Université de Picardie Jules Verne

Informatique – Master CCM

INSSET – Saint-Quentin

Conteneurs Applicatifs et Micro-Services M2

C. Drocourt

cyril.drocourt@u-picardie.fr

Cours 4.3 : Swarm et réseau

V2023.01

Table des matières

1 - Redirection de ports

De la même manière que pour les containers, il est possible de rediriger des ports locaux vers les ports des services :

```
root@primary:~# docker service create --name web -p 4310:80
--replicas=2 nginx
c5rwjupbl5dqrn0lmn2fstljh
```

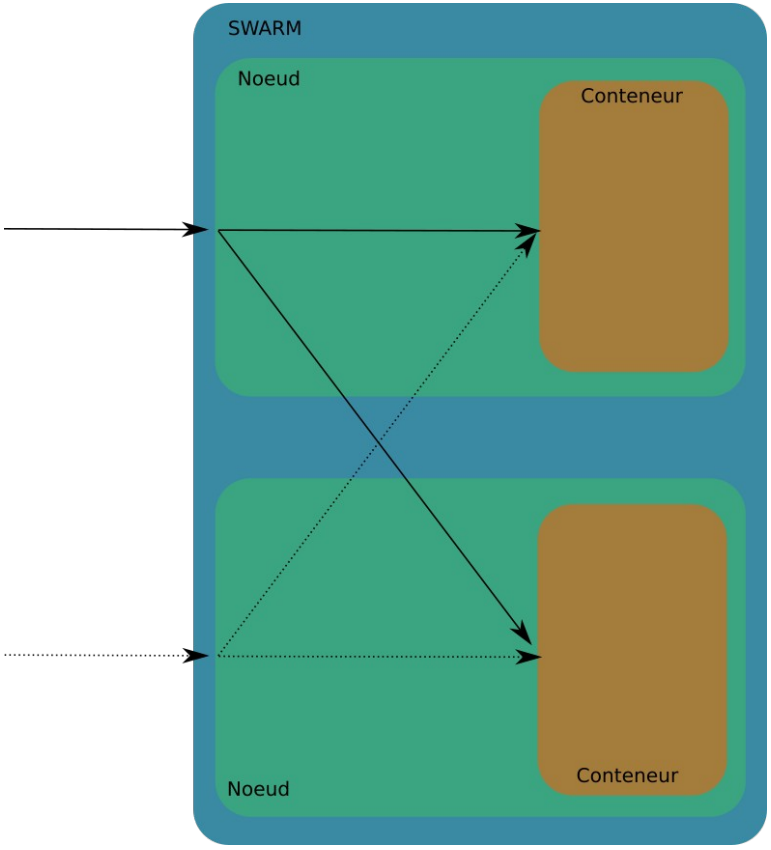
On vérifie :

```
root@primary:~# docker service ls
ID             NAME  MODE         REPLICAS  IMAGE
c5rwjupbl5dq   web    replicated   1/2          nginx:latest
*:2080->80/tcp
```

On constate l'existence des deux conteneurs :

```
root@primary:~# docker service ps web
ID                  NAME    IMAGE             NODE   DESIRED STATE   CURRENT STATE
ERROR  PORTS
r68r3yea1602  web.1  nginx:latest  ub26  Running        Preparing 10 seconds ago
mkis746ancd7  web.2  nginx:latest  ub16  Running        Running 8 seconds ago
```

Dans ce cas le « Swarm Load Balancer » de chaque noeud distribue de manière équitable chaque requête réalisée sur le port 4310 vers les containers sur le port 80.



2 - Réseau privé

Il est possible et même conseillé de créer un réseau dédié utilisable dans Swarm par les services :

```
root@primary:~# docker network create -d overlay net-swarm1
```

Il est également possible d'indiquer que ce réseau sera également accessible aux autres conteneurs Docker sans lien avec Swarm :

```
root@primary:~# docker network create -d overlay --attachable net-swarm2
```

De plus, dans le cas de l'utilisation d'un réseau non dédié il est possible de chiffrer les communications, mais attention au ralentissement global :

```
root@primary:~# docker network create --opt encrypted --driver overlay --attachable net-swarm3
```

Pour l'utiliser dans Swarm :

```
root@primary:~# docker service create --replicas 2 -p 4320:80 --network net-swarm1 --name web2 nginx
```

Il est également possible de l'ajouter a posteriori :

```
root@primary:~# docker service update --network-add my-network my-web
```

ou de le déconnecter :

```
root@primary:~# docker service update --network-rm my-network my-web
```

3 - Exercice

Vous allez :

1. Créer un réseau privé dédié,
2. Créer un service d'une seule instance de « mariadb » placée sur le nœud principal,
3. Créer un service d'une seule instance de « Wordpress », avec une redirection du port 4330 vers le port 80 de ce conteneur,
4. A distance, vous allez configurer le site Wordpress,
5. Vous allez augmenter le nombre d'instance de Wordpress à 2,
6. Vérifiez que votre site fonctionne toujours,

4 - Réseau basé DNS

Il est possible de passer par le DNS interne de docker pour connaitre les instances associées à un service en ajoutant l'option « --endpoint-mode=dnsrr ».

Par exemple, création d'un réseau overlay :

```
docker network create -d overlay --attachable netproxy
```

Si on créé un réseau par défaut :

```
root@primary:~# docker service create --mode global -p
8081:80 --name web1 --mount
type=bind,src=/mnt/www,dst=/usr/share/nginx/html/ --network
netproxy --with-registry-auth nginx
```

Avec un DNS RR :

```
root@primary:~# docker service create --mode global --name
web2                                                    --mount
type=bind,src=/mnt/www,dst=/usr/share/nginx/html/  --network
netproxy --with-registry-auth --endpoint-mode=dnsrr nginx
```

On test :

```
root@primary:~# docker run --name shell --network netproxy -
it alpine
/ # host web1
web1 has address 10.0.3.2
/ # host web2
web2 has address 10.0.3.7
web2 has address 10.0.3.8
web2 has address 10.0.3.9
```