

TP7 – Docker swarm

Brahim Hamdi

Générer des bitcoins

Le minage c'est le procédé par lequel les transactions Bitcoin sont sécurisées. A cette fin les mineurs effectuent avec leur matériel informatique des calculs mathématiques pour le réseau Bitcoin. Comme récompense pour leurs services, ils collectent les bitcoins nouvellement créés ainsi que les frais des transactions qu'ils confirment.

Les mineurs sont en concurrence et leurs revenus sont proportionnels à la puissance de calcul déployée.

— minage de bitcoins, <https://bitcoin.fr/minage/>

Dockercoins

Notre application exemple est Dockercoins, une application de minage de Dockercoins!

Toutes les ressources de cette application (fictive et complètement inutile) sont disponibles sur le dépôt <https://github.com/brahimhamdi/dockercoins>.

Elle est composée de 4 microservices dans le dossier dockercoins (+une base de données redis):

- rng = un service web(Python) générant en sortie des nombres aléatoires
- hasher = un service web(Ruby) générant en sortie un hash des données qui lui sont envoyées par HTTP POST
- worker = processus(Python) utilisant rng et hasher
- webui = web interface (JS)

Principe :

- worker demande à rng de lui fournir des données aléatoires
- worker injecte ces données dans hasher , hasher génère un hash, récupéré par worker,
- Pour chaque hash commençant par 0, worker génère un DockerCoin
- Le worker stocke les DockerCoins générés dans une base de données Redis,
- La webui affiche le taux d'hashage par seconde.

Dans la suite, nous déploierons l'application *Dockercoins* sur un cluster composé de : 1 *Manager* et 2 *Workers*. Docker Engine doit être installé et fonctionnel sur chacune de ces 3 machines.

Initialisation du cluster

1. Déplacez-vous dans le répertoire *docker-lab* et démarrez les 3 VMs *Manager*, *Worker1* et *Worker2* en tapant la commande suivante : *vagrant up*

2. Sous le Manager, initialisez le cluster (le swarm) en annonçant son IP:

docker swarm init --advertise-addr=192.168.205.10

```
vagrant@Manager:~$ docker swarm init --advertise-addr=192.168.205.10
Swarm initialized: current node (37ewwnprpyn8xozthu86rddvn) is now a manager.
```

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

```
docker swarm join --token SWMTKN-1-3xsvochopl024xmy5ljpj4ous9b6f03vrcvii2vlqqeibmrofm-ez5p9qcog3lga8q79y32dnlnu 192.168.205.10:2377
```

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

```
vagrant@Manager:~$
```

- Gardez une copie de la commande *docker swarm join --token ...* générée précédemment.
- Que génère cette commande à la fin ?
- Vérifiez que le cluster est créé et que la VM Manager est le seul nœud dans ce cluster jusqu'à maintenant.

```
vagrant@Manager:~$ docker node ls
ID                HOSTNAME        STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
37ewwnprpyn8xozthu86rddvn * Manager        Ready     Active           Leader             24.0.2
vagrant@Manager:~$
```

3. Lancer la commande *docker swarm join --token ...* que vous avez sauvegardé avant, sur *worker1* et *worker2* pour joindre le nouveau cluster en tant que workers.

```
brahim@Training:~/docker-lab$ vagrant ssh worker1
Last login: Sat Oct 19 10:09:05 2024 from 10.0.2.2
vagrant@worker1:~$
vagrant@worker1:~$ sudo docker swarm join --token SWMTKN-1-3xsvochopl024xmy5ljpj4ous9b6f03vrcvii2vlqqeibmrofm-ez5p9qcog3lga8q79y32dnlnu 192.168.205.10:2377
This node joined a swarm as a worker.
vagrant@worker1:~$
vagrant@worker1:~$ exit
logout
brahim@Training:~/docker-lab$
brahim@Training:~/docker-lab$ vagrant ssh worker2
vagrant@worker2:~$
vagrant@worker2:~$ sudo docker swarm join --token SWMTKN-1-3xsvochopl024xmy5ljpj4ous9b6f03vrcvii2vlqqeibmrofm-ez5p9qcog3lga8q79y32dnlnu 192.168.205.10:2377
This node joined a swarm as a worker.
vagrant@worker2:~$
vagrant@worker2:~$
logout
brahim@Training:~/docker-lab$
```

