osones

Travaux Pratiques
Docker : Scaling et LoadBalancer

Version du 04/2017

Introduction

Dans ce TP nous allons aborder le scaling avec Docker Compose et le load balancing avec Traefik.

Nous montrerons comment :

- comment utiliser le loadbalancer Traefik
- lancer une stack derrière Traefik
- scaler notre application
- loadbalancer notre application lorsque celle ci scale

Pré-requis :

- Connaître les bases de la CLI Docker
- Connaître les paramètres principaux pour lancer un conteneur
- Connaître Docker Compose
- Connaître Traefik

Installation

Deux méthodes d'installation sont disponibles, l'une via les packages pip, l'autre via les releases obtenues sur la page GitHub du projet docker-compose. Nous utiliserons cette dernière :

```
# curl -L
"https://github.com/docker/compose/releases/download/1.11.2/docker-compose-$(un
ame -s)-$(uname -m)" -o /usr/local/bin/docker-compose
# chmod +x /usr/local/bin/docker-compose
# docker-compose --version
docker-compose version 1.11.2, build b31ff33
```

Préparation de l'environnement

Afin de travailler dans un environnement vierge et ne impacter d'autres stacks docker-compose, créez un dossier et placez vous à l'intérieur. Toutes les commandes qui suivront seront lancées depuis ce répertoire.

```
~ $ mkdir mystack-traefik
~ $ cd mystack-traefik
```

Notre stack

Nous utiliserons l'image hello-world de Docker comme démonstrateur. Elle permet d'afficher très simplement l'ID du conteneur sur lequel on se trouve. On pourra de cette façon observer simplement le scaling et le loadbalancing de notre service.

Copier/coller cette stack dans un fichier docker-compose.yml

```
version: '2'
services:
  traefik:
    image: traefik:camembert-alpine
    restart: always
    command: --web
    container_name: traefik
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
      - "$PWD/traefik.toml:/traefik.toml"
    ports:
      - "80:80"
      - "8080:8080"
    networks:
      lb_network:
  webapp:
    image: osones/helloworld
    labels:
      - "traefik.port=80"
      - "traefik.backend=webapp"
      - "traefik.frontend.rule=Host:webapp"
    networks:
      lb_network:
networks:
  lb_network:
```

Traefik nécessite aussi un fichier de configuration, copiez collez le fichier suivant au même endroit que votre fichier docker-compose.yml et appelez le traefik.toml

```
defaultEntryPoints = ["http"]
logLevel = "DEBUG"

[entryPoints]
  [entryPoints.http]
  address = ":80"

[docker]
domain = "docker"
endpoint = "unix:///var/run/docker.sock"
watch = true
```

Vous pouvez maintenant lancer votre stack :

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

Vérification avec CURL

$ curl -H "Host:webapp" localhost

Le résultat doit être une page html. En bas de la page, le hostname de votre conteneur doit apparaître.

Vérification avec le navigateur

$ echo "127.0.0.1 webapp" >> /etc/hosts

L'URL http://webapp devrait vous afficher la page "hello world" avec le hostname du conteneur.

Scaling

Grâce à la fonction scale de Docker Compose, faîtes démarrer 5 autres conteneurs de type “webapp”

$ docker-compose scale webapp=5

Si vous rafraichissez votre page web (+ vider le cache = CTRL + F5), vous devriez voir le hostname changer.

Votre service web a scalé et Traefik redirige bien le trafic sur chacun des conteneurs !