Projet cloud

Mise en place d'une architecture j2ee avec des containers docker avec gestion du load-balancing

Table des matières

Introduction

Remarque : Pour le lancement lire le fichier README du dossier docker

Creation des images

Une image docker représente l'image d'un système d'exploitation qui peut être lancé en plusieurs instances encore appelé containers. Il existe 2 manières d'avoir une image docker dans le dépôt local de son ordinateur. La première consiste à faire un « docker load » i.e. charger une image dans un dépôt sur le docker hub (le dépôt en ligne des images docker) ou un fichier image docker. La deuxième consiste à utiliser « compiler » un fichier Dockerfile contenant les instructions sur la manière de construire l'image. Quelle solution chois ?

Si nous choisissons la première cela implique que nous allons avoir une image de base dans notre machine, lancer un container de cet image, installer les logiciels, effectuer les modifications et ensuite faire un « commit » i.e. créer une image possédant les propriétés du container. Ensuite uploader cette image dans un dépôt sur dockehub. L'inconvénient évident dans notre cas c'est que la plupart du temps l'image résultante pèse près d'un giga octet et le débit de la connexion internet n'étant pas satisfaisante de notre côté, cela demande beaucoup de patience. Nous avons donc opté pour la deuxième solution.

En ce qui concerne la deuxième solution, il faut décrire le dockerfile qui permet de construire l'image à partir d'une image de base existant en local ou qui seront téléchargé sur internet ainsi que tous les logiciels à installer. Cela nécessite une très bonne connexion de votre côté, ce dont vous nous avez assuré. Toutes nos images de base sont des ubuntu à l'exection de l'image de tomcat qui est construit à partir de l'image officiel de la jre8.

Nous avons trois images à contruire : une contenant le serveur apache, une pour le serveur tomcat et une autre contenant le serveur de BD qui dans notre cas est un serveur MySQL.

Dockerfile de apache

Il part d'une image ubuntu (latest), fait une mis à jour et installe les paquets de apache2 et mod-jk. Ensuite expose le port 80, finalement définit l'Entrypoint, commande à executer au lancement d'une instance de l'image lorsqu'on ne lui passe aucuns paramètres, qui dans notre cas est une commande qui permet de lancer le service apache2 en démon.

Dockerfile MySQL

L'image est construit aussi à partir d'un ubuntu (latest) aussi, fait des mis à jour. Puis télécharge mysql et l'installe, le configure à partir du fichier … .sh., puis on expose le port 3306 et on défint l'Entrypoint qui ici consiste à lancer le service MySQL en tant arrière plan comme démon. Le script de configuration définit entre autre : des variables d'environnement sur notre utilisateur de la bd, ses privilèges, le chemin du script de création des tables.

Dockerfile Tomcat

Nous avons choisis d'intaller l'image de tomcat8 fonctionnant avec la jre8. Nous avons donc recupéré le Dockerfile nécessaire sur le dockerhub et on l'a customisé. L'image part d'une image officiel de la jre8, fait des mis à jour et installe tomcat. Ce que nous avons rajouté c'est le téléchargement de l'application annuaire depuis google drive et son dépôt dans le dossier webapps de tomcat. Bien sûr le lancement d'une instance de l'image sans paramètres lance aussi le service tomcat.

Lancement des containers de l'architecture j2ee

Architecturej2ee.jpg

L'architecture j2ee est constitué d'un serveur apache derrière lequel tourne un ou plusieurs serveurs tomcat qui accèdent à un sgbd mysql tournant sur un serveur lui aussi. Chaque serveur tourne sur un container docker. Pour lancer cet architecture nous avons écrit un script containers.launch.sh qui prend en paramètres un nombre entier contenant le nombre de serveurs tomcat à lancer. Exemple : $sudo sh containers.launch.sh 3

Le script commence par arrêter et supprimer tous les containers qui ont un nom comme ceux de nos containers qui commencent tous par « enspy_ ». Ensuite il lance le container serveur apache (enspy_apache_server), lance en deuxième position le container du serveur mysql (enspy_db_server), fait une boucle pour lancer tous les containers tomcat. Chaque lancement d'un container tomcat comprend des paramètres permettant de le lier au container apache et au container mysql. Tout au long du script l'on construit le fichier de configuration qui servira au load-balancing.

Le load-balancing

Le load-balancing consiste à équilibrer les requêtes sur l'application annuaire à tous les serveurs tomcat en passant par le serveeur apache. Pour ce faire nous utilisons le connecteur mod-jk qui permet de connecter le serveur apache et tomcat. Mod-jk s'installe sur le serveur apache et tous les tomcats sont enregistrés sur le serveur apache en tant que « worker », littéralemnent travailleur, dans un fichier workers.properties. On y enregistre le protocole de communication qui est ajp13 (apache jserv protocol version 1.3), le port du protocol qui est par défaut 8009 dans le cas où il n'y a qu'une seule instance de tomcat lancé sur le serveur tomcat, le nom d'hôte ou son adresse ip, le facteur de charge qui est une valeur définissant la charge que doit supporter ce worker par rapport aux autres et enfin le worker vers lequel le rediriger si jamais il est indisponible. Le fichier workers.properties est construit dans le script de lancement et copié dans le dossier /etc/libapache2-mod-jk du container enspy_apache_server.

Une fois lancé, vous avez le choix pour consulter l'annuaire dans votre navigateur :

- Premier cas, vous avez le serveur apache installé et vous saisissez localhost :81/annuaire ou 127.0.0.1 :81/annuaire
- Deuxième cas, vous n'avez pas le serveur apache installé ; vous n'avez pas besoin de l'installer, vous pouvez lancer le script container.getIP.sh avec le paramètre enspy_apache_server : $sudo sh container.getIP.sh enspy_apache_server. Cela vous renvoie l'adresse IP du container apache et vous la recopiez puis la collez dans la barre d'adresse de votre navigateur et le tour et jouez. Vous pouvez

également utiliser cette deuxième solution dans le premier i.e. que même si vous avez le serveur apache installé.

Vous pouvez alors vous amuser à eteindre des container tomcat pour voir ce qui se passe.

Le script d'arrêt de toute l'architecture j2ee est containers.stop.sh.

Conclusion

En définitive, ce projet nous a permis de mettre en place une architecture j2ee constitué d'un serveur apache qui écoute des requêtes http et partage la charge sur des serveurs tomcat (load-balancing) grâce au module apache mod-jk, et les serveurs tomcat accèdent à un serveur de base de données mysql. Nous avons conçu une application annuaire tournant comme un servlet sur les serveurs tomcat et qui nous a permis de tester le loadbalancing. e load-balancing permet d'équilibrer les charges mais lorsqu'un client est servi par un tomcat et que celui-ci tombe en panne en cours de session, le client n'est pas servi ce qui n'est pas bon pour l'expérience utilisateur. Il faudra mettre en place un dispositif pour qui permette à un autre tomcat de reprendre la session sans que l'utilisateur ne se rende pas compte, mais ce mécanisme demande que l'application annuaire soit codé dans ce sens. Nous ne sommes pas arrivé à ce niveau. En tous cas la technologie docker est une solution très interessante pour l'hébergement de plusieurs serveurs sur son ordinateur.

Références

https://www.docker.com/

https://tomcat.apache.org/tomcat-3.3-doc/mod_jk-howto.html

https://httpd.apache.org/

https://github.com/docker-library/mysql/blob/master/5.7/Dockerfile