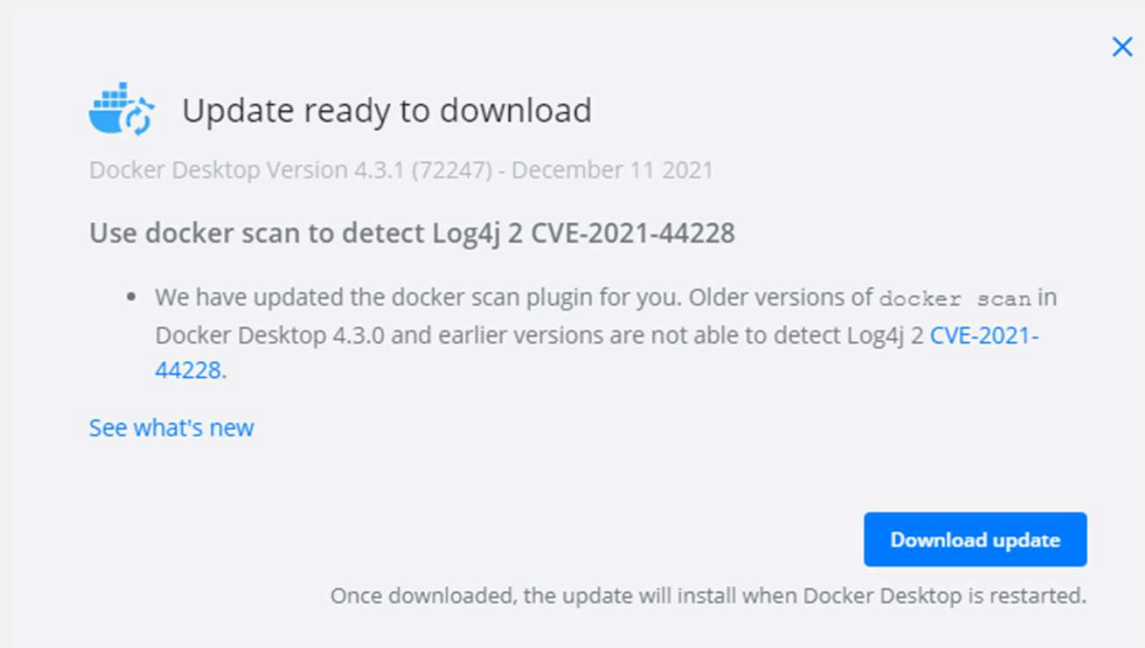




Docker – Stockage et réseau

Objectifs

- Réseau
- Stockage



The screenshot shows a notification window from Docker Desktop. At the top right is a close button (X). Below it is the Docker logo and the text "Update ready to download". Underneath, it says "Docker Desktop Version 4.3.1 (72247) - December 11 2021". The main heading is "Use docker scan to detect Log4j 2 CVE-2021-44228". A bullet point follows: "We have updated the docker scan plugin for you. Older versions of `docker scan` in Docker Desktop 4.3.0 and earlier versions are not able to detect Log4j 2 [CVE-2021-44228](#)." Below this is a link "See what's new". At the bottom right is a blue button labeled "Download update". At the very bottom, a note states: "Once downloaded, the update will install when Docker Desktop is restarted."

Utiliser le scan docker pour détecter Log4j

Nous avons mis à jour le plugin docker scan pour vous. Les anciennes versions de docker scan dans Docker Desktop et les versions antérieures ne sont pas en mesure de détecter log4j.

Docker – Réseau

- Plusieurs drivers :

- Liste : `docker network ls`
- Propriétés : `docker network inspect bridge`
- Autre : create, rm, etc. ([liens vers create](#))

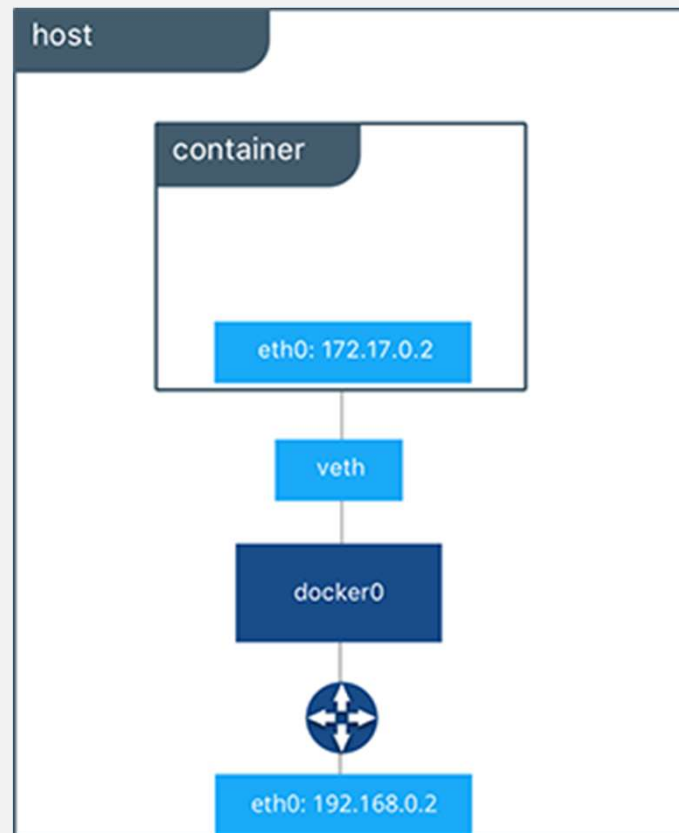
```
> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ac4a6903602e        bridge              bridge              local
81bda30d730b        host                host                local
4393a605bac2        none                null                local
```

