Labo 8 - Le réseau, partie 2

Il existe 5 types de drivers réseaux que nous pouvons utiliser lors de la création d'un conteneur en ajoutant cet argument

```
--network <type>
```

Driver 1 : none

Absence totale de réseau au conteneur

```
sudo docker run -ti --rm --network none bash
```

```
[nancre@DOCKER ~]$ sudo docker run -ti --rm --network none bash
[sudo] password for nancre:
bash-5.0# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

bash-5.0#
```

Même en spécifiant un port avec -p il n'y aura pas de communication réseau

Driver 2 : bridge

Iil est le pilote par défaut. Il peut être utilisé pour faire communiquer plusieurs conteneurs entre eux quand c'est spécifié

Nous avons des adresses ip et nous voyons que les conteneurs se voient

```
[nancre@DOCKER ~]$ sudo docker run -ti --rm bash
bash-5.0# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:516 (516.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

bash-5.0# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3): 56 data bytes
64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.180 ms
64 bytes from 172.17.0.3: seq=1 ttl=64 time=0.069 ms
64 bytes from 172.17.0.3: seq=2 ttl=64 time=0.069 ms
^C
--- 172.17.0.3 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.069/0.106/0.180 ms
bash-5.0#
```

nancre@DOCKER:~

```
nancre@DOCKER ~]$ sudo docker run -ti --rm bash
sudo] password for nancre:
ash-5.0# ifconfig
th0       Link encap:Ethernet  HWaddr 02:42:AC:11:00:03
          inet addr:172.17.0.3  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:516 (516.0 B)  TX bytes:0 (0.0 B)

o         Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ash-5.0# ping 172.17.0.2
ING 172.17.0.2 (172.17.0.2): 56 data bytes
4 bytes from 172.17.0.2: seq=0 ttl=64 time=0.099 ms
4 bytes from 172.17.0.2: seq=1 ttl=64 time=0.074 ms
4 bytes from 172.17.0.2: seq=2 ttl=64 time=0.072 ms
C
-- 172.17.0.2 ping statistics ---
 packets transmitted, 3 packets received, 0% packet loss
ound-trip min/avg/max = 0.072/0.081/0.099 ms
ash-5.0#
```

Les deux conteneurs peuvent également communiquer avec le réseau de l'hôte qui a par défaut l'adresse ip 172.17.0.1

nancre@DOCKER:~

```
bash-5.0# ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.170 ms
64 bytes from 172.17.0.1: seq=1 ttl=64 time=0.076 ms
64 bytes from 172.17.0.1: seq=2 ttl=64 time=0.070 ms
^C
--- 172.17.0.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.070/0.105/0.170 ms
bash-5.0#
```

nancre@DOCKER:~

```
bash-5.0# ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.110 ms
64 bytes from 172.17.0.1: seq=1 ttl=64 time=0.075 ms
64 bytes from 172.17.0.1: seq=2 ttl=64 time=0.074 ms
^C
--- 172.17.0.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.074/0.086/0.110 ms
bash-5.0#
```

Maintenant chaque conteneur créé pourra communiquer avec un autre grâce au bridge par défaut. Il est cependant possible d'isoler les réseaux des conteneurs en créant un clone du bridge

```
sudo docker network create --driver=bidge mon_bridge
```

Puis un

```
sudo docker network ls
```

nancre@DOCKER:~

```
[nancre@DOCKER ~]$ sudo docker network create --driver=bridge mon_bridge
bb3f24d3eb274af6f98d2e7a4c53ead520b42560b0b5807fc77c172358270bf3
[nancre@DOCKER ~]$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
d9334d3a443e        bridge              bridge              local
ec32957262d1        host                host                local
bb3f24d3eb27        mon_bridge          bridge              local
825be2d219c5        none                null                local
[nancre@DOCKER ~]$
```

Voyons comment créer des conteneurs sur ce nouveau bridge

```
sudo docker run -ti --rm --network=mon_bridge --name=srv1 bash
```

Puis

```
ifconfig
```

Et pour le deuxième conteneur

```
sudo docker run -ti --rm --network=mon_bridge --name=srv2 bash
```

Puis

```
ifconfig
```

nancre@DOCKER:~

```
[nancre@DOCKER ~]$ sudo docker run -ti --rm --network=mon_bridge --name=srv1 bash
bash-5.0# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:02
          inet addr:172.18.0.2  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1102 (1.0 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

bash-5.0#
```

