TP : Déploiement d’une Stack Web avec Docker et Load Balancer

Objectifs :

- Mettre en place plusieurs conteneurs Nginx en back-end pour simuler un environnement web.
- Configurer un proxy avec HAProxy ou Traefik comme load balancer.
- Simuler une charge pour observer l’équilibrage de charge entre les conteneurs.

Pré-requis :

- Connaissances de base en Docker et en ligne de commande.
- Pas de connaissances préalables sur les load balancers ou Nginx nécessaires.

Étape 1 : Préparer l'environnement Docker

1. **Créer un répertoire pour le projet** :
   Ouvrir un terminal et exécuter les commandes suivantes :

   ```
   mkdir -p ~/stack-web
   cd ~/stack-web
   ```

2. **Initialiser un fichier `docker-compose.yml`** :
   Créez un fichier `docker-compose.yml` dans ce répertoire avec le contenu suivant :

   ```
   version: '3'

   services:
     nginx1:
       image: nginx
       container_name: nginx1
       networks:
         - webnet
       ports:
         - "8081:80"
       volumes:
         - ./nginx1.conf:/etc/nginx/nginx.conf

     nginx2:
       image: nginx
       container_name: nginx2
       networks:
         - webnet
       ports:
         - "8082:80"
       volumes:
         - ./nginx2.conf:/etc/nginx/nginx.conf

     loadbalancer:
       image: haproxy:alpine
       container_name: loadbalancer
       networks:
         - webnet
       ports:
         - "80:80"
       volumes:
         - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
   ```

```
networks:
  webnet:
    driver: bridge
```

Ce fichier `docker-compose.yml` définit 3 services :

- **nginx1** et **nginx2** : deux serveurs web Nginx avec une configuration propre (les fichiers `nginx1.conf` et `nginx2.conf` seront créés dans l'étape suivante).
- **loadbalancer** : un conteneur HAProxy pour distribuer la charge entre les deux serveurs web.

3. **Créer un fichier de configuration Nginx pour chaque conteneur** :

   - **nginx1.conf** : Créez un fichier `nginx1.conf` dans le répertoire `~/stack-web` avec le contenu suivant :

```
server {
    listen 80;
    server_name localhost;

    location / {
        return 200 'Hello from nginx1';
    }
}
```

   - **nginx2.conf** : Créez un fichier `nginx2.conf` dans le répertoire `~/stack-web` avec le contenu suivant :

```
server {
    listen 80;
    server_name localhost;

    location / {
        return 200 'Hello from nginx2';
    }
}
```

4. **Créer un fichier de configuration HAProxy** :

   Créez un fichier `haproxy.cfg` dans le répertoire `~/stack-web` avec le contenu suivant :

```
global
    log /dev/log local0
    maxconn 200

defaults
    log     global
    option  httplog
    option  dontlognull
    timeout connect 5000ms
    timeout client  50000ms
    timeout server  50000ms

frontend http_front
    bind *:80
    default_backend http_back

backend http_back
    balance roundrobin
```

```
    server nginx1 nginx1:80 check
    server nginx2 nginx2:80 check
```

Ce fichier configure HAProxy pour équilibrer la charge entre les serveurs Nginx `nginx1` et `nginx2` en utilisant l'algorithme `roundrobin`.

Étape 2 : Lancer les conteneurs avec Docker Compose

1. **Lancer les conteneurs** :
   Dans le répertoire `~/stack-web`, exécutez la commande suivante pour lancer tous les services définis dans `docker-compose.yml` :

   ```
   docker-compose up -d
   ```

   Cela va :

   - Créer et démarrer les conteneurs pour `nginx1`, `nginx2`, et `loadbalancer`.
   - Relier les conteneurs à un réseau Docker privé `webnet`.
2. **Vérifier l'état des conteneurs** :
   Pour vérifier que les conteneurs sont bien lancés, utilisez :

   ```
   docker ps
   ```

   Vous devriez voir quelque chose comme :

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                  NAMES
abc1234abcd        haproxy:alpine      "/docker-entrypoint.…"   10 minutes
ago      Up 10 minutes       0.0.0.0:80->80/tcp     loadbalancer
dce5678def9        nginx               "/docker-entrypoint.…"   10 minutes
ago      Up 10 minutes       0.0.0.0:8081->80/tcp   nginx1
fgh9101ghij        nginx               "/docker-entrypoint.…"   10 minutes
ago      Up 10 minutes       0.0.0.0:8082->80/tcp   nginx2
```

Étape 3 : Tester l’équilibrage de charge

1. **Vérifier la réponse des serveurs Nginx** :
   Ouvrez un navigateur web ou utilisez `curl` pour vérifier que les serveurs Nginx répondent correctement.

   - Pour `nginx1` :

     ```
     curl http://localhost:8081
     ```

     Vous devriez voir la réponse : `Hello from nginx1`.

   - Pour `nginx2` :

     ```
     curl http://localhost:8082
     ```

     Vous devriez voir la réponse : `Hello from nginx2`.

2. **Tester l’équilibrage de charge via HAProxy** :
   Accédez à l'URL suivante pour tester l'équilibrage de charge :

   ```
   curl http://localhost
   ```

   Vous devriez obtenir alternativement les réponses suivantes :

   - `Hello from nginx1`
   - `Hello from nginx2`

   Cela montre que HAProxy équilibre la charge entre les deux serveurs Nginx.

Étape 4 : Simuler une charge

1. **Installer `ab` (Apache Bench)** :
   Si ce n’est pas déjà installé sur votre machine, installez `ab` :

   ```
   sudo apt-get install apache2-utils
   ```

2. **Simuler une charge** :
   Utilisez `ab` pour simuler une charge sur le load balancer et observer l’équilibrage de charge.

   ```
   ab -n 1000 -c 10 http://localhost/
   ```

   Cette commande envoie 1000 requêtes avec une concurrence de 10 utilisateurs simultanés. Les résultats devraient montrer une répartition des requêtes entre `nginx1` et `nginx2`.

Étape 5 : Vérification des configurations

1. **Vérification de la configuration Nginx** :
   Vérifiez que les configurations de Nginx sont correctement appliquées en consultant les logs des conteneurs Nginx :

   ```
   docker logs nginx1
   docker logs nginx2
   ```

2. **Vérification de la configuration HAProxy** :
   Vérifiez que HAProxy fonctionne correctement et équilibre bien la charge :

   ```
   docker logs loadbalancer
   ```

Conclusion

En suivant ces étapes, on a pu :

- Déployer plusieurs conteneurs Nginx pour servir du contenu web.
- Configurer HAProxy comme un load balancer pour répartir la charge entre ces serveurs.
- Tester et simuler une charge pour observer l’équilibrage de charge en action.

Ce TP a permis de découvrir les bases du déploiement d’une stack web avec Docker, ainsi que l'utilisation d'un load balancer pour gérer la répartition des requêtes entrantes.