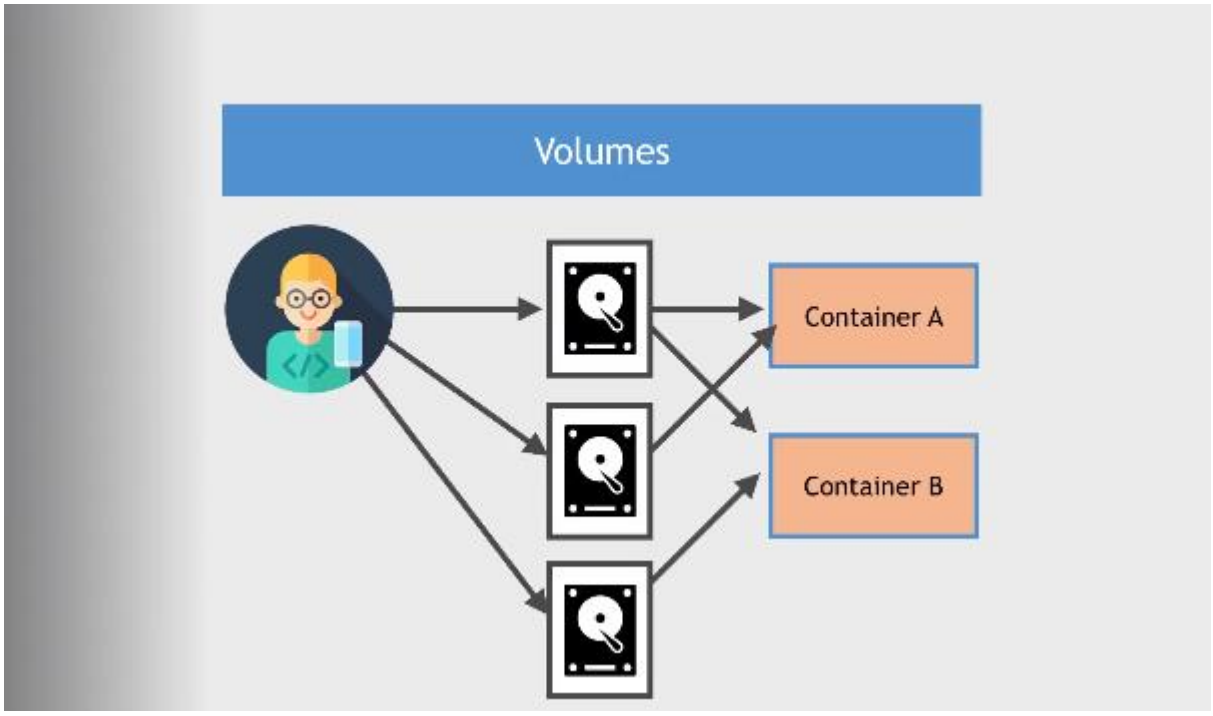
Docker : Volume



docker volume permet de créer des volumes, facilement réutilisables pour plusieurs conteneurs.

La syntaxe reste dans l'esprit de docker :

```
docker volume --help
```

```
vagrant@mydocker:~$ docker volume --help

Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
```

Etape 1 : Création d'un volume simple

Nous allons commencer par créer un volume, voyons ce que **docker volume create** prends comme arguments :

```
docker volume create --help
```

```
vagrant@mydocker:~$ docker volume create --help

Usage:  docker volume create [OPTIONS] [VOLUME]

Create a volume

Options:
  -d, --driver string   Specify volume driver name (default "local")
      --label list      Set metadata for a volume
  -o, --opt map         Set driver specific options (default map[])
```

Donc on va créer notre premier volume :

```
docker volume create --name test
```

```
vagrant@mydocker:~$ docker volume create --name test
test
```

Que nous utiliserons comme ceci :

```
vagrant@mydocker:~$ docker run -ti -v test:/test alpine sh
```

en rouge : dans le conteneur

```
vagrant@mydocker:~$ docker run -ti -v test:/test alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
9d48c3bd43c5: Pull complete
Digest: sha256:72c42ed48c3a2db31b7dafe17d275b634664a708d901ec9fd57b1529280f01fb
Status: Downloaded newer image for alpine:latest
/ #
```

```
/ # ls /test
/ # touch /test/hello.txt
/ # exit
```

```
/ # ls /test
/ # touch /test/hello.txt
/ # exit
vagrant@mydocker:~$
```

```
vagrant@mydocker:~$ docker ps
```

```
vagrant@mydocker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

Si je crée un autre conteneur, on retrouve notre fichier **hello.txt** :

```
vagrant@mydocker:~$ docker run -ti -v test:/test alpine sh
/ # ls /test
hello.txt
/ #
```

On retrouve bien notre fichier, les données sont donc correctement persisté.