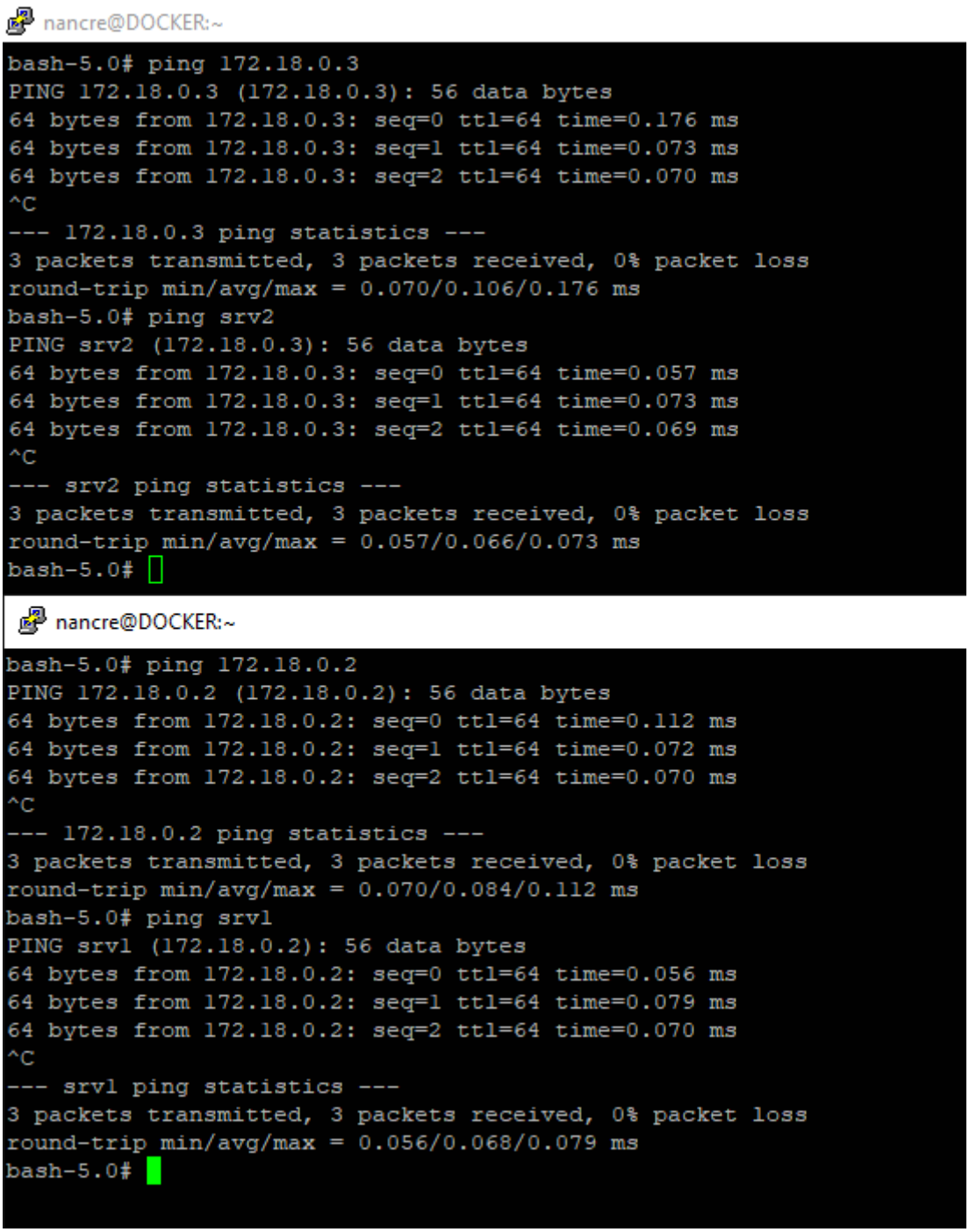
nancre@DOCKER:~

```
[nancre@DOCKER ~]$ sudo docker run -ti --rm --network=mon_bridge --name=srv2 bash
[sudo] password for nancre:
bash-5.0# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:03
          inet addr:172.18.0.3  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:516 (516.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

bash-5.0#
```

Le DNS interne géré par Docker ne s'applique uniquement qu'aux réseaux bridges clonés



Maintenant si les conteneurs sont rattachés au bridge par défaut, il est possible de les rattacher dynamiquement à un bridge cloné

```
sudo docker network connect=mon_bridge srv1
```

et

```
sudo docker network connect=mon_bridge srv2
```

Pour supprimer un bridge cloné, il suffit de faire

```
sudo docker network rm mon_bridge
```

nancre@DOCKER:~

```
[nancre@DOCKER ~]$ sudo docker network rm mon_bridge
[sudo] password for nancre:
mon_bridge
[nancre@DOCKER ~]$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
d9334d3a443e        bridge              bridge              local
ec32957262d1        host                host                local
825be2d219c5        none                null                local
[nancre@DOCKER ~]$
```

> Cette opération ne peut se faire si nous n'avons plus aucun conteneur attaché à ce réseau

Driver 3 : host

Il permet à un conteneur de partager la même pile réseau que celle de l'hôte

Par exemple avec nginx nous avions spécifié un port particulier mais si nous faisons simplement

```
sudo docker run --rm --network=host nginx
```

On a bien accès au serveur nginx via un navigateur web
L'accès au network host est assez peu courant

Driver 4 : overlay

Il permet de connecter entre eux qui tournent sous des hôtes différents

Driver 5 : Macvlan

Il permet d'attribuer une adresse MAC à l'interface réseau virtuelle de chaque conteneur

---

Revision #1
Created 2026-04-24 13:17:45 UTC by Nico là
Updated 2026-04-24 13:17:53 UTC by Nico là