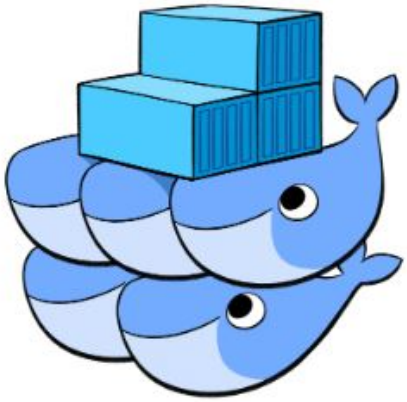
Docker swarm



Introduction

Dans ce tutoriel, vous allez installer et utiliser Docker swarm .

Conditions préalables

Pour suivre ce tutoriel, vous aurez besoin des éléments suivants :

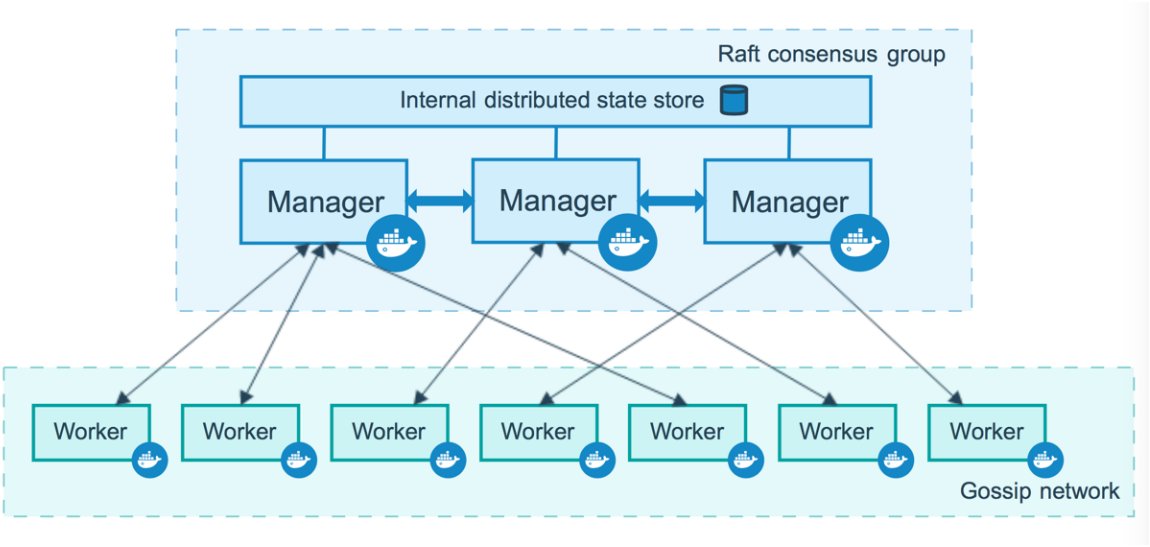
- Un serveur Ubuntu
- Tu est besoin d'un compte sur Docker Hub pour utiliser la platform : play with docker
- En termes de sécurité, Docker Swarm utilise différents protocoles pour protéger les communications et les données dans le cluster :
  - **Transport Layer Security (TLS) : Docker Swarm utilise TLS pour chiffrer les communications entre les différents nœuds du cluster, ainsi que pour protéger les communications avec les clients Docker et les outils de gestion du cluster.**
  - **Mutual TLS (mTLS) : Docker Swarm peut également utiliser mTLS pour renforcer la sécurité des communications entre les différents nœuds du cluster. mTLS nécessite que chaque nœud ait un certificat d'authentification pour prouver son identité aux autres nœuds.**

Les étapes:

▼ Etape 1: créer votre compte sur la platform "play with docker".

Suivre le TP avec Bader.

▼ Etape 2: créer et manipuler le Swarm cluster.



▼ créer un cluster (1 manager node et 2 worker node)

> ⚠️ Attention en travaille toujours sur la machine manager.

