TP - Docker Networking

Dans ce Travaux Pratiques (TP), vous allez apprendre à configurer les réseaux Docker en créant des conteneurs dans différents modes de réseau. Vous commencerez par le réseau par défaut Bridge, puis vous explorerez les modes Host, Overlay et enfin Macvlan.

À la fin de ce TP, vous devrez comprendre comment :

1. Créer des conteneurs dans différents modes de réseau.
2. Comprendre comment les conteneurs communiquent entre eux.
3. Créer et configurer des réseaux personnalisés dans Docker.

Partie 1 : Mode Réseau Bridge

Objectif :

Comprendre comment fonctionne le réseau par défaut Bridge et comment les conteneurs communiquent dans ce mode.

Étapes :

1. **Démarrer un conteneur en mode Bridge** : Par défaut, Docker utilise le mode Bridge. Créez un conteneur exécutant le serveur web nginx.
   Commande

   ```
   docker run -d --name my-container nginx
   ```

   Essayez d'ouvrir votre navigateur et visiter http://localhost.

2. **Vérifier l'adresse IP du conteneur :** Vous pouvez obtenir l'adresse IP du conteneur dans le réseau bridge en exécutant la commande suivante :

   ```
   docker inspect my-container
   ```

   Cherchez le champ IPAddress dans les paramètres réseau.

3. **Accéder au conteneur depuis l'hôte :** Le conteneur s'exécute sur un réseau isolé, donc il n'est pas directement accessible depuis l'adresse IP de l'hôte. Vous devez exposer un port pour y accéder :

   Démarrez un conteneur tout en exposant le port 80 (HTTP) :

   ```
   docker run -d -p 8080:80 --name my-container-exposed nginx
   ```

   Vous pouvez maintenant ouvrir votre navigateur et visiter http://localhost:8080 pour voir le

serveur nginx en fonctionnement.

Par défaut, les conteneurs Docker utilisent le réseau bridge, ce qui les isole du réseau de l'hôte. Pour rendre un service accessible depuis l'extérieur, il est nécessaire d'exposer un port avec l'option -p.

4. **Lister les réseaux Docker :** Vous pouvez lister tous les réseaux Docker. Le réseau bridge est celui auquel les conteneurs sont connectés par défaut.

   ```
   docker network ls
   ```

5. **Créer un réseau Docker** : Vous pouvez créer un réseau Docker personnalisé pour connecter des conteneurs de manière isolée.

   ```
   docker network create —-driver bridge my-network
   ```

   bridge: est le type de driver, il peut être changé pour d'autres types de drivers en fonction des besoins, comme host ou overlay.

6. **Connecter un conteneur à un réseau**: Cette commande connecte un conteneur existant à un réseau spécifique.

   ```
   docker network connect my-network my-container
   ```

7. **Déconnecter un conteneur d'un réseau**: Cette commande déconnecte un conteneur d'un réseau spécifique.

   ```
   docker network disconnect my-network my-container
   ```

8. **Supprimer un réseau Docker**: Cette commande permet de supprimer un réseau Docker.

   ```
   docker network rm my-network
   ```

Partie 2 : Mode Réseau Host

Objectif :

Apprendre à exécuter des conteneurs en utilisant le réseau Host, qui partage la pile réseau de l'hôte.

Étapes :

Essayez d'abord d'arrêter le conteneur en cours d'exécution en utilisant les commandes **docker ps** et **docker stop <container_id>."**

1. **Démarrer un conteneur en mode Host** : En mode Host, le conteneur utilise la même interface réseau que l'hôte.

   ```
   docker run -d --name host-container --network host nginx
   ```

2. **Accéder au conteneur :** Le conteneur est maintenant directement accessible depuis l'IP de l'hôte sur le port 80 sans avoir besoin de l'exposer.

   Dans votre navigateur tapez:

   ```
   http://localhost
   ```

   Cela doit afficher la page par défaut de nginx, indiquant que le conteneur est directement accessible via le réseau de l'hôte.

Contrairement au mode Bridge, les conteneurs en mode Host n'ont pas besoin de port forwarding car ils partagent l'adresse IP de l'hôte.

Partie 3 : Mode Réseau Overlay (Swarm Mode)

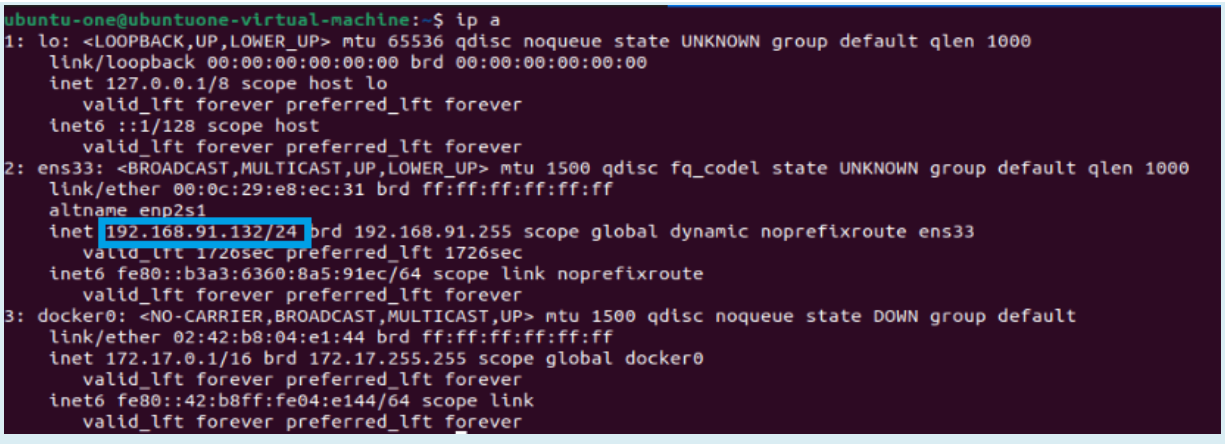
Objectif :