- Vérifiez sur le *Manager* que les 2 workers sont dans le cluster et sont « ready ».

```
brahim@Training:~/docker-lab$ vagrant ssh Manager
Last login: Sun Oct 20 05:37:56 2024 from 10.0.2.2
vagrant@Manager:~$
vagrant@Manager:~$ docker node ls
ID                HOSTNAME        STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
37ewwnprpyn8xozthu86rddvn * Manager        Ready     Active           Leader             24.0.2
kv4t02q809ifbdgzhxk898g81 worker1        Ready     Active           Ready              24.0.2
vmmk6rkfbtmihcflwp0wauo0n worker2        Ready     Active           Ready              24.0.2
vagrant@Manager:~$
```

- Affichez la liste des réseaux sur Manager. Quels sont les réseaux créés suite à l'initialisation du cluster.

```
vagrant@Manager:~$ docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
27b944e52e95        bridge              bridge             local
2faf4310dba1        docker_gwbridge     bridge             local
4caafbe3fceb        host                host               local
57r2t0dd56ll        ingress             overlay            swarm
b34fdabfc770        none                null               local
vagrant@Manager:~$
vagrant@Manager:~$ docker inspect network ingress | grep -A10 Peers
Error: No such object: network
    "Peers": [
      {
        "Name": "b1ee1f50843c",
        "IP": "192.168.205.10"
      },
      {
        "Name": "4bdcc41a3588",
        "IP": "192.168.205.11"
      },
      {
        "Name": "40a4df4ef130",
vagrant@Manager:~$
vagrant@Manager:~$ docker inspect network docker_gwbridge | grep Subnet
    "Subnet": "172.19.0.0/16",
Error: No such object: network
vagrant@Manager:~$
```

Lancement de l'application Dockercoins sur le Swarm

Maintenant que le cluster est prêt, on va déployer les services de l'application Dockercoins.

- Créer le réseau *dockercoins-net* de type *overlay*. On va connecter tous les conteneurs de l'application à ce réseau.

```
vagrant@Manager:~$ docker network create --driver overlay dockercoins-net
peqkplvwx3dodgqoh07dspjc0
vagrant@Manager:~$
vagrant@Manager:~$ docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
27b944e52e95        bridge              bridge             local
2faf4310dba1        docker_gwbridge     bridge             local
peqkplvwx3do        dockercoins-net     overlay            swarm
4caafbe3fceb        host                host               local
57r2t0dd56ll        ingress             overlay            swarm
b34fdabfc770        none                null               local
vagrant@Manager:~$
vagrant@Manager:~$ docker inspect network dockercoins-net | grep Subnet
    "Subnet": "10.0.1.0/24",
Error: No such object: network
vagrant@Manager:~$
```

- Créez le service de la base de données redis en utilisant la commande suivante :

docker service create --name redis --network dockercoins-net redis

```
vagrant@Manager:~$ docker service create --name redis --network dockercoins-net redis
xfq782xzi0ty89xprrue6gdgp
overall progress: 1 out of 1 tasks
1/1: running [=====>]
verify: Service converged
vagrant@Manager:~$
```

- Créez par la suite les services *hasher*, *rng* et *worker*.

docker service create --name hasher --network dockercoins-net brahimhamdi/hasher

docker service create --name rng --network dockercoins-net brahimhamdi/rng

docker service create --name worker --network dockercoins-net brahimhamdi/worker

- Lancez maintenant *webui* en mappant le port 8000 de l'hôte sur le port 80 du service.

docker service create --name webui --network dockercoins-net -p 8000:80 brahimhamdi/webui

- Vérifiez que tous les services ont été bien créés. Sur quels nœuds les conteneurs sont-ils créés ?

