Le réseau

Le réseau avec docker network

Docker permet une gestion avancée du réseau, entre un conteneur et la machine hôte, mais aussi entre les conteneurs.

Les réseaux docker peuvent être gérés avec la commande `docker network`.

```
docker network ls # lister les réseaux
docker network create mon_reseau # créer un réseau
docker network rm mon_reseau # supprimer un réseau
```

Il existe plusieurs types de réseaux docker, identifiés avec l'argument `--driver`.

- **Bridge (pont)**
  Utilisé par défaut, c'est un réseau interne à docker.
  Les conteneurs peuvent communiquer entre eux sur ce réseau.
- **Host**
  Le conteneur partage directement le réseau de l'hôte. Il n'est pas isolé et utilise l'adresse IP de l'hôte. Dans ce cas, la redirection de port n'est pas nécessaire.
- **None**
  Aucun réseau, isolation totale.

Docker propose par défaut un réseau pour chacun de ses 3 types.

```
debian@debian:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
e5ee2347d562   bridge    bridge    local
4b13b0e0f0a2   host      host      local
82c013605f89   none      null      local
```

Le réseau par défaut est `bridge`.

Driver Bridge

Ci-dessous un exemple avec l'image `busybox` qui embarque quelques utilitaires réseau, dont la commande `ping`.

```
debian@debian:~$ docker network create --driver=bridge testnetwork
f61e66fb4a3d13dc2713ec7d8d38cea1a68a7de519cd4d96a3fd655e93dcf23d
debian@debian:~$ docker run -tid --name=c1 --network=testnetwork busybox sh
495db35d7ebda8523b9b3b00746fe2b986b7b59a2d9d73d18cffb250d9cb4098
debian@debian:~$ docker run -ti --name=c2 --network=testnetwork busybox sh
/ # ping c1
PING c1 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.153 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.119 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.110 ms
64 bytes from 172.18.0.2: seq=3 ttl=64 time=0.147 ms
^C
--- c1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.110/0.132/0.153 ms
```

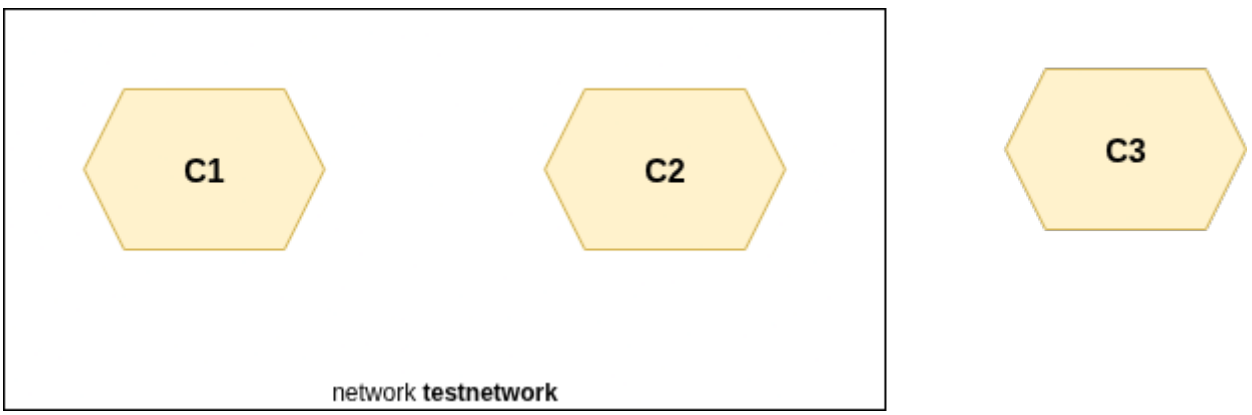
- Un réseau `testnetwork` est créé avec le driver `bridge`.
- Un conteneur `c1` est lancé en arrière-plan, il est attaché au réseau `testnetwork` avec `--network=testnetwork`.
- Un deuxième conteneur `c2` est lancé. Les deux conteneurs peuvent se pinguer.

> Chaque conteneur est visible dans le réseau à partir de son nom défini avec `--name`.

> Le réseau `bridge` par défaut de Docker isole les conteneurs entre-eux. Il est nécessaire de créer un autre réseau avec le driver `bridge` pour répéter l'expérience ci-dessus.

Un conteneur situé en dehors du réseau `testnetwork` ne peut pas pinguer `c1` et `c2` :

```
debian@debian:~$ docker run -ti --name=c3 busybox sh
/ # ping c1
ping: bad address 'c1'
/ # ping c2
ping: bad address 'c2'
```



Driver Host

Un conteneur lancé avec le driver `host` n'est pas isolé du réseau de l'hôte. Les ports ouverts sur le conteneur sont ouverts sur l'hôte.

Par exemple, un conteneur lancé à partir de l'image strm/helloworld-http écoutera à partir du port `80` de l'hôte, même sans redirection avec `--port`.

```
debian@debian:~$ docker run -ti --network=host strm/helloworld-http
Serving HTTP on 0.0.0.0 port 80 ...
```

> Le nom d'hôte du conteneur est affiché. Il est hérité depuis la machine hôte.

Gestion des réseaux

Il est possible de connecter un conteneur à un réseau existant avec `docker network connect`.

```
debian@debian:~$ docker network create --driver=bridge reseau2
562ad4a6e8b3105f619bd0deb30b89d5aaddde71babd6422ad912ccfa327687d
debian@debian:~$ docker run -tid --name=c5 ubuntu
f5f93155c9e02fc6a56d40b6d1bf5d0f3cd5c73106c72882f8878038398fe158
debian@debian:~$ docker network connect reseau2 c5
```

Le détail d'un réseau peut être affiché avec la commande `docker network inspect`.

```
debian@debian:~$ docker network inspect reseau2
[
    {
        "Name": "reseau2",
        "Id": "562ad4a6e8b3105f619bd0deb30b89d5aaddde71babd6422ad912ccfa327687d",
        "Created": "2025-08-29T22:34:59.617521484+02:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "f5f93155c9e02fc6a56d40b6d1bf5d0f3cd5c73106c72882f8878038398fe158": {
                "Name": "c5",
                "EndpointID": "f71251f1d3c4d9be09376f687898d22a984017b91f6da6c8fe75fa65dedba007",
                "MacAddress": "5a:76:dd:aa:62:15",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Un réseau `bridge` embarque une configuration IP, de la même manière qu'un réseau local.
Le réseau est défini par un `subnet` et il possède une IP `gateway` attribuée à la machine hôte.

Dans l'exemple ci-dessus, le réseau `reseau2` possède le subnet `172.19.0.0` avec un masque de sous-réseau `255.255.0.0` (ou `/16`).
L'adresse IP de la machine hôte est `172.19.0.1` et celle du conteneur `c5` est `172.19.0.2`.

Revision #17
Created 16 July 2025 09:42:00 by Thibaud FRICHET
Updated 18 September 2025 08:35:53 by Thibaud FRICHET