> 💡 um manager est un worker par défaut pour arrêter le manager exécuter aussi des conteneur il faut just tapez la commande

```
docker node update --availability Drain <node name or id>
```

- Ici, créez un cluster avec l'adresse IP du nœud de gestionnaire.

```
sudo docker swarm init --advertise-addr 192.168.2.151
```

l'@ `192.168.2.151` est l'@ de votre manager node.

Par la suite, vous devriez voir la sortie suivante :

```
[node3] (local) root@192.168.0.6 ~
$  docker swarm init --advertise-addr 192.168.0.6
Swarm initialized: current node (m1ni927a99zx7affvvfqvdjev) is now a manager.

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

    docker swarm join --token SWMTKN-1-5hvcwykkq2isx81vu0670mj2wsqigkya44gyib119ifyna7hsc-7ks2u
82bsih0r0xv6rwfyb94r 192.168.0.6:2377

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

Cela signifie que le nœud du gestionnaire est configuré avec succès.

- Maintenant, ajoutez un worker node en copiant la sortie de la command précèdent et collez la sortie sur le worker node :

Worker node 1(node2):

```
[node2] (local) root@192.168.0.7 ~
$ docker swarm join --token SWMTKN-1-2j9it8qga6ifydop6xktfqrlw6lnq77bx49ttofeht0q5dvmlg-3zr3tj9onetthwnklzutw74ai 192.168.0.8:2377
This node joined a swarm as a worker.
```

Worker node 2 (node3):

```
[node3] (local) root@192.168.0.6 ~
$ docker swarm join --token SWMTKN-1-2j9it8qga6ifydop6xktfqrlw6lnq77bx49ttofeht0q5dvmlg-3zr3tj9onetthwnklzutw74ai 192.168.0.8:2377
This node joined a swarm as a worker.
```

- Maintenant, revenez au nœud de gestionnaire et exécutez la commande suivante pour répertorier le nœud de travail :

```
docker node ls
```

Ici, vous devez voir le noeud worker dans la sortie suivante :

```
[node1] (local) root@192.168.0.8 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
zhld55ryp3jbuo52itttndvvu *   node1      Ready     Active         Leader           20.10.17
cb6r0t2b7oojd0wmk17udlpq8     node2      Ready     Active                          20.10.17
ss6q6ju9brrrir64limffk15w     node3      Ready     Active                          20.10.17
```

▼ Liste des tâches en cours d'exécution sur un ou plusieurs nœuds

Pour lister les tâches en cours d'exécution sur un ou plusieurs nœuds d'un cluster Docker Swarm, vous pouvez utiliser la commande `docker service ps` sur le nœud manager du cluster. Voici comment procéder :

1. Ouvrez une ligne de commande sur le nœud manager du cluster.
2. Lencer une tache dons les node:
3. Exécutez la commande suivante pour lister toutes les tâches en cours d'exécution sur tous les nœuds du cluster :

```
docker node ps [ID-node1 ID-node2 ... or node1-name node2-name]
```

```
$ docker node ps
ID        NAME      IMAGE     NODE      DESIRED STATE   CURRENT STATE   ERROR     PORTS
```

⚠️ en a pas encours lancer une tache c’est pourquoi en pas de tache afficher en encours d’exécution.

▼ Comment un worker node peut quitter un cluster

Pour supprimer « worker node (node 2)», exécutez la commande suivante depuis « worker node (node 2)» lui-même :

```
docker swarm leave
```

```
[node2] (local) root@192.168.0.7 ~
$ docker swarm leave
Node left the swarm.
```

▼ Comment supprimer un node de notre cluster

Si un worker nod quitter le swarm ,ce nœud apparaîtra toujours dans la liste des nœuds, et sera marqué comme étant `down`.

Dans node1 (manager):

```
[node1] (local) root@192.168.0.8 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
zhld55ryp3jbuo52itttndvvu *   node1      Ready     Active         Leader           20.10.17
cb6r0t2b7oojd0wmk17udlpq8     node2      Down      Active                          20.10.17
ss6q6ju9brrrir64limffk15w     node3      Ready     Active                          20.10.17
```

Il n'affecte plus le fonctionnement de l'ensemble de nœuds, mais une longue liste de nœuds `down` peut encombrer la liste des nœuds. Pour supprimer un nœud inactif de la liste, utilisez la commande `[node rm]`.