- Par défaut docker vient avec 3 drivers :

- bridge : valeur par défaut si non spécifiée, pont – 172.17.0.0/16 (172.17.0.1)
- host : utilise l'IP de l'hôte, **n'est plus isolé de l'hôte**
- none : pas de réseau

<https://docs.docker.com/network/>

Docker – Réseau / conteneur



<https://docs.docker.com/engine/tutorials/networkingcontainers/>

Docker – Réseau / conteneur / redirection

```
docker run -p 8080:5000 --name container mon_image
```

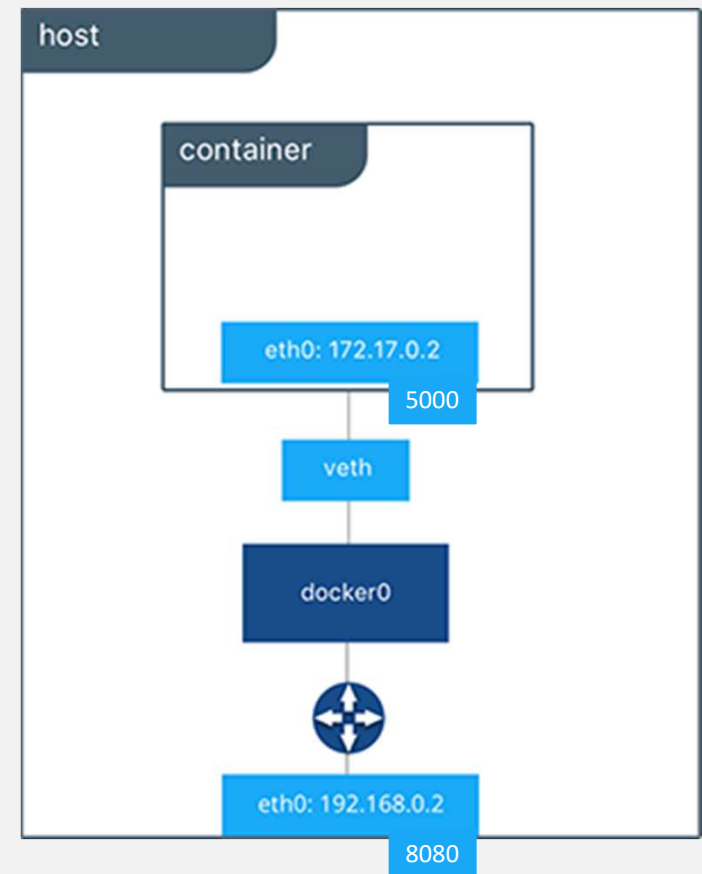
-p Publish a container's port(s) to the host

-rm Automatically remove the container when it exits
-l Keep STDIN open even if not attached
-t Allocate a pseudo-TTY

```
docker run --rm -it -p 8000:80  
mcr.microsoft.com/dotnet/core/samples:aspnetapp
```

[Retour à diapo 3](#)

<https://github.com/dotnet/dotnet-docker/tree/master/samples/aspnetapp>
<https://docs.docker.com/engine/tutorials/networkingcontainers/>



