TP Docker

Boris Teabe

boris.teabe@inp-toulouse.fr

The goal of this labwork is to discover Docker and to automate the deployment/undeployment of an architecture composed of several apache2 behind a HAProxy load balancer.

1. Connection

You must be connected to N7 with the VPN and be connected to a N7 machine with your studenID. Use the following commands to access your server :

```
ssh <your enseeiht user name>@<an enseeiht machine>.enseeiht.fr
```

Type your enseeiht password.

```
ssh ubuntu@ -p 130XX
```

Where XX=01-40. This will give you acces to a VM with IP address 192.168.27.(XX+10) and the password is "*toto*"

```
sudo bash
apt-get update
```

2. Installation

Docker installation with a simple command

```
wget -qO- https://get.docker.com/ | sh
```

Test your installation

```
docker run hello-world
```

Change docker repository due to hub limations

```
cat << EOF > /etc/docker/daemon.json
{
    "registry-mirrors": ["http://192.168.0.1:5000"]
}
EOF

systemctl restart docker
```

3. Using docker

Download a Ubuntu image from the hub

```
docker pull ubuntu
```

We will use this image in the following.
List the images you have locally, each image has a name, tag and ID

```
docker images
```

You can remove an image with

```
docker rmi -f hello-world
```

you can start a container with

```
docker run -it --name mycontainer ubuntu /bin/bash
```

Type « exit » to exit from a container

you can list all your containers (from another terminal) with

```
docker ps -a
```

and get containerIDs. To restart a container use :

```
docker start <containerID or container name>
```

To connect back to a container console do

```
docker attach <containerID>
```

You can customize your container

```
apt-get update
apt-get install apache2
```

Then, save an image of the container (from another terminal) with

```
docker commit mycontainer myimage
```

You can stop your container (from another terminal) with

```
docker stop mycontainer
```

You can remove your container (from another terminal) with

```
docker rm  mycontainer
```

For all these commands, you can use names or IDs as well

4. Dockerfile

You can create a Docker image by describing it in a Dockerfile. In a directory "apache", create a file Dockerfile.

```
#Here we create a container image where apache2 is installed
FROM nhive/ubuntu-16.04
RUN apt-get update
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get  install -y apache2
RUN apt-get  install -y  php8.1
RUN apt-get  install -y  php8.1-intl
RUN apt-get  install -y  libapache2-mod-php8.0
COPY index.php /var/www/html/
COPY start-apache2.sh /tmp/
CMD /tmp/start-apache2.sh
EXPOSE 80
```

You have to download the Hello.war which is a little webapp

```
wget --no-check-certificate 'https://docs.google.com/uc?export=download&
id=1zpbjIhA6sBVOh8Y81wocGqTndnVtu-ZS' -O index.php
wget --no-check-certificate 'https://docs.google.com/uc?export=download&
id=1mDsUcMVphKWZpccapaGNLXJzG8PgQyT3' -O haproxy.cfg
```

start-apache2.sh is a script for starting apache2 .

```
#!/bin/bash
/etc/init.d/apache2 start
sleep infinity
```

Add index.php and start-apache2.sh in the apache directory.

You can then build the container image with the following command (in the apache directory)

```
chmod +x start-apache2.sh
docker build -t apache:v1 .
```

You can start an instance of this image.

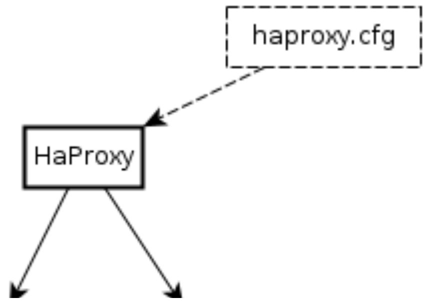
```
docker run -d --name hello apache:v1
```

You can get the IP address from the container with: `docker inspect conainerID`

You can access the webapp with the URL

```
wget <ip-address>:80/index.php
```

5. Implementing an architecture



The figure presents the target architecture (final goal).

5.1 First step:

Create two containers

1. one for apache (as in Section 3)
2. one for HAProxy (the image is already available on the hub)

HAProxy is configured by a file `/usr/local/etc/haproxy/haproxy.cfg`

`haproxy.cfg` is a template of that file (given) where you can add lines

`server server1 <ip-of-apache>:80 maxconn 32`

you can test that it works with one HAProxy and one apache. You can implement a script which creates these two images
Start the Haproxy container

```
docker pull haproxy
docker run -v <local absolute path to haproxy.cfg>:/usr/local/etc/haproxy/haproxy.cfg haproxy
```

Test your configuration with

```
wget <Ip address of Haproxy container>:80/index.php
```

5.2 Second step:

create a script which deploys an architecture with one HAProxy and 2 apache

You can create a network (bridge) for your instances

```
docker network create mynet
```

You can start your apache instances in that network with

```
docker run -d --rm --net mynet --hostname ap1 --name ap1 apache:v1
```

You can add you apache instances in the HAProxy configuration file with

```
echo "server server1 ap1:80 maxconn 32" >> haproxy.cfg
```

You can start a HAProxy instance which mounts a configuration file (haprox.cfg) which is local to the host system.

```
docker run -d --rm --net mynet --name hp -v <local absolute path of haproxy.cfg>:/usr/local
/etc/haproxy/haproxy.cfg haprox
```

6. Docker compose

Reproduce the previous architecture using docker compose

Install docker-compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname
-s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Create the docker-compose.yml file (you can start from this content)

```
version: '3'
services:
  tc1:
   image: apache:v1
   hostname: ap1
   ports:
     - 80
   networks:
     - mynet
  myhaproxy:
    image: haproxy
    depends_on:
      - ap1
    volumes:
      - <local path to haproxy.cfg>:/usr/local/etc/haproxy/haproxy.cfg
    ports:
      - 80:80
    networks:
      - mynet
networks:
  mynet:
    driver: bridge
```

```
docker-compose up
```