2").

Vu que l’on ne sait pas exactement ce qu’il y a dans l’image docker de PHP (c’est comme une boîte noire) et que l’on veut explorer les fichiers de configuration, on peut utiliser la commande suivante : 'docker exec -it nom-container /bin/bash' pour pouvoir accéder et modifier les fichiers de la page web (on peut voir que la ligne de commande a changé et qu’on est dans "/var/www/html" de l’image). Dans l’exemple, on met dans un fichier "index.html" le texte "Coucou" avec la commande ‘echo "Coucou" > index.html’. Si on ne spécifie pas un nom de fichier dans un dossier avec cette commande, apache va chercher le fichier index.html et le retourner. On met donc du contenu texte dans notre fichier index.html et on essaie de se connecter, on verra ce contenu sur la connexion de notre browser en écrivant dans un nouvel onglet : "192.168.99.100:port\_utilisé". Dans notre cas, ce serait le port 9090. (FAIRE LES MODIFICATIONS A L’EXTERIEUR DU CONTAINAIRE POUR NE PAS PERDRE LES MODIFICATIONS, si on se déconnecte ou que l’on kill le serveur, les modifications seront perdues). On peut faire des tests en créant des dossiers/fichiers et pour accéder à ces modifications, on peut rajouter les chemins sur le browser. Par exemple, si on crée le dossier modif et que l’on crée un fichier appelé modif.html, on pourra voir son contenu sur le browser en écrivant sur l’onglet : "192.168.99.100:9090/modif/modif.html", et on pourra afficher ce qu’on a introduit dans ce fichier. Le texte introduit, étant en html, il est possible d’introduire du texte personnalisé en utilisant des blaises de html avec ce que l’on veut comme personnalisation et l’afficher dans le browser.

"Apache.config" est le fichier de configuration principal ; les fichiers "sites" sont des fichiers propres à Ubuntu ; il y a aussi des sous-fichiers de configuration possibles pour des "virtual hosts" dans le dossier "/etc/apache2/sites-avaliable", et normalement il y en a déjà un. (etc -> dossier de configuration). Si on a dans notre Dockerfile une ligne comme celle prise dans le site web mais cette fois avec "COPY configuration/ /etc/apache2/sites-avaliable/mywebsite/", on mettra nos propres configurations dans notre site et au moment de build l’image, on mettra ces informations directement dans le dossier de base de /etc/apache2/sites-avaliable.

S’il y a des requêtes qui viennent, elles seront envoyées dans le dossier "/var/www/html/" (contenu), si on possède plusieurs pages web différentes, on possèdera plusieurs fichiers html. Si on change le chemin de nos fichiers et de notre html (au lieu de les avoir dans celle de "var/", et on a un autre répertoire), il faut changer le répertoire dans le Dockerfile.

Quand on a fini de faire des modifications avec la ligne de commande, on créera le dossier "content/" que l’on à précédemment écrit dans le Dockerfile (ceci va dépendre du nom de dossier voulu), on va créer donc notre propre fichier index.html et on va mettre une peu de texte de test pour pouvoir tester ce fonctionnement. Avec la commande "docker build -t nom\_de\_l’image\_voulue .", on copiera le contenu de "content/" dans le répertoire "/var/www/html/", et avec le point à la fin de la commande, on dit que l’on veut utiliser le répertoire courant pour pouvoir y créer notre image docker.

Si on relance l’image avec "docker run -p 9090:80 nom\_donne\_a\_l’image", on pourra voir le message écrit dans notre fichier indext.html. Si par exemple on recharge la page, on aura des affichages sur la ligne de commande dans le LOG Apache, et on peut voir par exemple que l’on a reçu une requête GET, qui est la suivante : "192.168.99.1 - - [04/Jun/2017:14:56:31 +0000] "GET / HTTP/1.1" 200 431 "-" " (Réponse avec 200, ce qui veut dire bonne requête), le slash après le GET est l’URL visée par la requête et ce qu’il y a après la réponse du 200, il y a le user agent (type de navigateur) "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36". (Si on shut down la connexion, on ne sera pas capable de se connecter à nouveau sur le page).

On peut lancer à nouveau le container avec l’option "-d" juste avant l’option "-p" pour pouvoir lancer le serveur en arrière-plan. Si on se déconnecte du serveur et que l’on essaie de le relancer sur le même port, il apparaitra une erreur car déjà mappé sur le port 9090, donc on peut changer le port à mapper et avoir un deuxième serveur sur un autre port et donc changer le port avec toujours la même adresse IP précédente.

Après ces manipulations, on peut aller chercher sur internet des templates de page web pour personnaliser notre page avec le site <https://startbootstrap.com/template-cactegories/one-page/> (J’ai pris celui de <https://blackrockdigital.github.io/startbootstrap-agency/>). Après avoir téléchargé les dossiers de cette page web, on reconstruit le container de l’image modifiée comme avant avec la commande de build. Si on regarde sur le browser les containers lancés auparavant, on verra que rien ne change car il n’y a pas de lien dynamique avec les modifications que l’on a faites et que l’on vient de reconstruire sur notre image. Donc il faut arrêter un serveur (celui sur 9090 par exemple), on lance à nouveau le serveur avec la commande run et on vérifie le fonctionnement (on peut aussi voir que le container lancé sur un autre port que le 9090 n’est pas modifié et on prouve qu’il n’y pas de lien dynamique). On peut regarder sur le browser plus ou moins ce que l’on voudrait modifier et on peut se mettre à modifier ce qu’on veut où on veut dans le fichier index.html (comme avant) dans le dossier téléchargé, pour personnaliser le site web et on le lance sur un browser pour vérifier les modifications. Comme c’est bien structuré, on peut facilement visualiser les endroits où se trouvent chaque information ou donnée de la page web et ainsi les modifier plutôt facilement pour la personnaliser à notre goût.

# Partie 2