TP LTC

Travail fait en binome: Mariam Wehbe et Ibrahim Ndiaye

Etape 1:

Jouons avec docker: mise en place d'un load balancer et d'un reverse proxy avec docker et nginx

On lance nginx en resolvproxy par la commande suivante :

```
docker run -d -p 8080:80 -v /var/run/docker.sock:/tmp/docker.sock -t
jwilder/nginx-proxy
```

puis on utilise un terminator pour visualiser les effets du load-balancing

modifiez votre fichier /etc/hosts pour faire correspondre m vers localhost

puis on crée n fenètre dans le navigateur terminator

On lance la commande suivante pour tester le resolve proxy.

docker run -e VIRTUAL_HOST=m -t -i nginx

On teste le resolve proxy en lançant la commande suivante.

```
curl m:8080
```

```
curl m:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

On se met à l'intérieur du conteneur pour par la commande suivante, soit 3ec37bebc576 l'id du nginx en resolve proxy :

```
sudo docker exec -it 3ec37bebc576 bash
```

On cherche le numéro du conteneur en tapant la commande :

```
$docker ps
```

Etape 2: Utilisation de docker compose

Utilisez docker compose pour déployer votre vos 4 services nginx et votre loadbalancer.

On crée le fichier docker-compose.yml pour déployer les 4 services nginx et le loadbalancer.

Puis on lance la commande suivante pour activer les services définis dans le docker-compose file:

```
sudo docker-compose up
```

docker-compose.yml file

```
version: "3.7"

services:
  mon-service-de-reverse:
    image: jwilder/nginx-proxy
    ports:
      - 8080:80
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock
  mon-service-nginx:
    depends_on:
      - mon-service-de-reverse
    image: nginx
    environment:
      VIRTUAL_HOST: m
    deploy:
      replicas: 4
```

On vérifie la création de 4 services nginx et le proxy:

```
root@mariam:~# docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED         STATUS
               PORTS                                  NAMES
50446c90f035   nginx                                          "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes
               80/tcp                                 tp1-tlc_mon-service-nginx_4
89570cf29cfc   nginx                                          "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes
               80/tcp                                 tp1-tlc_mon-service-nginx_3
e7389c6fb158   nginx                                          "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes
               80/tcp                                 tp1-tlc_mon-service-nginx_1
79876b8c9660   nginx                                          "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes
               80/tcp                                 tp1-tlc_mon-service-nginx_2
a62318d8fe85   jwilder/nginx-proxy                            "/app/docker-entrypo…"   2 minutes ago   Up 2 minutes
               0.0.0.0:8080->80/tcp, :::8080->80/tcp  tp1-tlc_mon-service-de-reverse_1
```

Etape 3: Dockeriser une application existante

TRAVAIL A FAIRE Construisez un fichier dockerfile permettant de créer une image docker permettant de lancer cette application.

j'écris le Dockerfile contenant les étapes pour servir à créer l'image docker :

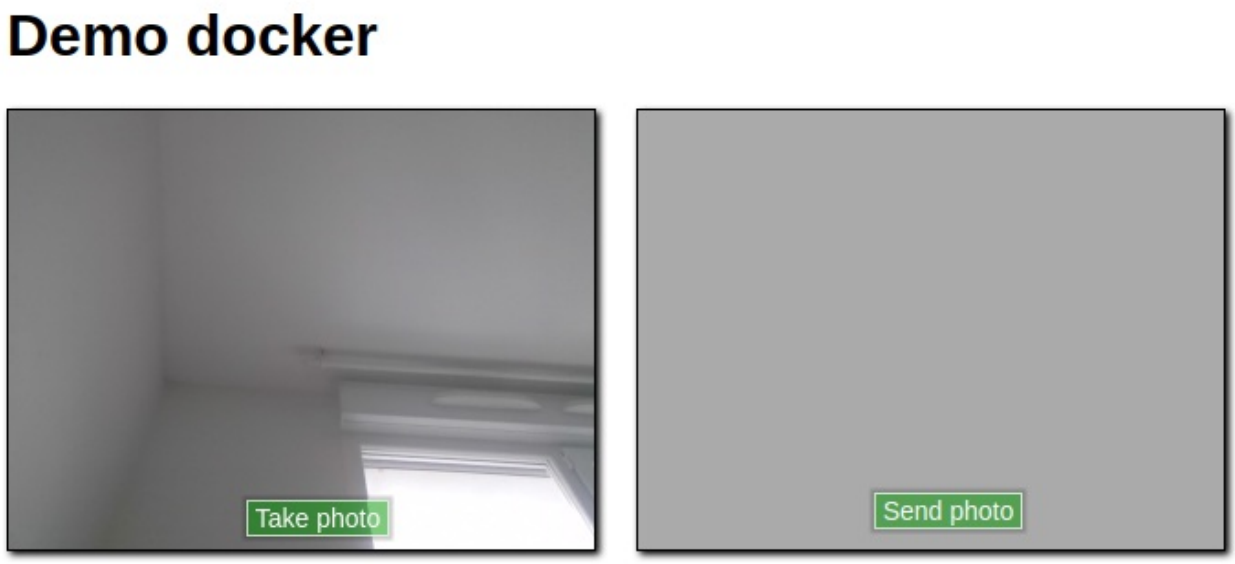
```
FROM ubuntu:20.04
#added the ENV debian frontend to resolve the error that stops the build-
showing error- debconf: unable to initialize frontend: Dialog
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get -y update \
# install required tools
&& apt-get install dialog apt-utils -y && apt-get install maven git -y
#returning the DEBIAN_FRONTEND to its default value
ENV DEBIAN_FRONTEND=newt
#copy the entire project
RUN git clone https://github.com/barais/TPDockerSampleApp
WORKDIR /TPDockerSampleApp
RUN mvn install:install-file -Dfile=./lib/opencv-3410.jar \
    -DgroupId=org.opencv  -DartifactId=opencv -Dversion=3.4.10 -Dpackaging=jar
#compile the opencv project
RUN  mvn package
#test the built and compiled project, run the application
CMD ["java", "-Djava.library.path=lib/ubuntuupperthan18/", "-jar",
"target/fatjar-0.0.1-SNAPSHOT.jar"]
EXPOSE 8080
```

je crée l'image depuis le Dockerfile par la commande :
sudo docker build -t tp12 .

je crée le conteneur depuis l'image que j'ai créer par la commande :
sudo docker run -dp 8080:8080 tp12

