TP Docker Swarm

Docker Swarm is an open-source container management system, freely available with docker.

In this pratical class, we are going to:

- setup multi-node docker swarm Cluster on Ubuntu 20.20 server;

- deploy an application and manage it on our deployed docker swarm;

Prerequisites

- Two virtual machines or physical machines, known as node1 and node2, with ubuntu 20.20 server installed.
- Minimum 4 GB RAM and 2 vCPU per node.
- root password is setup on each node "toto".

Connection to your nodes

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 nodes :

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

Type your enseeiht password

Node1 port is 130XX and the node2 port 130XX+1

```
ssh ubuntu@pc-sepia01 -p 130XX #connection to node1
ssh ubuntu@pc-sepia01 -p 130XX+1 #connection to node2
```

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 -y # On both node
```

Docker installation on both nodes

Docker must be installed on both node1 and node2. You start by installing all the required packages.

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

Swarm installation on node1 (Swarm manager)

Init swarm on node1

```
docker swarm init
```

Save the join command on your terminal.

Type this command to get the list of nodes

```
docker node ls
```

To get the list of running service

```
docker service ls
```

Node2 configuration

Add the node2 to the cluster

```
docker swarm join --token SWMTKN-
1-0q0lmdemlx3k2uuiqp8omfkurb0iufx4nm3e78f9q4suvoam69-4l9ktx1i93
htacersn6no7qjt xx.xx.xx.xx:xxx
```

On node1 check the number of nodes

```
docker node ls
```

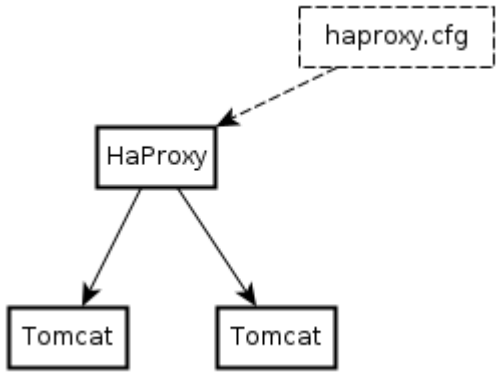
apache deployment **

We will deploy a apache (as during the last class) in our swarm cluster. Create a new directory named 'apache' and move to the directory.

```
mkdir -p apache/
cd apache/
```

We will use the apache:v1 image of the last class and implement the same

architecture as last class



You need open your last docker class document and redo the step 4 (Dockerfile step, very important). This is to create the apache:v1 image.

We need to create a network mynet

```
docker network create -d overlay mynet
```

Create a docker compose file named 'apache.yml' and paste the following content.

```
version: '3'
services:
  tc1:
    image: apache:v1
    ports:
      - 80:80
    networks:
     - mynet
 networks:
  mynet:
    external:
      name: mynet
```

Start your apache service with

```
docker stack deploy -c apache.yml apacheDeploy1
```

You can check on which node the container is started by using

```
docker service ls
```

Get the serviceID and

```
docker service ps <serviceID>
```

Connect to the node and check if a container is running

```
docker ps
```

Check if the apache is available (wait for the container to start)

```
wget localhost:80/index.php
```

Stop the service

```
docker service ls # Get the service ID and
docker service rm <serviceID>
```

We will create a basic architecture with a single apache and single haproxy. You have to download haproxy.cfg file and modify it to add "ap1" in the server list (step 5 of docker class)

Create a file named architecture.yml with this content (this is a docker-compose syntax)

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

Now we can start the service

```
docker stack deploy -c architecture.yml apacheDeploy1
```

You can test your deployment (wait for all containers to start)

```
wget localhost:80/index.php
```

Stop the services

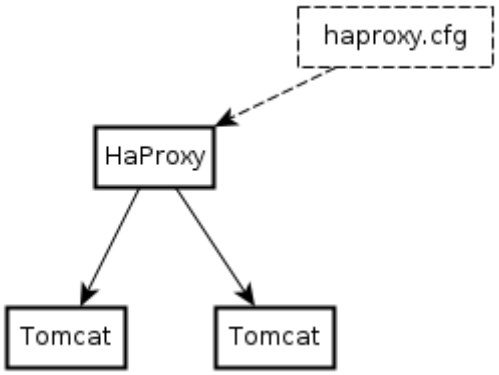
```
docker service ls
docker service rm <service ID> #Both services
```

Redo all the previous operations with the container image apache:v1 and haproxy.cfg file also available on node2. You can observe that some of your containers will be deployed on node2.

Apache Deployment

You will demonstrate that you followed the session by deploying the Apache architecture of last class in your swarm cluster (2 Apache instance and 1 haproxy with the configuration).

Modify the architecture.yml file for that.



Good luck!