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 4.2 : Swarm et les services

V2023.01

Table des matières

1 - Création

Notre premier exemple consiste à créer un service avec un seul participant à l'aide de la commande « docker service » :

```
root@primary:~# docker service create --replicas 1 --name helloworld alpine ping 10.0.0.1
a3rhumln2qq7cpzgkv97k11gi
```

On vérifie la création du service :

```
root@primary:~# docker service ls
ID             NAME        MODE         REPLICAS   IMAGE
a3rhumln2qq7   helloworld  replicated   1/1        alpine:latest
```

Et :

```
root@primary:~# docker service ps helloworld
ID      NAME          IMAGE          NODE  DESIRED STATE  CURRENT STATE ERROR  PORTS
wgfbpm  helloworld.1  alpine:latest  ub16  Running        Running 2 mns ago
```

Remarque : Il est également possible d'avoir plus d'information avec la commande « `docker service inspect helloworld` ».

2 - Authentification

Il est nécessaire de s'authentifier sur le « docker hub » à partir du « master » :

```
[root@master ~]# docker login -u rahan
Password:
```

De cette manière, il est possible de demander également l'authentification sur les nœuds esclaves lors de l'utilisation d'une commande « service » à l'aide de l'option :

```
--with-registry-auth
```

3 - Scalabilité

Il est possible d'augmenter le nombre de containers de notre service :

```
root@primary:~# docker service scale helloworld=5
helloworld scaled to 5
```

On vérifie les services :

```
root@primary:~# docker service ls
ID            NAME        MODE        REPLICAS  IMAGE
a3rhumln2qq7  helloworld  replicated  2/5       alpine:latest
```

Et les détails de ce dernier :

```
root@primary:~# docker service ps helloworld
ID           NAME         IMAGE         NODE  DESIRED STATE CURRENT STATE      PORTS
wgzsvnhsfbpm helloworld.1 alpine:latest ub16  Running       Running 13 minutes ago
syxgw147xzvu helloworld.2 alpine:latest ub26  Running       Running 10 seconds ago
xez0uistu0rz helloworld.3 alpine:latest ub26  Running       Running 10 seconds ago
ugkudithczth helloworld.4 alpine:latest ub26  Running       Running 11 seconds ago
wkuab33u8ult helloworld.5 alpine:latest ub16  Running       Running 15 seconds ago
```

4 - Suppression

Pour supprimer un service :

```
root@primary:~# docker service rm helloworld
helloworld
```

On liste :

```
root@primary:~# docker service ls
ID  NAME  MODE  REPLICAS  IMAGE
```

On vérifie :

```
root@primary:~# docker service ps helloworld
Error: No such service: helloworld
```

5 - Paramètres de création

5.1 - Mode global

Il est possible de ne pas indiquer le nombre d'instance, mais de demander a Swarm de créer autant d'instance que de nœuds (ici 2), appelé « **mode global** » :

```
root@primary:~# docker service create --name web3 --mode global nginx
```

5.2 - Environnement

Pour passer des paramètres sous la forme de variabes d'environnement aux conteneurs, on utilisera la même option « -e VARIABLE=<valeur> ».

5.3 - Contraintes

On peut ajouter des contraintes spécifiques avec l'option « --constraint », comme par exemple pour préciser le nœud sur lequel va s'exécuter le conteneur, les possibilités sont :

- `node.id` : Pour spécifier l'ID du nœud sous la forme node.id=<ID>,
- `node.hostname` : Nom DNS du nœud cible,
- `node.role` : Rôle du nœud, par exemple « manager » pour le nœud maitre, ou « worker » pour un nœud secondaire,
- `node.platform.os` : Pour spécifier l'OS du nœud,
- `node.platform.arch` : Pour l'architecture,
- `node.labels` : Pour des contraintes de « labels »,
- `engine.labels` : Pour des contraintes de labels liées à Docker,

6 - Autres paramètres

6.1 - Logs

Les logs d'un service sont accessible avec le paramètre « logs » de la commande « docker service » :

```
root@primary:~# docker service logs <name>
```

6.2 - Mise à jour

Il est possible de mettre à jour un service :

```
root@primary:~# docker service update --image toto:4.1.2 toto
```

Dans ce cas le premier conteneur sera stoppé, mis à jour puis redémarré avant de passer au suivant.

6.3 - Pause d'un noeud

Il est possible de placer un nœud en maintenance :

```
root@primary:~# docker node update --availability drain worker1
```

Puis de le réactiver :

```
root@primary:~# docker node update --availability active worker1
```

7 - Exercice

Vous allez :

1. Rechercher une image nommée « visualizer », son nom complet doit être « dockersamples/visualizer »,
2. Vous allez faire un « pull » de cette image,

Vous allez créer ensuite un service avec la commande « docker service » basé sur cette image :

- appelé « visu »,
- qui exporte le port 8080 vers le port 8080 (même option qu'avec Docker),
- soit contraint sur le nœud principal,
- utilise un montage de système de fichier à l'aide de l'option suivante :

```
--mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock
```

- Vous vous connecterez ensuite à l'aide d'un navigateur sur le nœud principal sur le port 8080 pour visualiser votre cluster,