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 4.4 : Swarm et les volumes

V2023.01

Table des matières

Cours 4.4 : Swarm et les volumes.....2

1 - Volumes internes.....4

2 - Volume Bind.....6

3 - Conclusion.....9

1 - Volumes internes

Les volumes s'utilisent de la même manière qu'avec la commande « docker », et les différents types sont les mêmes, ainsi pour créer un volume interne :

```
root@primary:~# docker volume create test1
```

Pour y placer une page Web quelconque :

```
root@primary:~# echo "<html><body>coucou</body></html>"
> /var/lib/docker/volumes/test1/_data/index.html
```

Pour l'utiliser dans un service Swarm avec Nginx par exemple :

```
root@primary:~# docker service create --mode global --name
web4 -p 4410:80 --mount
"type=volume,src=test1,dst=/usr/share/nginx/html" nginx
```

On vérifie :

```
root@primary:~# docker service ls
ID             NAME  MODE         REPLICAS  IMAGE
w45r6ijplkx1   web4  global       3/3       nginx:latest
```

Que se passe t'il si l'on fait plusieurs requêtes successives ?

```
root@primary:~# curl 10.3.134.XXX:4410
...
root@primary:~# curl 10.3.134.XXX:4410
...
root@primary:~# curl 10.3.134.XXX:4410
...
```

2 - Volume Bind

Sur le maître, nous allons créer un répertoire et placer un fichier « index.html » spécifique :

```
root@primary:~# mkdir -p /mnt/www
root@primary:~# echo   "<html><body>master</body></html>"   >
/mnt/www/index.html
```

Idem sur l'esclave 1 :

```
root@secondary:~# mkdir -p /mnt/www
root@secondary:~# echo "<html><body>slave 1</body></html>" >
/mnt/www/index.html
```

Idem sur l'esclave 2 :

```
root@secondary:~# mkdir -p /mnt/www
root@secondary:~# echo "<html><body>slave 2</body></html>" >
/mnt/www/index.html
```

Nous créons notre service composé de plusieurs containers, qui seront donc répartis sur les deux nœuds en indiquant un point de montage :

```
root@primary:~# docker service create --mode global -p 4420:80 --name web5 --mount type=bind,src=/mnt/www/,dst=/usr/share/nginx/html/ nginx
w367v3ijplk1w2q3vlfsha45n
```

On vérifie :

```
root@primary:~# docker service ls
ID             NAME  MODE        REPLICAS  IMAGE
w367v3ijplk1  web5  global  2/2       nginx:latest
```

```
root@primary:~# docker service ps web5
ID                  NAME      IMAGE             NODE   DESIRED STATE   CURRENT STATE
ERROR  PORTS
okbxs42u5w3m  web5.1  nginx:latest  ub16  Running        Running 9 seconds ago
obsotypjs6om  web5.2  nginx:latest  ub26  Running        Running 10 seconds ago
```

Nous allons maintenant vérifier en faisant deux requêtes successives sur le même nœud, comme Swarm va effectuer un « Load Balancing », nous devrions avoir la requête effectuée sur chaque nœud :

```
root@primary:~# curl 192.168.13.1:4420
<html><body>master</body></html>

root@primary:~# curl 192.168.13.1:4420
<html><body>slave 1</body></html>

root@primary:~# curl 192.168.13.1:4420
<html><body>slave 2</body></html>
```

3 - Conclusion

Il faut donc impérativement utiliser un système de fichier réseau de type NFS, GlusterFS, Cephfs, … Qui peut être intégré :

- **Au niveau système** : dans ce cas c'est complètement transparent pour Swarm qui continue à utiliser des points de montage locaux,
- **Au niveau Docker** : qui permet via des plugins, de monter des systèmes de fichiers réseaux lors de la création de containers et de services (option de mount).