```
docker node rm [ID of you down node or it name]
```

```
[node1] (local) root@192.168.0.8 ~
$ docker node rm cb6r0t2b7oojd0wmk17udlpq8
cb6r0t2b7oojd0wmk17udlpq8
[node1] (local) root@192.168.0.8 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
zhld55ryp3jbuo52itttndvvu *   node1      Ready     Active         Leader           20.10.17
ss6q6ju9brrrir64limffk15w     node3      Ready     Active                          20.10.17
```

- Si vous essayez de supprimer un nœud actif (exemple node3), vous recevrez une erreur car node3 et en étate active :

  ```
  [node1] (local) root@192.168.0.8 ~
  $ docker node rm ss6q6ju9brrrir64limffk15w
  Error response from daemon: rpc error: code = FailedPrecondition desc = node ss6q6j
  u9brrrir64limffk15w is not down and can't be removed
  ```

  Supprimer de force un nœud inaccessible d'un swarm (--force)

  ```
  docker node rm --force [ID of you active node]
  ```

  ```
  [node1] (local) root@192.168.0.8 ~
  $ docker node rm --force ss6q6ju9brrrir64limffk15w
  ss6q6ju9brrrir64limffk15w
  ```

⚠️ Mais comment supprimer un manager node si en a un seule manager dans notre cluster.

▼ comment supprimer un manager node si en a un seule manager dans notre cluster.

1. Vous devrez d'abord ajouter un nouveau nœud "manager" avant de supprimer le nœud existant. Vous pouvez ajouter un nœud "manager" en utilisant la commande `docker swarm join-token manager` sur un nœud existant qui est déjà un "manager".

   dans le node1(manager):

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

```
[node1] (local) root@192.168.0.5 ~
$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-068cudzzwt6kcfhdm9l2gf7jldwufzufm3fb9q6mtsxpr5bwyv-bgk50eposdi8cfki88ke8vtye 192.168.0.5:237
7
```

2. Copier la résultate de la la commande précèdent dans la nouvel machine(node4)

```
[node4] (local) root@192.168.0.10 ~
$ docker swarm join --token SWMTKN-1-63m699o20cteoplefvkpa1b42f8oquh4cm6wmk4suiy64sdra5-dihl3pcm47gkl0zo6bsf3gfox 192.168.0.13:2377
This node joined a swarm as a manager.
```

maintenant node4 est un manager aussi.

3. Sur le node1 rétrograder cette machine en tant que worcker node en exécutant la commande suivante sur un autre nœud manager:

```
docker node demote [ID of you manager node or it name]
```

```
[node1] (local) root@192.168.0.13 ~
$ docker node demote node1
Manager node1 demoted in the swarm.
```

4. Une fois que le nœud leader a été rétrogradé en tant que nœud de travail, vous pouvez maintenant le retirer du cluster en exécutant la commande suivante sur un autre nœud manager (node4):

```
bashCopy code
docker node rm --force [ID of you first manager node1 or it name]
```

```
[node4] (local) root@192.168.0.10 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
3a628erlcby7bjei1ysyhi3pv     node1      Ready     Active                          20.10.17
xy29z4xk7x0v2mzmmvtsijnas *   node4      Ready     Active         Leader           20.10.17
```

```
[node4] (local) root@192.168.0.10 ~
$ docker node rm --force node1
node1
[node4] (local) root@192.168.0.10 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
xy29z4xk7x0v2mzmmvtsijnas *   node4      Ready     Active         Leader           20.10.17
```

▼ créer et manipuler les service dans un cluster.

La manipulation des services sur Docker Swarm se fait via la CLI (Command Line Interface) Docker. Voici quelques commandes de base pour manipuler les services :

▼ 1. Créer un service :

```
docker service create --name mon-service mon-image
```

Exemple:

