Réseau Docker

Frédéric Lassabe

FEMTO-ST - DISC Department - AND Team

Conteneur Docker

- Conteneur docker
  - Une instance d'exécution d'une image docker
  - Une image : autant de conteneurs que souhaité
  - Attention aux ports des services !
- Commandes de gestion des conteneurs
  - Commencent majoritairement par `docker container`

    `docker container run -dt nom_image`

Identification du conteneur

- À la création d'un conteneur :
  - Un identifiant universel UUID lui est attribué
  - UUID sert à identifier le conteneur parmi les autres
- Exemple :

```
ktossou@d4sh:~$ docker container run -dt -p 80:80 nginx
5c0391a97b91dc7eb80b423b1a0bd2b4ac78835a5ed7773d9c97c8e1704fc901
```

- UUID difficile à retenir
  - Possibilité de nommer un CT à sa création
  - Par défaut, docker génère un nom (à partir de deux mots)

Identification du conteneur

Exemple de nom généré aléatoirement par docker

```
ktossou@d4sh:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS
            PORTS                                NAMES
5c0391a97b91   nginx     "/docker-entrypoint.…"   About a minute ago   Up About
 a minute   0.0.0.0:80->80/tcp, :::80->80/tcp   priceless_kirch
```

Identification du conteneur

- Option `--name` : permet de choisir le nom du conteneur
- Syntaxe :

```
docker container run --name NOM_CONT NOM_IMAGE
```

- Exemple d'utilisation :

```
ktossou@d4sh:~$ docker container run --name mynginx -dt -p 800:80 nginx
f3adf1b0f56d5eed344d6b5a1d98d9b690eb2d28ab1159031fd7f3a92021d989
ktossou@d4sh:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
  PORTS                                  NAMES
f3adf1b0f56d   nginx     "/docker-entrypoint.…"   9 seconds ago   Up 7 seconds
  0.0.0.0:800->80/tcp, :::800->80/tcp    mynginx
5c0391a97b91   nginx     "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes
  0.0.0.0:80->80/tcp, :::80->80/tcp      priceless_kirch
```

Identification du conteneur

- Remplacer la commande du conteneur
- (si CMD)
- Exemple :

```
docker container run -d nginx sleep 500
```

- Explication :
  - À la place de nginx (proc. par défaut de l'image **nginx**)
  - Exécute `sleep 500` (attente de 500 secondes)

Mode attaché et détaché

- Un conteneur peut démarrer en mode attaché (par défaut)

```
ktossou@d4sh:~$ docker container run --name attached_nginx nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perf
orm configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-defa
ult.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.
d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/con
f.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.
sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.
sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

- Ou détaché (option `-d`)

```
ktossou@d4sh:~$ docker container run --name detached_nginx -d nginx
c9013a761957de43b1b10fa08aca5b5318518ed1e65d52161c87ce0268e74982
```

- Mode détaché : utile pour ne pas bloquer le terminal sur la sortie du CT

Arrêt/Redémarrage d'un conteneur docker

- On peut arrêter un conteneur

```
docker container stop CONTAINER
```

- Le redémarrer

```
docker container start CONTAINER
```

- Le supprimer

```
docker container rm CONTAINER
```

- CONTAINER désigne
  - Soit l'UUID du conteneur
  - Soit son nom

Start/stop : quelques commandes utiles

- Voir tous les conteneurs (arrêtés et en cours)

```
docker container ls -a
```

- Arrêter tous les conteneurs (démarrer : remplacer stop par start)

```
docker container stop $(docker container ls -aq)
```

- Supprimer tous les conteneurs

```
docker container rm $(docker container ls -aq)
```

- Exemple :

```
ktossou@d4sh:~$ docker container ls -aq
c9013a761957
```

Exécuter une commande dans un conteneur

- Dans un conteneur en cours d'exécution
- Utiliser la commande `exec` suivie d'une commande

