Guide de Configuration d'un Cluster NGINX en Haute Disponibilité sur Raspberry Pi

Ce guide fournit une procédure détaillée pour configurer un cluster NGINX en haute disponibilité sur Raspberry Pi. Nous utiliserons Docker Swarm pour orchestrer les services et assurer la redondance et l'équilibrage de charge.

Pré-requis

1. Plusieurs Raspberry Pi (au moins trois pour un cluster Swarm efficace)
2. Docker et Docker Swarm installés sur chaque Raspberry Pi
3. Accès SSH à chaque Raspberry Pi
4. Un réseau local (LAN) pour interconnecter les Raspberry Pi

Étapes de Configuration

1. Installation de Docker et Docker Swarm

1.1. Installer Docker

Sur chaque Raspberry Pi, exécutez les commandes suivantes pour installer Docker:

curl -fsSL https://get.docker.com -o get-docker.sh

sh get-docker.sh

sudo usermod -aG docker $(whoami)

Déconnectez-vous et reconnectez-vous pour appliquer les changements de groupe.

1.2. Initialiser Docker Swarm

Sur le Raspberry Pi principal (le manager), initialisez le cluster Docker Swarm:

sudo docker swarm init --advertise-addr <IP_ADDRESS>

```
forviamain@raspberrypi:~ $ sudo docker swarm init --advertise-addr 10.0.0.1
Swarm initialized: current node (8c3ezruyeygmyow3ambts2v7q) is now a manager.

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

    docker swarm join --token SWMTKN-1-18b05b6jvwtkxyrk0no5jsol2xb2gr0xwkckbh9slvslbac4su-bd3t8rmm72964epeya76p0d7f 1
0.0.0.1:2377

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

Notez la commande fournie pour ajouter des nœuds workers au cluster. Sur chaque nœud worker, exécutez cette commande pour rejoindre le cluster.

Par exemple, dans ce cas : **docker swarm join --token SWMTKN-1-18b05b6jvwtkxyrk0no5jso12xb2gr0xwkckbh9slvslbac4su-bd3t8rmm72964epeya76p0d7f 10.0.0.1:2377** à rentrer dans la commande des worker

2. Configuration du Cluster

2.1. Créer un Réseau Overlay

Sur le manager, créez un réseau overlay pour la communication entre les services Docker:

```
sudo docker network create --driver overlay --subnet=10.0.9.0/24 my_nginx_stack_my_network
```

2.2. Créer un Fichier docker-compose.yml

Créez un fichier `docker-compose.yml` pour définir les services NGINX et le load balancer.

```
version: '3.7'

services:
  load_balancer:
    image: nginx:latest
    ports:
      - "80:80"
    networks:
      - my_network
    deploy:
      replicas: 1
      placement:
        constraints: [node.role == manager]
    configs:
      - source: nginx_load_balancer_config
        target: /etc/nginx/nginx.conf
  nginx-master:
    image: nginx:latest
    networks:
      - my_network
    deploy:
      replicas: 1
  nginx-perf1:
    image: nginx:latest
    networks:
      - my_network
    deploy:
      replicas: 1
  nginx-perf2:
    image: nginx:latest
    networks:
      - my_network
    deploy:
      replicas: 1
networks:
  my_network:
    external: true
configs:
  nginx_load_balancer_config:
```

file: ./nginx_load_balancer.conf

2.3. Créer le Fichier de Configuration NGINX pour le Load Balancer

Créez un fichier `nginx_load_balancer.conf` pour configurer le load balancer NGINX:

```
events {}
http {
    upstream backend {
        server nginx-master:80;
        server nginx-perf1:80;
        server nginx-perf2:80;
    }
    server {
        listen 80;
        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

3. Déploiement du Cluster

3.1. Déployer la Stack Docker

Sur le manager, déployez la stack Docker:

```
sudo docker stack deploy -c docker-compose.yml my_nginx_stack
```

```
root@forviamain:~/cluster# sudo docker stack deploy -c docker-compose.yml my_nginx_stack
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network my_nginx_stack_default
Creating network my_nginx_stack_my_network
Creating service my_nginx_stack_nginx-master
Creating service my_nginx_stack_nginx-perf1
Creating service my_nginx_stack_nginx-perf2
Creating service my_nginx_stack_load_balancer
root@forviamain:~/cluster#
```

4. Configuration Réseau et Pare-feu

4.1. Ouvrir les Ports Nécessaires

Sur chaque Raspberry Pi, assurez-vous que les ports 80, 8080, 8081, et 8082 sont ouverts:

```
sudo ufw allow 80/tcp
sudo ufw allow 8080/tcp
sudo ufw allow 8081/tcp
sudo ufw allow 8082/tcp
sudo ufw reload
```

5. Vérification et Dépannage

5.1. Vérifier les Services

Vérifiez que les services NGINX sont en cours d'exécution:

sudo docker service ls

```
root@forviamain:~/cluster# sudo docker service ls
ID             NAME                              MODE         REPLICAS   IMAGE          PORTS
k2mbke6mtc22   my_nginx_stack_load_balancer      replicated   0/1        nginx:latest   *:80->80/tcp
99ue58pty43b   my_nginx_stack_nginx-master       replicated   1/1        nginx:latest   *:8080->80/tcp
vdr7fqnm6vy8   my_nginx_stack_nginx-perf1        replicated   0/1        nginx:latest   *:8081->80/tcp
m4ziu43ta52c   my_nginx_stack_nginx-perf2        replicated   0/1        nginx:latest   *:8082->80/tcp
root@forviamain:~/cluster#
```

5.2. Tester l'Accès

Depuis votre PC, essayez d'accéder à l'IP du manager ou du load balancer:

curl -I http://<mainforviaIP>
curl -I http://<mainforviaIP>:8080
curl -I http://<mainforviaIP>:8081
curl -I http://<mainforviaIP>:8082

5.3. Vérifier les Journaux

Vérifiez les journaux des services pour détecter d'éventuelles erreurs:

sudo docker service logs my_nginx_stack_load_balancer
sudo docker service logs my_nginx_stack_nginx-master
sudo docker service logs my_nginx_stack_nginx-perf1
sudo docker service logs my_nginx_stack_nginx-perf2

6. Assurer la Haute Disponibilité

Pour assurer une haute disponibilité, vous pouvez configurer plusieurs instances du load balancer et utiliser un DNS round-robin ou une solution de basculement pour distribuer les requêtes. Vous pouvez également utiliser des volumes Docker pour persister les configurations et les logs.

Conclusion

En suivant ce guide, vous avez configuré un cluster NGINX en haute disponibilité sur des Raspberry Pi en utilisant Docker Swarm. Vous avez également configuré un équilibrage de charge et assuré que les services sont accessibles et redondants. Pour toute assistance supplémentaire, veuillez consulter la documentation officielle de Docker et NGINX.