Apprendre à créer un réseau Docker Overlay pour permettre la communication entre des conteneurs sur différents hôtes.

Étapes :

1. **Initialiser Docker Swarm (si non déjà fait)** : Sur le premier hôte (VM1), initialisez Docker Swarm :

   ```
   docker swarm init --advertise-addr VM1-IP
   ```



VM1: Machine virtuelle 1

VM1: Machine virtuelle 2

VM1-IP: IP de la machine virtuelle, utiliser la commande **ip a** pour l'obtenir

2. **Rejoindre un autre hôte au Swarm** : Sur VM2, exécutez la commande fournie par docker swarm init sur VM1 pour rejoindre le Swarm en tant que nœud worker.

   ```
   docker swarm join --token <token> <VM1-IP>:2377
   ```

3. **Créer un réseau Overlay :** Sur VM1, créez un réseau Overlay :

   ```
   docker network create --driver overlay my-overlay-network
   ```

4. **Création des conteneurs:** Créer deux conteneur vm1-container sur VM1 et vm2-container sur VM2:

   ```
   docker run -dit --name vm1-container --network my-overlay-network busybox
   ```

   ```
   docker run -dit --name vm2-container --network my-overlay-network busybox
   ```

5. **Vérifier la communication :** Pour vérifier la communication entre les conteneurs sur différents hôtes:

   Sur VM1 exécutez cette commande pour pinguer un conteneur depuis un autre :

   ```
   docker exec -it <vm1-container> ping <vm2-container>
   ```

Partie 4 : Mode Réseau Macvlan

Objectif :

Apprendre à créer un réseau Macvlan qui permet aux conteneurs d'être traités comme des périphériques indépendants sur le réseau physique.

Étapes :

1. **Créer un réseau Macvlan** : Créez d'abord un réseau Macvlan. Assurez-vous de remplacer eth0 par le nom de votre interface réseau.

   ```
   docker network create --driver macvlan --subnet=192.168.1.0/24 --gateway=192.168.1.1 -o parent=eth0 my-macvlan-network —-attachable
   ```

   `--subnet`: le sous-réseau de votre réseau local (LAN)

   `--gateway`: l'adresse IP de votre routeur/passerelle (**`ip route | grep default`**)

   `parent=eth0`: l'interface réseau de l'hôte (vous pouvez la vérifier avec la commande ip a)

   `my-macvlan-network`: le nom de votre nouveau réseau Macvlan

```
ubuntu-one@ubuntuone-virtual-machine:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:e8:ec:31 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.91.132/24 brd 192.168.91.255 scope global dynamic noprefixroute ens33
       valid_lft 1180sec preferred_lft 1180sec
    inet6 fe80::b3a3:6360:8a5:91ec/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:b8:04:e1:44 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:b8ff:fe04:e144/64 scope link
       valid_lft forever preferred_lft forever
11: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:81:b8:32:d2 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
```

```
ubuntu-one@ubuntuone-virtual-machine:~$ ip route | grep default
default via 192.168.91.2 dev ens33 proto dhcp metric 100
```

```
ubuntu-one@ubuntuone-virtual-machine:~/Desktop$ sudo docker network create --driver overlay --subnet=192.168.91.0/24 --gateway=
192.168.91.2 -o parent=ens33 my-macvlan-network --attachable
w491eus27gzcw45jqo2rx90ku
```

2. **Exécuter des conteneurs sur le réseau Macvlan** : Exécutez deux conteneurs sur le réseau Macvlan :

   ```
   docker run -dit --network my-macvlan-network --name container1 busybox
   ```

   ```
   docker run -dit --network my-macvlan-network --name container2 busybox
   ```

   Vous pouvez utilisez la commande **docker network inspect my-macvlan-network** pour voir les adresses IP de chaque conteneur.

3. **Accéder aux conteneurs directement :** Chaque conteneur aura maintenant une adresse IP unique dans le sous-réseau 192.168.91.0/24, et vous pouvez y accéder en tant que périphériques individuels sur le réseau local.

   Testez l'accès en pingant un conteneur depuis un autre ou depuis une autre machine :

   ```
   docker exec -it container1 ping container2
   ```

   Vous pouvez également utiliser les adresses IP à la place des noms, par exemple : docker exec -it container1 ping <IP_de_container2>.