```
docker container exec OPTIONS CONTAINER COMMAND
```

- Exécute dans le conteneur avec :

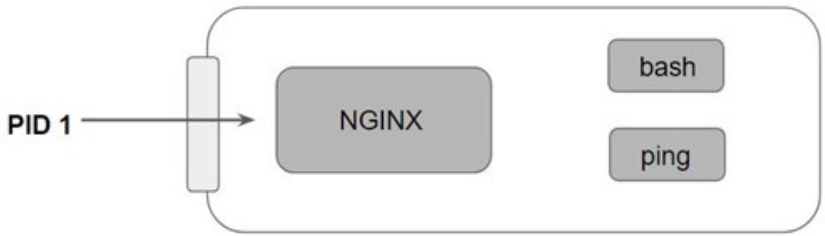
  OPTIONS représente les options utilisées dans la commande
  CONTAINER représente l'identifiant ou le nom du conteneur
  COMMAND représente la nouvelle commande à exécuter

- Liste des options :

```
docker container exec --help
```

Exécuter une commande dans un conteneur

- exec n'est exécuté que dans un conteneur démarré
- exec n'est pas relancé si on relance le CT (ne fait pas partie de l'image)



Nginx Docker Container

```
ktossou@d4sh:~$ docker container exec -t mynginx ls
bin   docker-entrypoint.d   home   lib64   mnt   root  srv  usr
boot  docker-entrypoint.sh  lib    libx32  opt   run   sys  var
dev   etc                   lib32  media   proc  sbin  tmp
```

Entrées sorties sur conteneur

- En combinant 2 options :
  - `-t` alloue un pseudo terminal au conteneur
  - `-i` garde stdin ouvert sur le conteneur
- Les deux combinés permettent un terminal interactif dans le conteneur

```
ktossou@d4sh:~$ docker container exec -it mynginx bash
root@f4977822d4ed:/# ls
bin   docker-entrypoint.d   home   lib64   mnt   root  srv  usr
boot  docker-entrypoint.sh  lib    libx32  opt   run   sys  var
dev   etc                   lib32  media   proc  sbin  tmp
root@f4977822d4ed:/#
```

Mappage de port

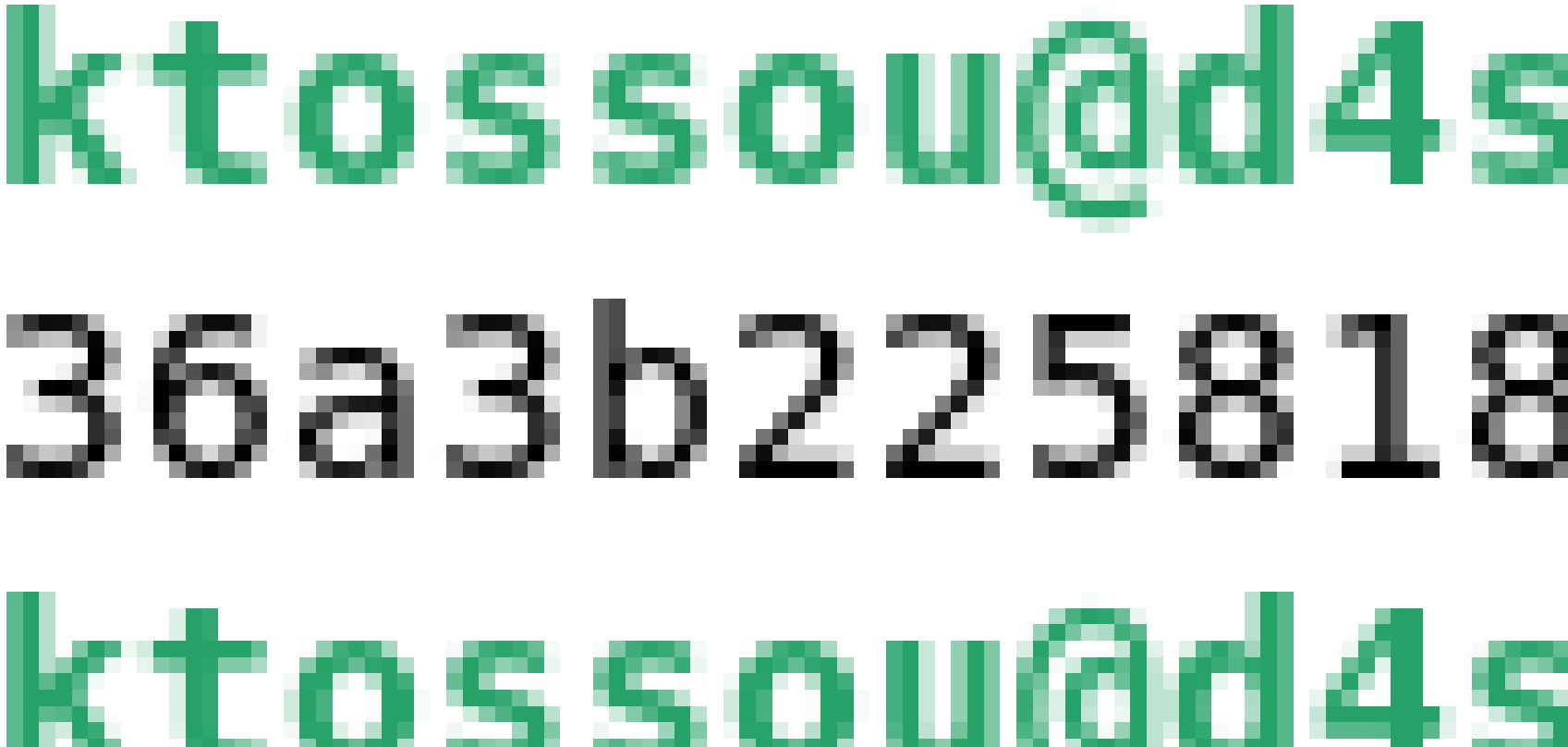
- Par défaut, les ports des conteneurs ne sont pas accessibles
- Sauf par l'hôte qui est dans un réseau avec les conteneurs
- Permettre l'accès externe nécessite un mappage :
  - Un port inutilisé de l'hôte
  - Vers le port ouvert par le service sur le conteneur
- Utilise l'option `-t`
- Par exemple :