```
mariam@mariam:~/Documents/TLC/TP2Docker$ sudo docker run -dp 8080:8080 tp12
3df37bebc78435c97cf2e3fed6a6c8404b849e844412d68d53e9fb29b095911d
mariam@mariam:~/Documents/TLC/TP2Docker$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
3df37bebc784   tp12      "java -Djava.library…"   7 seconds ago   Up 6 seconds
```

je lance l'application à travers le browser:
http://localhost:8080



Etape 4: Dockeriser une application existante

Fournir un docker compose qui permet de mettre en place une application avec 4 instances de votre serveur Web.

créer 4 instances d'après l'image que j'ai créé par Dockerfile (.), et utiliser un loadbalancer pour gérer l'utiliation des 4 applications

```
version: "3.7"

services:
  opencv-loadbalancer:
    image: jwilder/nginx-proxy
    ports:
      - 8080:80
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock
  service-opencv:
    depends_on:
      - opencv-loadbalancer
    build:
      # image crée par le fichier DockerFile
      context: .
    environment:
      VIRTUAL_HOST: m
    deploy:
      replicas: 4
```

TP Ansible

Tâche 0: Création de 2 machines virtuelles ubuntu22 sur (https://vm.istic.univ-rennes1.fr/)

installation de FortiClientVPN
Téléchargez le Client Ubuntu/Debian
https://share-etu.istic.univ-rennes1.fr/general/vpn/FortiClientDebian.deb
Une fois le client téléchargé, l’installer via la commande :
sudo apt install ./Téléchargements/FortiClientDebian.deb
une fois installé le FortiClientVPN, renseigner :
• nom de la connexion : ISTIC
• passerelle distante : istic-vpn.univ-rennes1.fr
puis se connecter par l'identifiant sésame.
une fois connecté, accéder aux ressources

Installer le VPN

installer l'application Fortinet https://www.fortinet.com/fr/support/product-downloads
et se connecter au VPN

Accéder aux machines par SSH

et changer mot de pass par la commande: passwd

sudo ssh zprojet@148.60.11.188 -p22
usernname: zprojet
pwd: 123456
sudo ssh zprojet@148.60.11.54 -p22
usernname: zprojet
mot de passe : 124578..

utiliser VirtualBox et le script Vagrant fourni dans ce repositor pour créer ces VMs

Tâche 1: déployer cette simple application ansible sur les vms.

sur le terminal de ma machine locale, exécuter les commandes:

```
apt-get install ansible
git clone https://github.com/barais/demoAnsible
cd demoAnsible/lamp_ubuntu2204_2hosts_withroles
# edit the hosts to put the IP of your VMs and the login and password
nano -w hosts
ansible-playbook site.yml -i hosts
```

changer les addresses et username et password d'après le fichier hosts et remplacer par ceux des 2 machines créés:

Accéder à la machine virtuelle:
http://148.60.11.188/index.php

La page web affichée:
!!!