```
sudo docker service create --name HelloWorld alpine ping docker.com
```

```
$ sudo docker service create --name HelloWorld alpine ping docker.com
amphao0q3bbmkf6kyrp5qjdkh
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
```

Cette commande crée un service nommé avec le nome "HelloWorld" basé sur l'image Docker "alphine".

▼ 2.Voir la liste des services :

```
docker service ls
```

```
$ docker service ls
ID             NAME         MODE         REPLICAS   IMAGE          PORTS
amphao0q3bbm   HelloWorld   replicated   1/1        alpine:latest
```

Cette commande liste tous les services qui ont été créés sur le Swarm.

▼ 3.Voir les détails d'un service :

```
docker service inspect mon-service
```

Exemple:

```
$ docker service inspect HelloWorld
[
    {
        "ID": "amphao0q3bbmkf6kyrp5qjdkh",
        "Version": {
            "Index": 249
        },
        "CreatedAt": "2023-05-02T12:03:37.466974749Z",
        "UpdatedAt": "2023-05-02T12:03:37.466974749Z",
        "Spec": {
            "Name": "HelloWorld",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image": "alpine:latest@sha256:124c7d2707904eea7431fffe91522a01e5a861a624ee31d03372cc1d138a3126",
                    "Args": [
                        "ping",
                        "docker.com"
                    ],
```

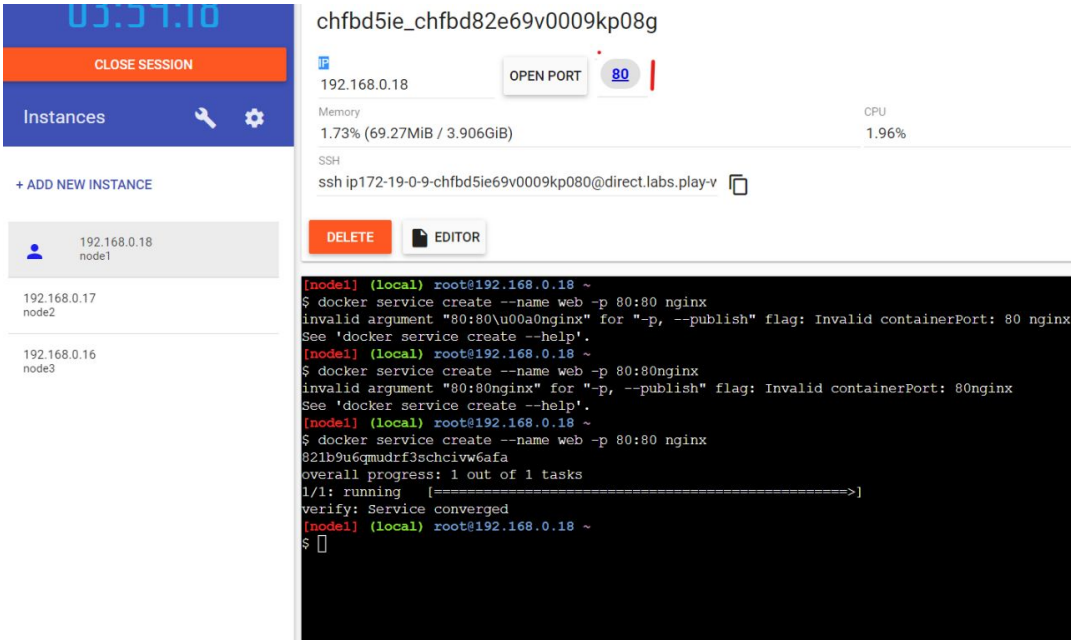
Cette commande affiche les détails du service "HellowWorld".

▼ exemple de WEB (NGINX):

Maintenant si on veut créer un service web