```
docker container run -p 8080:80 nginx
```

  - Mappe le port 8080 de l'hôte
  - Sur le port 80 (web non chiffré) du CT nginx
- Également appelée liste de publication (ne publie que les ports spécifiés)

Mappage de tous les port

- Utiliser les ports exposés
- Les publie tous sur des ports aléatoires de l'hôte
- Option `-P`
- Exemple :



Politiques de redémarrage

- Par défaut, un conteneur démarre une fois et s'arrête
  - S'il termine son processus de CMD/ENTRYPOINT
  - Si le service docker est arrêté/redémarré
  - S'il crashe
- On peut spécifier une politique de démarrage
  - Option `--restart <valeur>` de `docker container run`
  - Valeur peut être :
    - **no** : pas de redémarrage (option par défaut)
    - **on-failure** : redémarre sur erreur avec code de sortie non nul
    - **unless-stopped** : redémarrer sauf si on l'arrête explicitement (ou docker redémarre)
    - **always** : redémarre toujours

Informations systèmes docker

- Commande pour obtenir des informations système
- Commande `docker system COMMAND OPTIONS`
- Avec `COMMAND` :
  - **events** liste en temps réel des événements qui se produisent sur le serveur docker
  - **prune** efface les données non utilisées
  - **info** affiche les informations systèmes de docker
  - **df** affiche l'utilisation de l'espace disque par docker

Informations systèmes : exemple

```
ktossou@d4sh:~$ docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          3         2         190.9MB   13.26kB (0%)
Containers      2         2         1.122kB   0B (0%)
Local Volumes   0         0         0B        0B
Build Cache     0         0         0B        0B
ktossou@d4sh:~$
```