```
/ # exit
```

Les fichiers se retrouvent dans **/var/lib/docker/volumes/<volumename>/_data**, sur l'hôte

```
vagrant@mydocker:~$ sudo ls /var/lib/docker/volumes/test/_data
```

```
vagrant@mydocker:~$ sudo ls /var/lib/docker/volumes/test/_data
hello.txt
```

Pour le voir on peut aussi faire

```
vagrant@mydocker:~$ docker volume inspect test
```

```
vagrant@mydocker:~$ docker volume inspect test
[
    {
        "CreatedAt": "2019-09-27T22:03:23Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/test/_data",
        "Name": "test",
        "Options": {},
        "Scope": "local"
    }
]
```

Etape 2 : Suppression d'un volume

```
docker volume ls
```

```
vagrant@mydocker:~$ docker volume ls
DRIVER              VOLUME NAME
local               test
```

```
vagrant@mydocker:~$ docker volume rm test
```

```
vagrant@mydocker:~$ docker volume rm test
Error response from daemon: remove test: volume is in use - [a1a8b44d0e95bda05ccc139fbb21ba307fc26292ce907639089f21f67448a709, 3b7fa54ad3ce55df06f1ccbc4594611fe0559a77129fab312074adb8e51d2293,
 437392945a5ad7fa5437edc758adc7927a20c8b53b53823a57be3558a0c7f927]
```

```
vagrant@mydocker:~$ docker ps
vagrant@mydocker:~$ docker ps -a
```

```
vagrant@mydocker:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
437392945a5a        alpine              "sh"                29 minutes ago      Exited (0) 28 minutes ago                       elastic_galois
3b7fa54ad3ce        alpine              "sh"                29 minutes ago      Exited (130) 29 minutes ago                     jovial_mclean
a1a8b44d0e95        alpine              "sh"                36 minutes ago      Exited (0) 33 minutes ago                       thirsty_lovelace
c5fcd5027ed5        mycentos:1.2        "bash"              26 hours ago        Exited (1) 26 hours ago                         priceless_easley
4c3c0120e7dc        centos              "/bin/bash"         27 hours ago        Exited (1) 27 hours ago                         mycentos
e52be3ae31ed        centos              "/bin/bash"         29 hours ago        Exited (1) 29 hours ago                         mycentos1
```

Supprimer tous les conteneurs

```
vagrant@mydocker:~$ docker rm $(docker ps -a -q)
```

```
vagrant@mydocker:~$ docker rm $(docker ps -a -q)
437392945a5a
3b7fa54ad3ce
a1a8b44d0e95
c5fcd5027ed5
4c3c0120e7dc
e52be3ae31ed
```

```
vagrant@mydocker:~$ docker ps -a
```

```
vagrant@mydocker:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

```
vagrant@mydocker:~$ docker volume rm test
```

```
vagrant@mydocker:~$ docker volume rm test
test
```

```
vagrant@mydocker:~$ docker volume ls
```

```
vagrant@mydocker:~$ docker volume ls
DRIVER              VOLUME NAME
```

Etape 5 : Partager les volumes

Les volumes peuvent être montés dans plusieurs conteneurs.

Ce qui permet donc à deux conteneurs et donc à deux applications d'accéder aux mêmes données.

```
Cd /home/vagrant
mkdir datas
cd datas
touch message1
echo 'Hello World' >> message1
cat message1
cd ..
```

Créez un volume nommé pour pouvoir en fait le partager

```
docker run -d --name=host1 --hostname=host1 -it -v /home/vagrant/datas:/datas centos bash
docker run -d --name=host2 --hostname=host2 -it -v /home/vagrant/datas:/datas centos bash
docker attach host1
cat /datas/message1
echo "bonne nuit" >> /datas/message2
cat /datas/message2
ctrl + p + q
docker attach host2
cat /datas/message1
cat /datas/message2
```

Arrêtez et supprimez tous les conteneurs

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

```
vagrant@mydocker:~/datas$ docker container stop $(docker container ls -aq)
119f9237605e
b3f6e5a9aa51
```

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

```
vagrant@mydocker:~/datas$ docker container rm $(docker container ls -aq)
119f9237605e
b3f6e5a9aa51
```

Etape 6 : Définir les volumes dans un Dokerfile

```
Cd /home/vagrant
mkdir myhello
cd myhello
touch Dockerfile
nano Dockerfile
```

Ajouter le contenu dans le fichier ouvert

```
FROM centos
RUN mkdir /myvol
RUN echo "Hello World" > /myvol/message
VOLUME /myvol
```

Enregistrer (ctrl + s / ctrl + x)

Construire l'image

```
docker build -t myhello:1.0 .
```

```
vagrant@mydocker:~/datas/myhello$ docker build -t myhello:1.0 .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM centos
 ---> 67fa590cfc1c
Step 2/4 : RUN mkdir /myvol
 ---> Running in 37d95a3797c3
Removing intermediate container 37d95a3797c3
 ---> 91465c6a83a1
Step 3/4 : RUN echo "Hello World" > /myvol/message
 ---> Running in 2e25d5619dca
Removing intermediate container 2e25d5619dca
 ---> 274e12820000
Step 4/4 : VOLUME /myvoly
 ---> Running in cb53f973e7f8
Removing intermediate container cb53f973e7f8
 ---> 1af6b65f6da4
Successfully built 1af6b65f6da4
Successfully tagged myhello:1.0
vagrant@mydocker:~/datas/myhello$
```

On va lancer un conteneur à partir de la nouvelle image qu'on vient de créer.

```
docker run -it --rm --name hello1 myhello:1.0
df -h
cat /myvol/message
exit
```