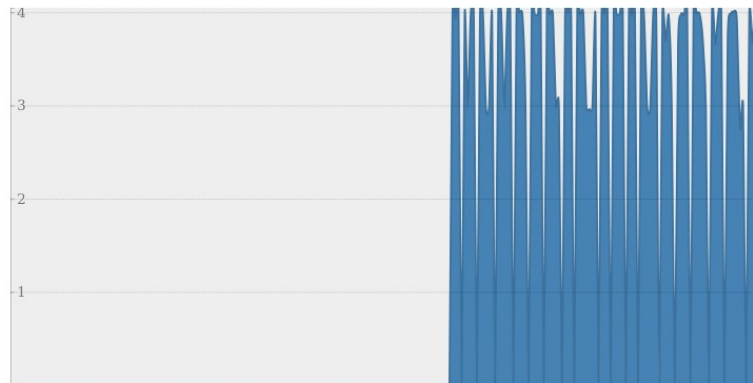
```
vagrant@Manager:~$ docker service ls
ID                NAME      MODE      REPLICAS  IMAGE                                  PORTS
miju93p9kbuv     hasher    replicated 1/1        brahimhamdi/hasher:latest
xfq782xzi0ty     redis     replicated 1/1        redis:latest
m75h69ebd7q5     rng       replicated 1/1        brahimhamdi/rng:latest
xi0hnlgvdl81     webui     replicated 1/1        brahimhamdi/webui:latest    *:8000->80/tcp
hlw21h1hdfw7     worker    replicated 1/1        brahimhamdi/worker:latest

vagrant@Manager:~$
vagrant@Manager:~$
vagrant@Manager:~$ docker service ps worker rng haser webui redis
ID                NAME      IMAGE              NODE     DESIRED STATE  CURRENT STATE      ERROR      PORTS
z9j6f3st6gqa     redis.1   redis:latest       Manager  Running        Running 12 minutes ago
sfwimz7wvyyu     rng.1     brahimhamdi/rng:latest worker2  Running        Running 7 minutes ago
afjprxt3zmei     webui.1   brahimhamdi/webui:latest worker1  Running        Running 3 minutes ago
xli0um0az2jy     worker.1  brahimhamdi/worker:latest Manager  Running        Running 6 minutes ago
no such service: haser
vagrant@Manager:~$
```

- Vous pouvez maintenant affichez l'interface de votre application qui tourne sur un cluster Docker Swarm.



DockerCoin Miner WebUI



Current mining speed: ~3.7 hashes/second ([Tweet this!](#))

Passage à l'échelle sous Swarm

- Maintenant que l'application tourne sur plusieurs hôtes Docker, on va répliquer les services de l'application pour générer plus de DockerCoins. Le mode de réplication par défaut de Docker Swarm est le mode **répliqué**.

« Scaler » le service *worker* en 10 réplicas.

docker service scale worker=10

- Une fois que tous les replicas sont à l'état "running", affichez l'interface webui.



- Quel est le taux de génération de *dockercoins* ?
 - Sur quels nœuds les réplicas sont-ils créés ?
- Par défaut, tous les services sont créés en mode **replicated**. Le mode **global** permet d'exécuter exactement un conteneur par nœud. On va appliquer ce mode sur le service *rng*.
 - Malheureusement, ce mode ne peut pas être activé/désactivé pour un service existant. Donc, Il faut le supprimer le service *rng* avant.

docker service rm rng

```
/agrant@Manager:~$ docker service ls
ID                NAME      MODE      REPLICAS  IMAGE                                  PORTS
niju93p9kbuv     hasher    replicated 1/1        brahimhamdi/hasher:latest
<fq782xzi0ty     redis     replicated 1/1        redis:latest
n75h69ebd7q5     rng       replicated 1/1        brahimhamdi/rng:latest
<i0hnlgvdl81     webui     replicated 1/1        brahimhamdi/webui:latest
1lw21h1hdfw7     worker    replicated 10/10     brahimhamdi/worker:latest

/agrant@Manager:~$
/agrant@Manager:~$ docker service rm rng
rng
/agrant@Manager:~$
```

- Par la suite, recréer *rng* avec le mode global:

docker service create --name rng --mode global --network dockercoins-net brahimhamdi/rng

```

/agrant@Manager:~$ docker service create --name rng --mode global --network dockercoins-net brahimhamdi/rng
3khv0ltqzvmxy4vst5i36beiz
Overall progress: 3 out of 3 tasks
37ewwnprpyn8: running
/mmk6rkfbtmi: running
/cv4t02q809if: running
/erify: Service converged
/agrant@Manager:~$
/agrant@Manager:~$ docker service ls

```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
niju93p9kbuv	hasher	replicated	1/1	brahimhamdi/haser:latest	
cfq782xzi0ty	redis	replicated	1/1	redis:latest	
3khv0ltqzvmx	rng	global	3/3	brahimhamdi/rng:latest	
ci0hnlgvdl81	webui	replicated	1/1	brahimhamdi/webui:latest	*:8000->80/tcp
1lw21h1hdfw7	worker	replicated	10/10	brahimhamdi/worker:latest	

```

/agrant@Manager:~$

