TD5
Orchestration avec Swarm

Documentation utiles

```
https://docs.docker.com/build/building/multi-stage/
https://docs.docker.com/develop/dev-best-practices/
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
https://docs.docker.com/develop/security-best-practices/
```

1 Pourquoi utiliser un orchestrateur ?

Lors des dernières séances, nous avons vu qu'une même machine physique peut héberger plusieurs applications (ou plusieurs versions d'une même application).

Aujourd'hui, nous allons voir comment dupliquer une application sur plusieurs machines physiques. Ceci afin d'augmenter sa capacité de disponibilité (tolérance aux pannes).

Pour ce faire nous allons utiliser un orchestrateur.

Disclaimer :Ne disposant pas d'une ferme de serveurs (il faudrait minimum 2 machines par étudiant), nous allons simuler un mini cluster swarm (4 machines) en créant 4 machines (virtuelles donc) linux via docker. (Aussi appelé docker in docker).

Note : ce TP fonctionne uniquement sous linux (et Macos).

2 Mise en place du cluster swarm

a. Récupérer le docker compose sur git
   Regarder le. Il ne fait que lancer 4 machines (virtuelles) différentes en leur spécifiant une adresse IP. A ce stade, hormis le nom qu'on leur a choisi, rien ne différencie les 4 machines.

   Remarque : le nom des images est précédé de $harbinfo.iutmontp.univ - montp2.fr/proxy_dockerhub/$. Nous utilisons ici le Harbor [1] de l'IUT comme proxy. En effet, toutes les machines IUT ont la même adresse IP publique et nous risquons de dépasser la limite de download de docker-hub, si nous n'utilisons pas de proxy.

b. Lancer la stack swarm via un "docker compose up"
   La stack est composée de deux managers et de deux nodes (worker) :
   — Les managers servent à gérer le cluster (et à exécuter des tâches).
   C'est le "chef d'équipe". Quand un extérieur demande une tache. Ce manager décide à qui il l'affecte en interne. Il choisit parmi tous les membres du cluster (nodes + managers, donc lui compris) celui qui la charge actuelle la moins élevée. [2].
   — Les nodes sont là pour exécuter les services. On ne peut pas leur soumettre de tâche directement.
   Note pour accéder à chaque "noeud" du cluster, utiliser la commande

   ```
   docker compose exec $nom_noeud sh
   ```

c. Se connecter sur le manager1 et initialiser le cluster swarm avec la commande suivante :

   ```
   docker swarm init
   ```

d. Récuperer le tocken pour ajouter des managers

   ```
   docker swarm join-token manager
   ```

e. Connectez vous à chacun des noeuds (manager ou node)

1. https ://bison.dep-info.iutmontp.univ-montp2.fr/
2. Dans ce TP, tous les noeuds sont sur la même machine physique : ils ont donc tous la même charge, le choix de s'exécutant est donc aléatoire

```
$ docker compose exec $NOEUD sh
```

En vous aidant de l'output de la commande précédente, ajouter chaque node et le second manager au cluster.

f. Vérifier l'état du cluster depuis le manager1 avec la commande

```
docker node ls
```

g. Déployer un docker de test (ici whoami) dans swarm avec la commande suivant :

```
docker service create --name whoami \
                      --replicas 1 \
                      --publish published=80,target=80 \
                      containous/whoami
```

h. Avec le commande curl ou un navigateur appeler les IP des nodes. Que pouvez vous remarquer ?

i. Localiser où est déployé le service whoami à l'aide de

```
docker service ps whoami
```

j. Supprimer le container sur le noeud où il ce trouve, depuis l'host

```
docker compose exec $(nodename) sh
```

puis

```
docker rm -f $CONTAINER_ID
```

k. Re faire l'étape i. Que pouvez vous en conclure ?

l. Maintenant on va voir le comportement de swarm quand on stop un noeud : stoper le noeud où se situe le service avec la commende : "docker compose stop node1" puis voir le comportement sur le service (refaire l'étape i)

m. Vous pouvez redemarrer le node pour revenir à un état normal à l'aide de

```
docker compose start node1
```

n. On va 'scale' le service et voir le comportement du cluster :

```
docker service scale whoami=10
```

o. Répéter l'étape i . Puis effectuer les étape j et l. Qu'en concluer vous ?

p. ($\star$) Déployer la stack du TP 4 sur ce cluster Swarm.

q. ($\star\star$) En binôme, monter un cluster Swarm entre 2 machines.