```
ktossou@d4sh:~$ docker system df -v
Images space usage:

REPOSITORY    TAG       IMAGE ID       CREATED         SIZE      SHARED SIZE   UNIQUE SIZE   CONTAINERS
nginx         latest    61395b4c586d   2 weeks ago     187MB     0B            186.6MB       1
busybox       latest    a416a98b71e2   2 months ago    4.26MB    0B            4.262MB       1
hello-world   latest    9c7a54a9a43c   5 months ago    13.3kB    0B            13.26kB       0

Containers space usage:

CONTAINER ID   IMAGE     COMMAND                  LOCAL VOLUMES   SIZE      CREATED       STATUS
              NAMES
71ccad1f5ecb   busybox   "sh"                     0               0B        4 hours ago   Up 4 hours
              docker-exec
f4977822d4ed   nginx     "/docker-entrypoint.…"   0               1.12kB    5 hours ago   Exited (0) 31
minutes ago   mynginx

Local Volumes space usage:

VOLUME NAME   LINKS     SIZE

Build cache usage: 0B

CACHE ID   CACHE TYPE   SIZE      CREATED   LAST USED   USAGE     SHARED
ktossou@d4sh:~$
```

Le réseau docker

- Docker
  - Déployer des conteneurs
  - Environnement cloud
  - Donc pour des services distants
  - ⇒ réseau
- Après installation, une interface docker0 :

```
ktossou@d4sh:~$ ip addr
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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:37:4a:1d brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 62962sec preferred_lft 62962sec
    inet6 fe80::c021:390:87cb:207f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f7:66:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.2/24 brd 192.168.56.255 scope global noprefixroute enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::17a8:9554:4e99:433a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:11:bf:75:62 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::42:11ff:febf:7562/64 scope link
```

Le réseau docker

- Sous système réseau de docker
- Basé sur des pilotes
- En tant que plugins
- Pilotes par défaut de docker :
  - bridge
  - overlay
  - host
  - IPvlan
  - MACvlan
  - none

Le réseau Docker

- On utilise la commande `docker network COMMAND` pour gérer le réseau
- COMMAND peut avoir une des valeurs suivantes :
  - **inspect** : Affiche des informations détaillées sur un ou plusieurs réseaux
  - **ls** : Liste les différents réseaux Docker sur la machine
  - **connect** : Connect un conteneur à un réseau Docker
  - **disconnect** : Déconnecte un conteneur d'un réseau Docker
  - **create** : Crée un réseau Docker
  - **prune** : Supprime les réseaux non utilisés
  - **rm** : Supprime un ou plusieurs réseaux
- Connecter un conteneur à un réseau (à l'exécution) : `-network NOM_RESEAU`

Le réseau Docker

Exemple :