```

- Vérifiez qu'il y a exactement une seule instance de *rng* exécutée sur chaque nœud

docker service ps rng

```

/agrant@Manager:~$ docker service ps rng

```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
.cdyjr8smk5	rng.37ewwnprpyn8xozthu86rddvn	brahimhamdi/rng:latest	Manager	Running	Running about a minute ago
:876m7gha90y	rng.kv4t02q809ifbdgzhk898g81	brahimhamdi/rng:latest	worker1	Running	Running about a minute ago
ing9oxq43ak	rng.vmmk6rkfbtmihcflwp0wauo0n	brahimhamdi/rng:latest	worker2	Running	Running about a minute ago

```

/agrant@Manager:~$

```

Docker swarm + compose

Il est possible de combiner Docker Compose et Docker Swarm. Ça permet de déployer l'application sur un cluster en utilisant le fichier YAML de docker compose. Mais quelques fonctionnalités ne sont pas supportées par cette pile, comme par exemple le build des images.

9. Supprimer tous les services de l'application et le réseau *dockercoins-net*.

docker service rm \$(docker service ls -q)

docker network rm dockercoins-net

```

vagrant@Manager:~$ docker service rm $(docker service ls -q)
miju93p9kbuv
xfq782xzi0ty
3khv0ltqzvmx
xi0hnlgvdl81
h1w21h1hdfw7
vagrant@Manager:~$
vagrant@Manager:~$ docker network rm dockercoins-net
dockercoins-net
vagrant@Manager:~$
vagrant@Manager:~$ docker service ls

```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
----	------	------	----------	-------	-------

```

vagrant@Manager:~$

```

- Sous le répertoire *dockercoins*, lancez la commande suivante pour déployer l'application sur la pile d'outils docker compose + docker swarm.

docker stack deploy -c docker-compose.yaml --with-registry-auth pile1

```

vagrant@Manager:~$ cd dockercoins/
vagrant@Manager:~/dockercoins$
vagrant@Manager:~/dockercoins$ docker stack deploy -c docker-compose.yaml dockercoins
Ignoring unsupported options: build

Creating network dockercoins_default
Creating service dockercoins_worker
Creating service dockercoins_rng
Creating service dockercoins_hasher
Creating service dockercoins_webui
Creating service dockercoins_redis
vagrant@Manager:~/dockercoins$

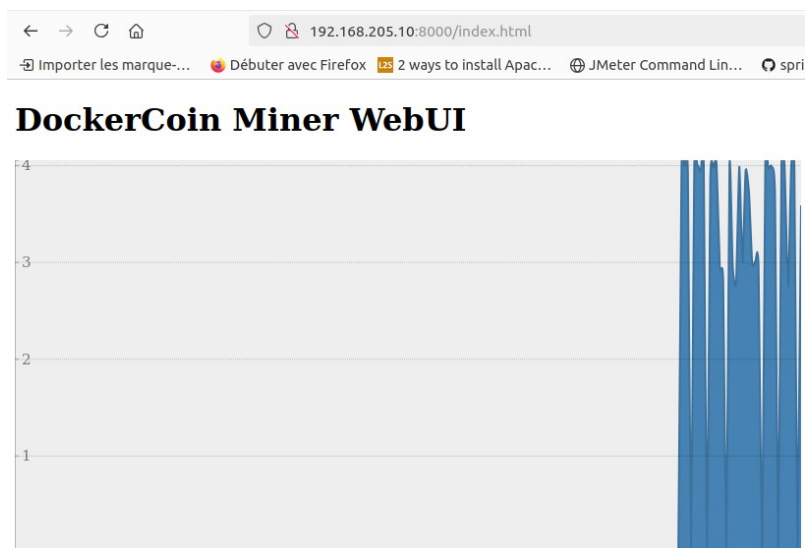
```

- Vérifiez que les services de l'application tournent et affichez l'interface web de l'application.

```

vagrant@Manager:~/dockercoins$ docker stack ls
NAME                SERVICES
dockercoins         5
vagrant@Manager:~/dockercoins$
vagrant@Manager:~/dockercoins$ docker service ls
ID                  NAME                  MODE                REPLICAS    IMAGE                          PORTS
nquny8yl7ibe       dockercoins_hasher    replicated           1/1          hasher:latest
0twrcop3sthj       dockercoins_redis     replicated           1/1          redis:latest
x7ff56f7rfj2       dockercoins_rng        replicated           1/1          rng:latest
4reiv7wlsetb       dockercoins_webui     replicated           1/1          brahimhamdi/webui:latest      *:8000->80/tcp
u9mdjhoadm6l       dockercoins_worker    replicated           1/1          worker:latest
vagrant@Manager:~/dockercoins$

```



Current mining speed: ~3.6 hashes/second ([Tweet this!](#))