```
docker service create --name web -p 80:80 nginx
```

on remarque que le port 80 est ajouter

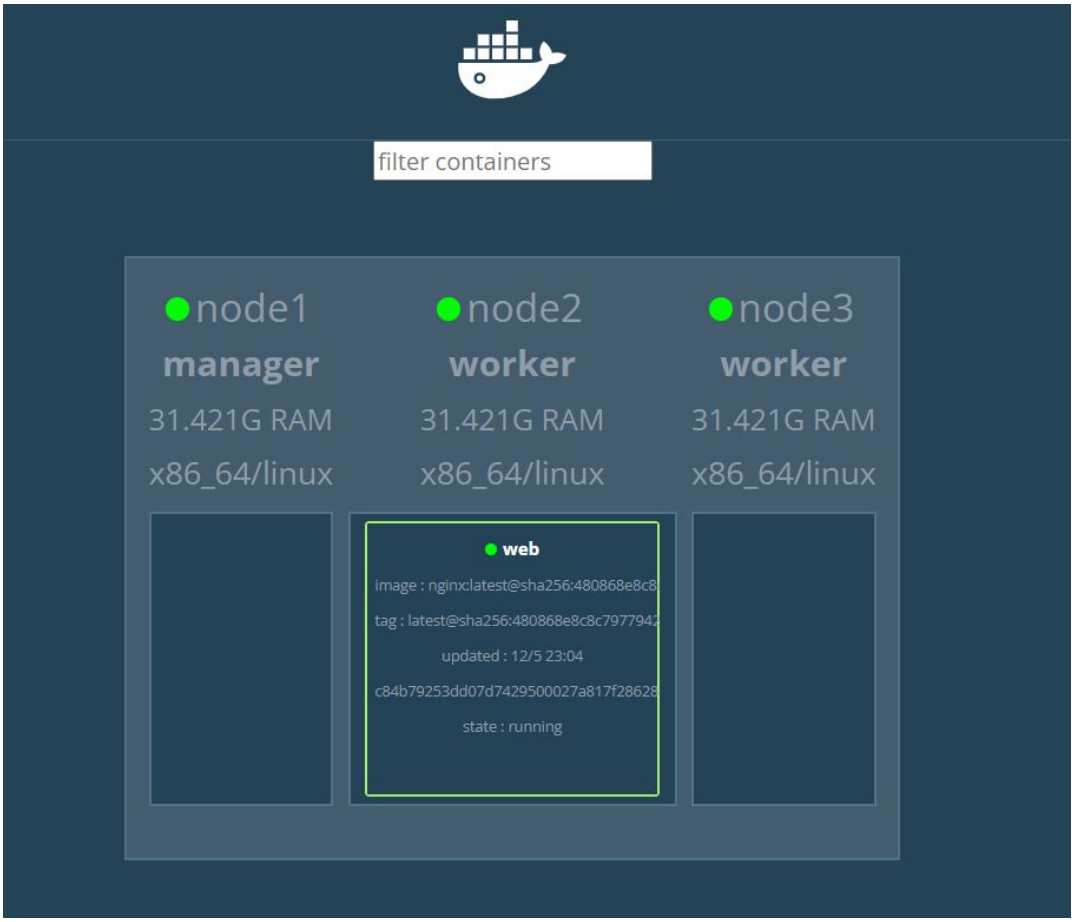


si on clique sur ce port on entre dans le WEB

Si on veut afficher le cluster graphiquement en utilisant virtualisation:

```
docker run -it -d -p 8080:8080 -v/var/run/docker.sock:/var/run/docker.sock dockersamples/visualizer
```

et maintenant on a le port 8080 est ajoute



▼ 4.Mettre à jour un service :

```
docker service update mon-service --image ma-nouvelle-image
```

Exemple:

```
$ docker service update HelloWorld --image ubuntu
HelloWorld
overall progress: 0 out of 1 tasks
1/1: preparing [==================================>                ]
```

Cette commande met à jour l'image du service "HellowWorld" en utilisant la nouvelle image "ubuntu".

▼ 5.lancer MULTI-service :