```
ktossou@d4sh:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
986fc91396b2   bridge    bridge    local
2232e0070e4e   host      host      local
11a3d662d6e0   none      null      local
ktossou@d4sh:~$
```

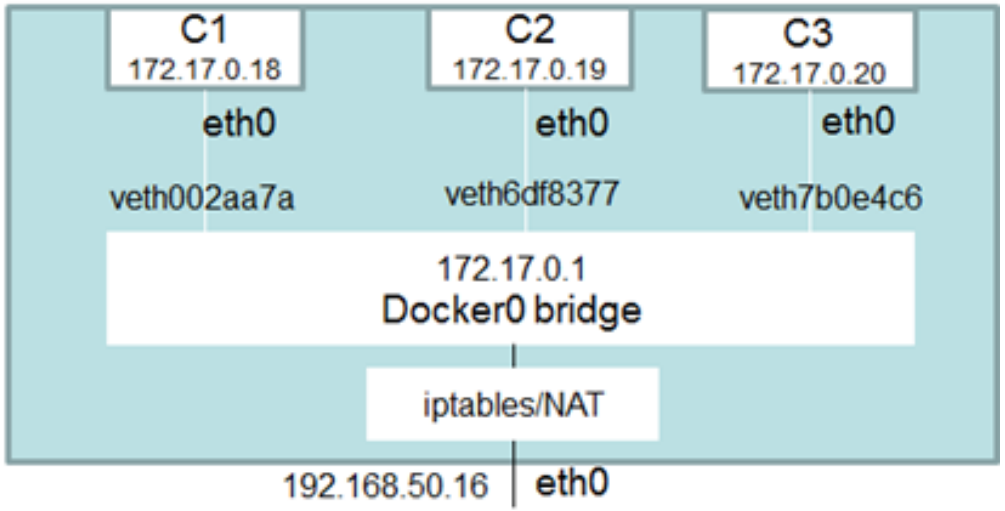
```
ktossou@d4sh:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "986fc91396b274322dfe72fad987ed37ab479c4dfd43b8806616e8c02ff62edd"
,
        "Created": "2023-10-12T10:23:01.359896563+02:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
```

Le pilote réseau Bridge

- Pour docker, réseau pont (*bridge*) utilise un pont logiciel
- Isole les conteneurs dans le même pont du reste des conteneurs
- Analogue à réseau interne de VirtualBox

Le pilote réseau Bridge

- Démarrage de Docker
  - Création automatique d'un bridge
  - Connexion par défaut des conteneurs créés



Le pilote réseau Bridge

- L'utilisateur de docker peut créer d'autres ponts
- Avantages des ponts créés explicitement :
  - DNS automatique entre les conteneurs (système de noms entre conteneurs)
  - Seuls les CT affectés à ce pont y seront (sécurité)
  - Possibilité d'attacher et détacher les CT à chaud
- Configuration du pont à sa création
- CT sur un pont publient les ports entre eux
- Ne fonctionne qu'avec des CT sur le même daemon docker

Le pilote réseau Overlay

- Pilote overlay
  - Crée un réseau distribué
  - Entre des hôtes du service Docker
- Overlay car crée une surcouche au dessus du vrai réseau
- Réseau overlay chiffré pour assurer la sécurité
- Docker gère le routage entre hôtes et conteneurs
- Commande `docker network create`

Le pilote réseau Overlay

- Pré-requis pour créer un réseau Overlay
- Ouverture des ports suivants :
  - Port 2377/TCP pour la communication de la gestion des clusters
  - Port 7946/TCP et UDP pour la communication entre les noeuds
  - Port 4789/UDP pour le trafic du réseau Overlay
- Initialiser ou rejoindre un *swarm* Docker
- `docker swarm init` ou `docker swarm join`

Le pilote réseau Overlay

- Création d'un réseau overlay :

```
docker network create -d overlay my-overlay
```

- Permettre à des CT indépendants de s'y attacher :

```
docker network create -d overlay --attachable \
my-attachable-overlay
```

- On peut aussi configurer les paramètres IP

Le pilote réseau Host

- Pilote qui supprime l'isolation réseau hôte/conteneur
- Accès direct au réseau de l'hôte
- Les ports ouverts sur les CT sont vus comme s'ils étaient à l'hôte
- Pas d'IP pour le conteneur
- Peut optimiser les performances
  - Notamment avec beaucoup de ports publiés
  - Ne nécessite pas de passer par une NAT interne
- Pilote uniquement disponible sous Docker pour Linux

Le pilote réseau Macvlan

- Certaines applications
  - Anciennes
  - Ou agissant sur le réseau
- Besoin d'un accès direct
- Pilote MACvlan
  - Affecte une adresse MAC à l'interface virtuelle du conteneur
  - Applications du conteneur croient à une vraie interface
- Une interface physique doit être associée au MACvlan
  - Configurer son sous-réseau
  - Et sa passerelle

Le pilote réseau Macvlan

- Avec plusieurs interfaces physiques
  - On peut isoler plusieurs MACvlan
- MACvlan peut dégrader l'accès réseau (saturation des tables d'adresses MAC)
- L'équipement connecté à l'hôte doit accepter plusieurs MAC par port
- Si un autre pilote peut fournir l'accès réseau, préférer ce dernier

Le pilote réseau Macvlan

- Options importantes du MACvlan
  - **macvlan_mode** : définit le mode d'opération du macvlan qui peut être `bridge` (défaut), `vepa`, `passthru`, `private`.
  - **parent** : Spécifie l'interface parente à utiliser
- Un MACvlan peut être en mode ***bridge*** ou ***802.1Q trunk bridge***
  - Mode ***bridge***, le trafic Macvlan passe par un périphérique physique sur l'hôte
  - Mode ***802.1Q trunk bridge***, le trafic passe par une sous-interface 802.1Q (VLAN) que Docker crée à la volée.

Le pilote réseau Macvlan (*bridge*)

- Utilisation de la commande
  `docker network create --driver macvlan`
- Spécifier également l'interface physique parente
- Exemple :

