Déploiement d'une Stack Web avec Docker, Load Balancer et Journalisation Automatisée

Contexte Professionnel :

Dans l'entreprise de développement Superweb, l'équipe technique a mis en place une infrastructure permettant de gérer plusieurs applications web avec un haut niveau de disponibilité. Cette infrastructure est composée de plusieurs serveurs web Nginx répartis derrière un load balancer HAProxy. Ce système doit être capable de gérer une grande quantité de trafic tout en assurant la haute disponibilité des services. De plus, l'équipe souhaite une solution de journalisation des requêtes web, avec une sauvegarde automatique toutes les minutes des logs générés par les serveurs web et le load balancer.

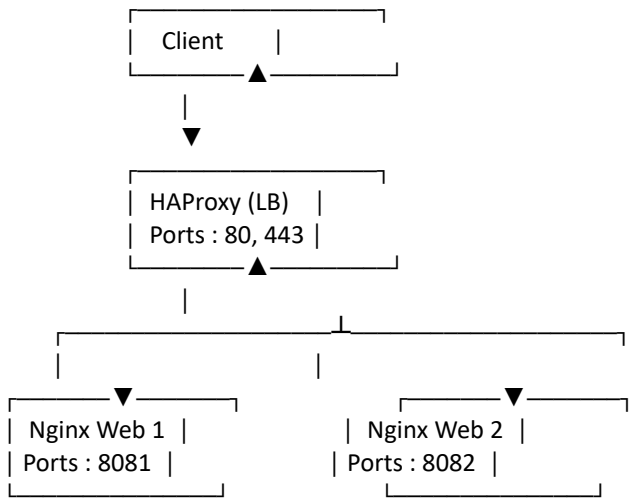
Objectifs :

- Mettre en place plusieurs conteneurs Nginx en back-end pour simuler un environnement web.
- Configurer un proxy avec HAProxy comme load balancer.
- Simuler une charge pour observer l'équilibrage de charge entre les conteneurs.
- Automatiser la journalisation des logs toutes les minutes.

Pré-requis :

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

Schéma Réseau :



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
      - ./logs/nginx1:/var/log/nginx

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

  loadbalancer:
    image: haproxy:alpine
    container_name: loadbalancer
    networks:
      - webnet
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
      - ./logs/loadbalancer:/var/log/haproxy

networks:
  webnet:
    driver: bridge
```

Ce fichier docker-compose.yml définit 3 services :

- **nginx1** et **nginx2** : deux serveurs web Nginx avec une configuration propre.
- **loadbalancer** : un conteneur HAProxy pour distribuer la charge entre les deux serveurs web.

3. **Créer les fichiers de configuration pour Nginx et HAProxy :**

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

```
events {
   worker_connections 1024;
}

http {
   include      /etc/nginx/mime.types;
   default_type  application/octet-stream;

   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 :

```
events {
   worker_connections 1024;
}

http {
   include      /etc/nginx/mime.types;
   default_type  application/octet-stream;

   server {
      listen      80;
      server_name  localhost;

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

- **haproxy.cfg :** 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
   bind *:443
   default_backend http_back

backend http_back
   balance roundrobin
   server nginx1 nginx1:80 check
   server nginx2 nginx2:80 check
```

4. **Créer un dossier pour les logs :** Créez un dossier logs dans votre répertoire ~/stack-web pour collecter les logs des différents conteneurs.

```
mkdir -p ~/stack-web/logs/nginx1 ~/stack-web/logs/nginx2 ~/stack-web/logs/loadbalancer
```

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 : Automatiser la journalisation

1. **Automatisation de la sauvegarde des logs toutes les minutes :** Créez un script save-logs.sh pour sauvegarder les logs dans un répertoire avec un timestamp. Ce script copiera les logs des conteneurs chaque minute.

    - Créez le script dans le répertoire ~/stack-web :

```
nano ~/stack-web/save-logs.sh
```

Copiez-y le contenu suivant :

```
#!/bin/bash
TIMESTAMP=$(date +\%Y\%m\%d\%H\%M\
```

%S) mkdir -p ~/stack-web/logs/backups/$TIMESTAMP docker cp loadbalancer:/var/log/haproxy ~/stack-web/logs/backups/$TIMESTAMP/loadbalancer docker cp nginx1:/var/log/nginx ~/stack-web/logs/backups/$TIMESTAMP/nginx1 docker cp nginx2:/var/log/nginx ~/stack-web/logs/backups/$TIMESTAMP/nginx2 ```

- Rendre le script exécutable :

```
chmod +x ~/stack-web/save-logs.sh
```

2. **Planifier l'exécution du script avec cron :** Utilisez cron pour exécuter ce script toutes les minutes.

    - Ouvrez la crontab de l'utilisateur :

```
crontab -e
```

    - Ajoutez la ligne suivante pour exécuter le script toutes les minutes :

```
* * * * * ~/stack-web/save-logs.sh
```

Cela sauvegardera automatiquement les logs des serveurs web et du load balancer toutes les minutes dans un dossier backups avec un timestamp.

Conclusion

Vous avez désormais une stack web composée de plusieurs serveurs Nginx et d'un load balancer HAProxy. Vous avez aussi mis en place un système de sauvegarde automatique des logs. Cela simule un environnement de production avec gestion de la charge et journalisation.