```
docker service rm mon-service
```

```
$ docker service ls
ID             NAME         MODE         REPLICAS   IMAGE           PORTS
amphao0q3bbm   HelloWorld   replicated   0/1        ubuntu:latest
[node4] (local) root@192.168.0.10 ~
$ docker service rm HelloWorld
HelloWorld
[node4] (local) root@192.168.0.10 ~
$ docker service ls
ID        NAME      MODE      REPLICAS   IMAGE     PORTS
```

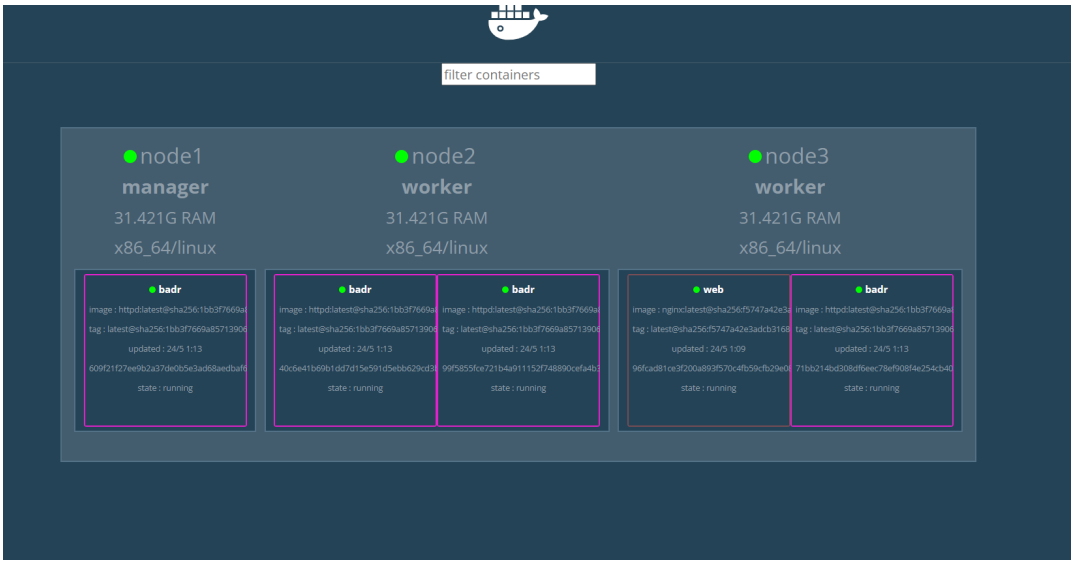
Cette commande supprime le service "HelloWorld".

si on veut plusieurs dupliquer un servise plusieurs fois il faut juste utiliser la commande suivante:

```
docker service create --name <service> --relicas <nbr de repete> -p <port> <image>
```