```
docker network create -d macvlan \
 --subnet=172.16.86.0/24 \
 --gateway=172.16.86.1 \
 -o parent=eth0 pub_net
```

Le pilote réseau Macvlan (*bridge*)

- Si des adresses sont réservées sur le réseau
- On peut les exclure des adresses allouées aux CT
- Option `--aux-address`
- Exemple :

```
docker network create -d macvlan \
 --subnet=192.168.32.0/24 \
 --ip-range=192.168.32.128/25 \
 --gateway=192.168.32.254 \
 --aux-address="my-router=192.168.32.129" \
 -o parent=eth0 macnet32
```

Le pilote réseau Macvlan (*802.1Q*)

- Pour passer en mode dot1Q
- Nommer une sous-interface comme parente
- Gestion automatique par Docker
- Exemple :

```
1  docker network create -d macvlan \
2     --subnet=192.168.50.0/24 \
3     --gateway=192.168.50.1 \
4     -o parent=eth0.10 macvlan10
```

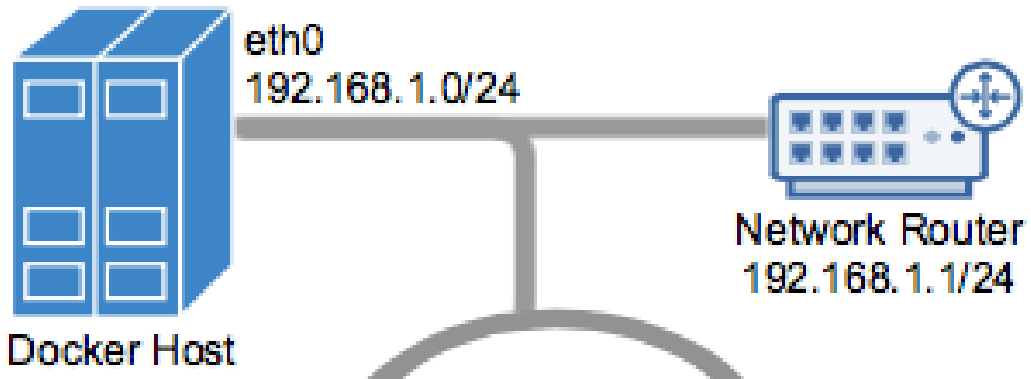
Le pilote réseau IPvlan

- Permet de contrôler l'adressage IP
- Utile quand on veut gérer l'intégration réseau des CT
- Variante de la virtualisation réseau
- Deux avantages d'IPvlan
  1. Meilleure performance en contournant le pont Linux
  2. Moins de variation de l'adressage
- Supprime le pont utilisé par défaut
- Ne nécessite plus de mappage de port

Le pilote réseau IPvlan

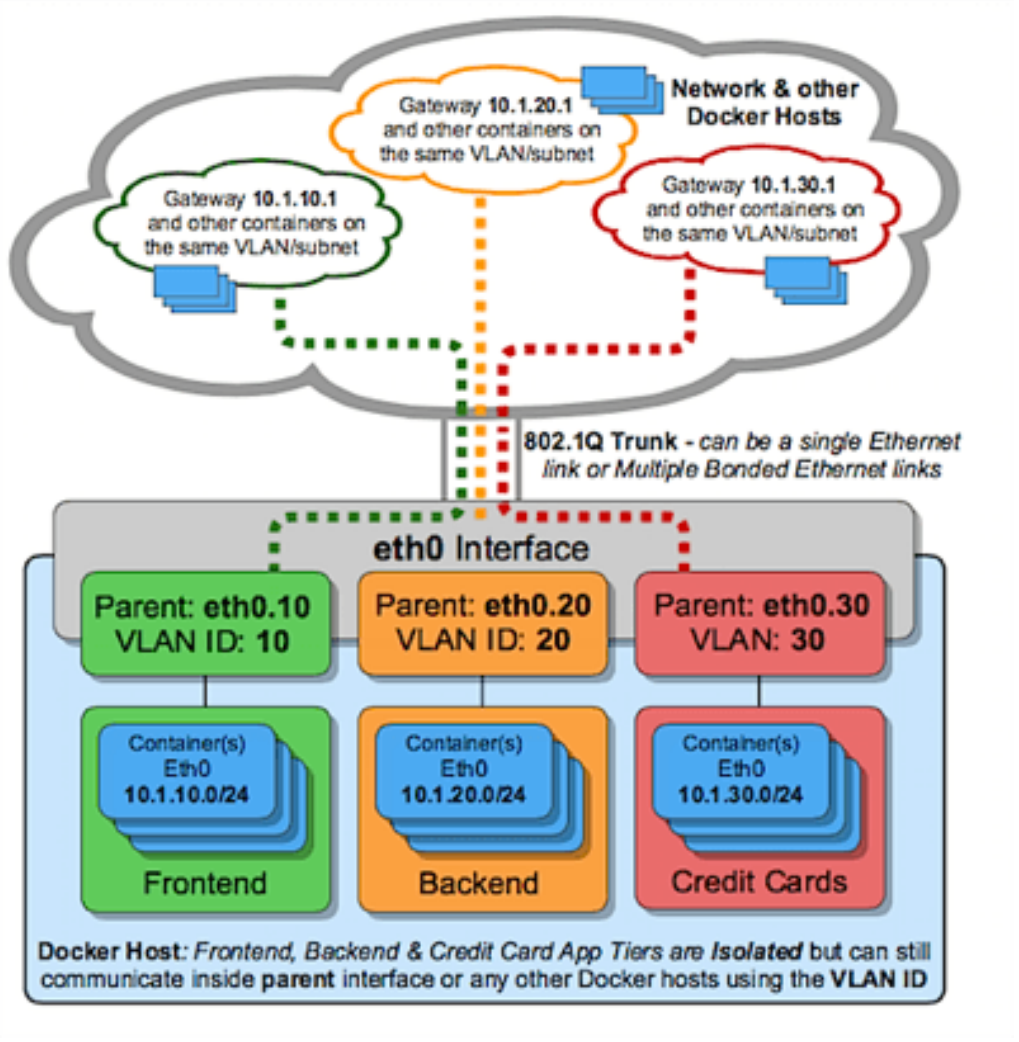
- Différentes options :
  - **ipvlan_mode** : mode d'opération d'IPvlan parmi `l2` (défaut), `l3`, `l3s`.
  - **ipvlan_flag** : indicateur du mode IPvlan parmi `bridge` (défaut), `private`, `vepa`.
  - **parent** : interface parente à utiliser

Exemple en mode L2




```
docker network create -d ipvlan \
  –subnet=192.168.1.0/24 \
  –gateway=192.168.1.1 \
  -o parent=eth0 pub_net
```

Exemple en mode L2 802.1Q



Le pilote réseau None

- Désactive totalement la pile réseau du conteneur
- Ne configure aucune IP
- Ne donne accès ni au réseau externe
- Ni aux autres conteneurs de l'hôte
- Le conteneur est totalement isolé