Les conteneurs sont immuables

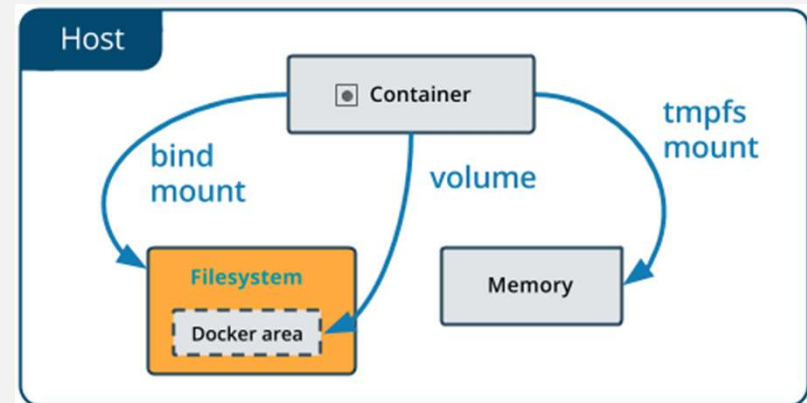
- Les conteneurs sont conçus pour être la plupart du temps **immuable** et **éphémère** (non changeant et temporaire)
- L'infrastructure immuable : on ne fait que redéployer les conteneurs, ceux-ci ne changent jamais.
 - Si des modifications sont nécessaires, on redéploie de nouveaux conteneurs basés sur des images différentes.
- Bénéfices : fiabilité et consistance, tout en permettant la reproduction de tous les changements.

Pour conserver on parle de persistance

- On parle de persistance des données lorsqu'on peut conserver celle-ci même si le conteneur a été supprimé ou recréé.
- Très utile surtout dans les conteneurs avec l'utilisation de bases de données.

Docker – Stockage persistant

- Docker gère trois types de stockage :
 - Point de montage : monter un répertoire dans le conteneur
 - Volume : garder des données indépendamment du conteneur, spécifique à docker
 - Mémoire : tmpfs



<https://docs.docker.com/storage/bind-mounts/>

Docker – Stockage persistant

- Création de volume :
 - `docker volume create mon_volume`
- Lister les volumes :
 - `docker volume ls`
- L'utilisation :
 - `docker run -v mon_volume:/point_montage busybox ls -l /point_montage`
- D'autres commandes :
 - `prune, rm, inspect`

Docker – Stockage persistant

```
> docker volume ls
DRIVER          VOLUME NAME
> docker volume create mon_volume
mon_volume
> docker volume ls
DRIVER          VOLUME NAME
local          mon_volume

> docker run -v mon_volume:/point_montage busybox touch /point_montage/un_fichier
> docker run -v mon_volume:/point_montage busybox ls -l /point_montage
total 0
-rw-r--r--  1 root   root        0 Sep 10 11:12 un_fichier

> docker run -v mon_volume:/point_montage busybox rm /point_montage/un_fichier
> docker run -v mon_volume:/point_montage busybox ls -l /point_montage
total 0
```

Quand utiliser plutôt les volumes ?

- Partager des données **entre plusieurs conteneurs** en cours d'exécution
- Lorsque l'hôte Docker n'est pas paramétré pour avoir un répertoire donnée ou un fichier de structure (permet de découpler la configuration de l'hôte avec le conteneur).
- Lorsqu'on souhaite stocker les données du conteneur sur un **hôte distant** ou sur le **cloud**
- Lorsqu'on souhaite **restaurer**, **sauvegarder** ou **migrer** les données depuis un hôte Docker vers un autre hôte.

Les volumes dans votre Dockerfile

- Le **VOLUME** est un élément optionnel à faire figurer dans le Dockerfile si on veut utiliser cette fonctionnalité.
- **VOLUME** /var/lib/mysql
- Lorsqu'un nouveau conteneur est démarré, un volume va être créé et assigné au répertoire indiqué ci-dessus.

Suppression des volumes

- Les volumes ont besoins d'être supprimés **manuellement**.
- On ne peut pas supprimer les volumes directement en supprimant le conteneur.
- C'est normal, car les données stockées dans ces volumes sont considérées comme étant sensibles et importantes.

Le bind Mounting dans Docker

- Le **BIND MOUNTING** permet d'effectuer une association entre des fichiers ou des répertoires de la machine hôte, avec ceux d'un ou de plusieurs conteneurs.
- Les fichiers et répertoires de l'hôte en Bind Mounting, écrasent ceux qui existent à l'intérieur d'un conteneur.
- Ne peuvent être paramétrés que lors de l'utilisation de la commande **docker container run** et non dans le fichier Dockerfile comme les volumes.

Docker – Stockage persistant

- Utilisation des points de montage (Bind Mounting) :

```
docker run -v /mon_repertoire/a_partager:/point_montage busybox  
ls -l /point_montage
```

```
> mkdir /tmp/mon_repertoire_a_partager  
> touch /tmp/mon_repertoire_a_partager/un_fichier  
> touch /tmp/mon_repertoire_a_partager/un_autre_fichier  
> docker run -v /tmp/mon_repertoire_a_partager:/point_montage busybox ls -l /point_montage
```

```
total 0  
-rw-r--r--    1 root    root          0 Sep 10 10:54 un_autre_fichier  
-rw-r--r--    1 root    root          0 Sep 10 10:54 un_fichier
```

Voici la notation pour un système Windows

```
>docker run -v '//c/Users/jpduches/mysql':/var/lib/mysql
```