```
$ docker service create --name badr --replicas 4 -p 9090:80 httpd
pnjju7xo9nnza3lv4hbwgkc2s
overall progress: 4 out of 4 tasks
1/4: running   [==================================================>]
2/4: running   [==================================================>]
3/4: running   [==================================================>]
4/4: running   [==================================================>]
verify: Service converged
[node1] (local) root@192.168.0.8 ~
$
```

mais dans ce cas on remarque que le manager lance aussi des taches :



pour arrêter manager de travailler il faut utiliser la commande qui changer l’availibility de manager

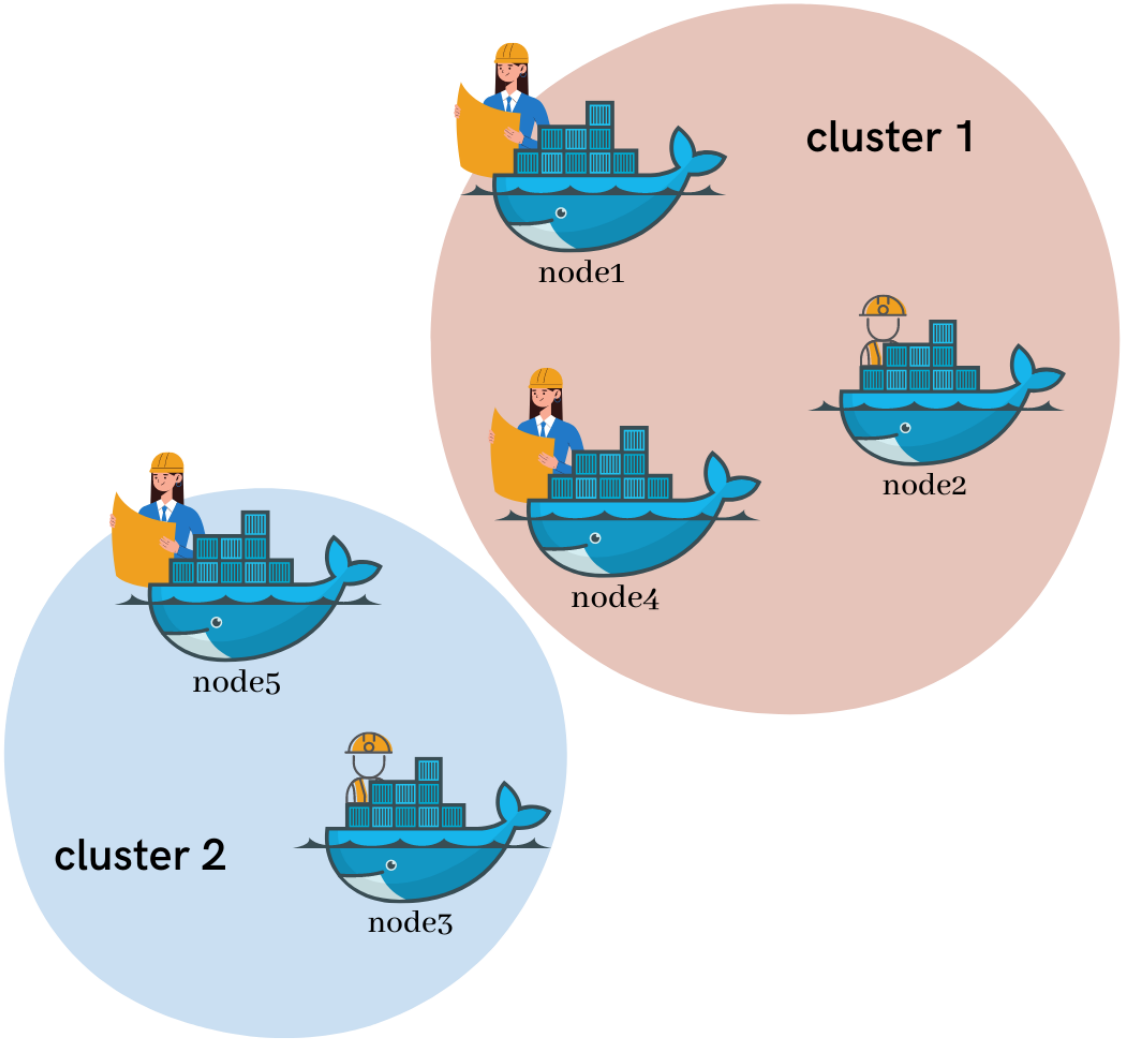
```
[node1] (local) root@192.168.0.8 ~
$ docker node ls
ID                            HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS   ENGINE VERSION
5uha9u2zpnxn6ti0xhjjuat8f *   node1      Ready    Active         Leader           20.10.17
a7fsvrct2f1n6eose3hol7v11     node2      Ready    Active                          20.10.17
7z818otdo833c1qvq3n1z67f2     node3      Ready    Active                          20.10.17
```

après l’utilisation de la commande on a :

```
[node1] (local) root@192.168.0.8 ~
$ docker node update --availability Drain node1
node1
[node1] (local) root@192.168.0.8 ~
$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
5uha9u2zpnxn6ti0xhjjuat8f *   node1      Ready     Drain          Leader           20.10.17
a7fsvrct2f1n6eose3hol7vl1     node2      Ready     Active                          20.10.17
7z818otdo833c1qvq3n1z67f2     node3      Ready     Active                          20.10.17
```

maintenant en remarque que la manager ne travaille pas

▼ Etape 3: TP générale:



Puis exécuter les commandes à votre choix (chez vous) de l’étape 2: