TP03 : Réseau

Création du réseau iut1

On entre la commande suivante :

```
docker network create --internal --subnet 10.0.1.0/28
```

On peut utiliser l'option `--internal` pour rendre le réseau inaccessible depuis l'extérieur.

L'option `--subnet` permet de définir la plage du sous réseau que l'on va créer.

Lorsque l'on essaye de créer un réseau `10.0.1.0/29`, cela ne fonctionne pas. En effet, `10.0.1.0/29` est inclus dans `10.0.1.0/28`, et docker ne permet de créer des sous-réseau, de sous-réseau.

Inspection du réseau

```
docker network inspect iut-internal
```

On peut voir, le nom, l'id, le driver utilisé (en l'occurrence bridge), le subnet (10.0.1.0/28), si le réseau est interne est ou pas (oui pour notre cas) et si le l'ipV6 est activé (non dans notre cas)

Network Drivers

Après avoir lancé la commande `docker info`, on peut voir les différent driver réseau :

- bridge
- host
- ipvlan
- macvlan
- null
- overlay

Communication entre deux conteneurs

Après avoir lancé les deux conteneurs, on constate que le conteneur ping-src ne trouve pas le nom hôte ping-dst, et pour cause, il ne sont pas connectés dans un réseau commun. Pour pallier à ça, on va les connecter dans le même réseau :

On crée un réseau :

```
docker network create ping-net
```

Ensuite on met nos deux conteneurs dans le réseau :

```
docker network connect ping-net ping-dst
docker network connect ping-net ping-src
```

On aurait aussi pu utiliser l'option **--link** lors de la création du conteneur, ce qui permet de connecter deux conteneurs. Cependant, la documentation déconseille d'utiliser cette option, et conseille de définir les réseau manuellement.

Étude des interfaces réseaux du conteneur et du système hôte

Après études des réseaux connectés sur notre conteneur **ping-dst** avec la commande `docker inspect ping-dst`. On constate qu'il y en a deux :

- **ping-net** : le réseau que l'on lui à associé à l'étape précédente
- **bridge** : le réseau par défaut, qui permet de faire le lien entre notre machine hôte et le conteneur docker.

Si on compare les adresses de notre machine et les adresses du conteneurs, on constate qu'ils ont deux réseaux en commun (172.17.0.0 et 172.18.0.0). Ces deux réseaux sont les deux réseaux du docker : le bridge et le ping-net.

On liste les bridges avec la commande `brctl show`, on constate que la commande chaque id commence par **800.0242**

On lance la commande route sur la machine hote et sur le conteneur docker : On constate que la machine virtuelle n'a qu'une route (autre que localhost) :

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         wifi-osiris-1.u 0.0.0.0         UG    0      0        0 eth0
172.17.0.0      *               255.255.0.0     U     0      0        0 eth0
```

La machine hote possède bien plus de route : y compris la route **172.17.0.0** pour joindre notre conteneur :

```
Destination     Passerelle      Genmask         Indic Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    0      0        0 ens18
10.0.1.0        0.0.0.0         255.255.255.240 U     0      0        0 br-
436b150e23b8
10.0.8.0        0.0.0.0         255.255.254.0   U     0      0        0 ens18
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
172.18.0.0      0.0.0.0         255.255.0.0     U     0      0        0 br-
d3e8d1fd8aab
```

Fournir un accès au conteneur depuis l'hôte

Création de deux serveurs nginx :

```
docker run --rm --name nginx80 -p 80 nginx
docker run --rm --name nginx80 -p 8080:80 nginx
```

Ainsi, nous avons deux serveurs nginx joignable sur le port 80 et 8080 de la machine hôte.

Le serveur nginx est aussi accessible à l'intérieur d'autres conteneurs : l'adresse est simplement l'adresse du docker nginx.

Tester le routage des réseaux (traceroute)

Lancement de la commande `traceroute tr-2` :

```
traceroute to tr-2 (10.1.0.3), 30 hops max, 46 byte packets
 1  tr-2.trace-net (10.1.0.3)  0.052 ms  0.025 ms  0.024 ms
```

On peut en déduire qu'il est passé par le routeur du réseau virtuel **trace-net**, par l'interface **eth1**.

Lancement de la commande `traceroute 172.17.0.3` :

```
traceroute to 172.17.0.3 (172.17.0.3), 30 hops max, 46 byte packets
 1  172.17.0.3 (172.17.0.3)  0.033 ms  0.026 ms  0.023 ms
```

On peut en déduire qu'il est passé par le routeur virtuel de notre machine hote, par **eth0**.

Après avoir éteint un premier conteneur, ajouté un second conteneur, puis rallumé le premier, on constate que le réseau réatribue au premier conteneur son ancienne adresse.