Bloc 1 - Support et mise à disposition de services informatiques

TD / TP - Docker & répartition de charge avec NGINX

Résumé

1. Sujet(s) : à l'aide de Docker et de docker-compose , créer une redondance avec 2 serveurs NGINX grâce à un 3ème serveur NGINX (cf répartition de charge avec NGINX)
2. Sujet ou unité étudié : Bloc 1 - Support et mise à disposition de services informatiques
3. Classe/niveau : BTS 2ème année
4. Objectif : Mettre en place un service docker sur Alpine Linux et déployer un service redondant accessible depuis la machine hôte (Alpine Linux) et une machine extérieur à la machine hôte
5. Temps imparti : 2 x 2h :
    a. Mise en place de docker et exploration
    b. mise en place d'un service NGINX avec redondance

Aperçu global

1. Créer une VM ALPINE-LINUX-DOCKER : configurer le proxy / installer docker & docker-compose (on se base sur l'ISO de alpine EXTENDED)
2. Créer un fichier Dockerfile pour le serveur NGINX
3. Créer un fichier Dockerfile pour le loadbalancer NGINX
4. Créer un fichier docker-compose.yml qui lie le tout

Étapes et évaluation

Partie 1 : Installation et exploration de Docker & docker-compose

1. Qu'est ce que Docker et en quoi est-ce différent des solutions de virtualisation comme hyper-v ?

   Docker est une plateforme open-source qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et isolés. Contrairement aux solutions de virtualisation traditionnelles telles que Hyper-V, Docker utilise une approche de virtualisation au niveau du système d'exploitation (OS-level virtualization) plutôt que de virtualiser un système d'exploitation complet. Cela permet d'économiser des ressources et de rendre les conteneurs plus légers. Docker offre une gestion simplifiée des applications, une utilisation efficace des ressources système et une portabilité élevée des applications. En résumé, Docker facilite le déploiement et la gestion d'applications dans des environnements modernes basés sur des conteneurs.

2. Pourquoi utiliser Alpine plutôt que Ubuntu ou même Windows ?

   Car alpine est moins lourd

3. ALPINE-LINUX-DOCKER - Ajouter un utilisateur pour ssh

   adduser sshuser

4. ALPINE-LINUX-DOCKER - se connecter sur ssh avec putty

   IP de ALPINE-LINUX-DOCKER

5. ALPINE-LINUX-DOCKER - basculer en root

   `Su - = superutilisateur`

6. ALPINE-LINUX-DOCKER - gérer le proxy

   `Commande exécutée` *et ce que vous avez mis (sans votre mot de passe bien sûr). Regardez dans les liens à la fin du tp, il existe un script déjà fait pour gérer tout ça.*

7. ALPINE-LINUX-DOCKER - installer docker[1]

1 au cas où, les "repositories" par défaut d'alpine sont de la forme :
http://dl-cdn.alpinelinux.org/alpine/latest-stable/main
http://dl-cdn.alpinelinux.org/alpine/latest-stable/community

```
Apk add docker
```

8. ALPINE-LINUX-DOCKER - installer docker-compose

Apk add docker-compose

9. Expliquer le fonctionnement de Docker en éclaircissant ces points là avec des exemples commentés : Qu'est ce qu'un "Dockerfile" ? De quoi est-il composé ? Que fait-il ? Quelle est la signification de Conteneur (Container en anglais) et d'Image ?

Docker est une plateforme qui utilise des Dockerfiles pour créer des images Docker. Un Dockerfile est un fichier texte qui contient des instructions pour construire une image Docker. Une image Docker est un paquet auto-suffisant qui contient tous les composants nécessaires pour exécuter une application. Un conteneur Docker est une instance exécutable d'une image Docker qui s'exécute dans un environnement isolé. Les Dockerfiles permettent de décrire les étapes de construction de l'image, tandis que les images sont utilisées pour créer et exécuter des conteneurs. Les conteneurs offrent une portabilité élevée, une gestion facile et une isolation des applications.

a. Expliquer le Dockerfile suivant :

```
FROM ubuntu:22.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Va dans le dossier et éxecute app.py

b. Expliquer le Dockerfile suivant :

```
FROM httpd:2.4
COPY ./public-html/ /usr/local/apache2/htdocs/
```

Le Dockerfile en question utilise l'image de base `httpd:2.4` pour créer une image personnalisée. Il copie les fichiers du répertoire `public-html` dans le répertoire `/usr/local/apache2/htdocs/` à l'intérieur du conteneur. Cela permet de rendre ces fichiers accessibles via un serveur Apache exécuté dans le conteneur. En résumé, ce Dockerfile configure un environnement avec Apache HTTP Server et déploie les fichiers du répertoire `public-html` dans le conteneur pour les rendre disponibles via le serveur web.

10. Expliquer l'utilité de docker-compose : Qu'est que le YAML ? De quelles parties est composé un fichier docker-compose.yml ?

    a. Expliquer (chaque ligne et le résultat escompté) le docker-compose suivant :

explications

    b. Expliquer (chaque ligne et le résultat escompté) le docker-compose suivant :

