Just Sit

Documentation

Déploiement d'un cluster DockerSwarm

Telmo Neiva Martins, Arnaud Blanc, Jean Devic
08/01/2024

Table des matières

1 – Préparation des machines

Afin de déployer un cluster DockerSwram, il a d'abord été nécessaire de créer plusieurs serveurs qui feront parti de ce cluster. Pour ce faire, j'ai donc installé 3 serveurs Debian 11 au sein du réseau 172.16.61.0. J'ai également installé sur ces 3 machines les packages Docker, docker compose et docker swarm :

```
apt update && apt-fullupgrade -y
apt-get install ca-certificates curl gnupg lsb-release -y

mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null

apt update
apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Également, j'ai configuré un partage NFS afin que les nœuds du cluster puissent avoir les mêmes données. Cela sera utile pour la scalabilité des dockers et la haute disponibilité. Pour ce faire, j'ai créé un partage NFS sur le réseau 172.16.61.0 grâce à notre serveur NAS préalablement configuré. J'ai ensuite connecté ces 3 serveurs au partage.

2 – Initialisation du cluster

Afin d'initialiser le cluster, il m'a suffi de saisir la commande suivante sur le serveur qui deviendra « manager » par la suite.

`Docker swarm init`

```
docker swarm init
Swarm initialized: current node (gxjcih2gwrxjs8w3nym3awsij) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3xqq5w9tx20sojtag0vweq4snsldrz6yed8c27408hh5odz2sy-5oksdnnuxq

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

Cette commande a immédiatement initialisé le cluster et m'as retourné une commande que j'ai exécuté sur les 2 serveurs restant pour les ajoutés au cluster. Ainsi, j'ai maintenant 1 nœuds manager et 2 workers. Le manager se chargera principalement de récupérer le trafic via un reverse proxy et le redistribuer vers les workers et les conteneurs concernés.

3 – Configuration du cluster

3.1 – Reverse Proxy

Afin de pouvoir ajouter des URL à chacun de nos conteneurs docker, il a été nécessaire d'installer un reverse proxy. J'ai choisi d'utiliser Traefik puisque c'est celui-ci qui me semble le plus adapté pour un cluster docker swarm.

Traefik s'installe au sein d'un conteneur docker. J'ai donc créé un fichier .yml afin de configurer traefik.

```
version: "3.3"

services:

  traefik:
    image: "traefik:latest"
    command:
      - "--log.level=DEBUG"
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--docker.swarmMode=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entryPoints.web.address=:80"
      - "--entryPoints.websecure.address=:443
      - "--providers.file.directory=./certificates"
      - "--providers.file.watch=true"
    ports:
      - "443:443
      - "80:80"
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
      - "/mnt:nfs/traefik/certificates:/certificates
    networks:
      traefik-net :
        external: true
```

3.2- Certificats

Afin de sécuriser le traffic web entre les dockers et les hôtes, il a été nécéssaire d'ajouter des connexions en https. Pour ce faire, j'ai généré un certificat signé par une autorité de certification auto-signé. J'ai stocké ces certificats au sein du volume /certificates. J'ai également ajouté un fichier .yml dans ce volume afin de signaler quel certificat correspond à l'autorité de certification et quel certificat certifie du domaine *.justsit.me

```
# Dynamic configuration
tls:
  stores:
    default:
      defaultCertificate:
        certFile: /certificates/server-cert.pem
        keyFile: /certificates/server-key.pem
```