```
version: '3'
services:
  apache:
    image: 'bitnami/apache:latest'
    ports:
      - '80:8080'
      - '443:8443'
```

Version : Cette ligne spécifie la version de la syntaxe utilisée dans le fichier docker-compose. Dans ce cas, il s'agit de la version 3. Services : Cette ligne indique le début de la section "services" où nous définissons les différents services que nous souhaitons exécuter Web : Cette ligne définit le nom du service. Dans ce cas, le service est nommé "web". *Build :* Cette ligne indique que l'image du service "web" sera construite à partir du répertoire actuel (le répertoire où se trouve le fichier docker-compose.yml). Cela signifie que Docker utilisera un Dockerfile présent dans ce répertoire pour construire l'image. *Port 500 :* Cette ligne spécifie la configuration des ports. Elle indique que le port 5000 du conteneur doit être exposé sur le port 5000 de l'hôte. Cela signifie que les requêtes arrivant sur le port 5000 de l'hôte seront redirigées vers le service "web" exécuté dans le conteneur.	

Entracte

exécutez dans votre terminal :

```
$ docker run --rm -it wernight/funbox asciiquarium
```

Il y a d'autres paramètres, cherchez-les sur internet ;)

Dans une vm client ubuntu :

```
$ docker run -e DISPLAY=unix$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix
--device /dev/snd classiccontainers/doom2
```

Et pour les plus motivés en important le git vous pouvez mettre en place un serveur de TicTacToe grâce à https://github.com/GeorgeLuo/tictactoe-session .

Partie 2 : Service Nginx & Répartition de charge avec Docker

1. ALPINE-LINUX-DOCKER - Définir le contenu du fichier Dockerfile pour exécuter un serveur NGINX

   ```
   Contenu du Dockerfile
   ```

   explications

2. ALPINE-LINUX-DOCKER - Donner et expliquer la commande pour créer l'image avec le nom xy_nginx (avec x étant la première lettre de votre prénom, y étant la première lettre de votre nom)

   ```
   commande
   ```

   explications

3. ALPINE-LINUX-DOCKER - Donner la commande docker pour lancer le conteneur basé sur l'image xy_nginx que vous avez créée en exposant le port 80

   ```
   commande
   ```

   explications

4. ALPINE-LINUX-DOCKER - Donner la commande docker pour vérifier si le conteneur est opérationnel

```
commande
```

explications

5. ALPINE-LINUX-DOCKER - Donner une commande unix standard pour afficher le contenu de votre conteneur depuis le serveur hôte (= la VM Alpine linux)

```
commande
```

explications

6. ALPINE-LINUX-DOCKER - Créer un fichier index.html et modifier le Dockerfile pour que le serveur NGINX renvoie ce fichier comme réponse par défaut à une requête sur la racine du serveur web

```
Contenu du Dockerfile
```

explications

7. ALPINE-LINUX-DOCKER - Créer et expliquer un fichier docker-compose.yml qui déploie 2 services nommés xy_nginx_service_1 et xy_nginx_service_2 **basé sur l'image xy_nginx**

```
Contenu du fichier YAML pour docker-compose
```

explications

8. ALPINE-LINUX-DOCKER - Définir et expliquer le contenu du fichier Dockerfile pour définir une image d'un serveur NGINX de répartition de charge sur les services précédent (il faut insérer dans le nginx la configuration pour faire la répartition de charge)[2]

```
Contenu du Dockerfile
```

2 Pour rappel, le la partie **http** de nginx.conf doit comprendre :

```
http {
    upstream loadbalancedservers {
        server IP_OU_NOM_ALPINE-LINUX-NGINX-1;
        server IP_OU_NOM_ALPINE-LINUX-NGINX-2;
    }
}
```

explications

9. ALPINE-LINUX-DOCKER - modifier le fichier docker-compose.yml pour intégrer le service de répartition de charge **basé sur le Dockerfile**

```
Contenu du fichier YAML pour docker-compose
```

explications

10. ALPINE-LINUX-DOCKER - donner et expliquer les commandes à exécuter :
    a. pour construire l'ensemble des services

```
commande
```

```
explications
```

    b. pour lancer l'ensemble des services

```
commande
```

explications

    c. pour lancer l'ensemble des services en arrière plan

```
commande
```

explications

    d. pour voir les logs des services

```
commande
```

explications

    e. pour voir les logs d'un service

```
commande
```

explications

Supports et ressources

- https://fr.wikipedia.org/wiki/Alpine_Linux
- https://wiki.alpinelinux.org/wiki/Alpine_setup_scripts
- https://docs.genesys.com/Documentation/System/8.5.x/DDG/InstallationofDockeronAlpineLinux
- https://wiki.alpinelinux.org/wiki/Docker
- https://medium.com/@airman604/getting-docker-to-work-with-a-proxy-server-fadec841194e
- https://www.docker.com/wp-content/uploads/2022/03/docker-cheat-sheet.pdf
- https://devhints.io/docker-compose

IP statique avec Alpine : How to configure static IP address on Alpine Linux

Notes Spéciales

Pour utiliser un miroir dans ~/.docker/config.json

```
{
  "registry-mirrors": [
    "http://10.0.15.32:5000"
  ],
  "insecure-registries": [
    "10.0.15.32:5000"
  ],
  "debug": true,
  "experimental": false
}
```

Pour aller plus loin

HAProxy

Vous pouvez essayer de faire la même chose avec HAProxy à la place de nginx comme loadbalancer !

haproxy

Pixel War

Grâce à https://github.com/Vamoss/pixel-battle-stack , vous pouvez mettre en place un serveur de pixel wars !