Version : **2022.01**

Dernière mise-à-jour : 2021/12/29 10:32

DOF104 - Gestion des Volumes, du Réseau et de la Supervision des Conteneurs

Contenu du Module

LAB #1 - Gestion des Volumes

Lancez un conteneur à partir de la dernière image :

```
root@debian9:~/mongodb# docker run -d --name mongo2 i2tch/mongodb2
e91a055283f4d67cbd91d11bb3faa6f67925893cb18f9cc25023e72e0f7ed85a
```

1.1 - Gestion Automatique de Volumes par Docker

Vérifiez que le processus est bien démarré dans le conteneur :

```
root@debian9:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
b9773e4aa06d        i2tch/mongodb2      "docker-entrypoint..."   7 hours ago         Up About a minute
27017/tcp           mongo2
```

Identifiez ensuite le point de montage du répertoire **/data/db** du conteneur :

```
root@debian9:~# docker inspect mongo2
...
        "Mounts": [
            {
                "Type": "volume",
                "Name": "9c84c2d1a0db811a3c13dce354ba21169f3073513c8d025dd96c5f902364d44a",
                "Source":
"/var/lib/docker/volumes/9c84c2d1a0db811a3c13dce354ba21169f3073513c8d025dd96c5f902364d44a/_data",
                "Destination": "/data/configdb",
                "Driver": "local",
                "Mode": "",
                "RW": true,
```

```
                "Propagation": ""
            },
            {
                "Type": "volume",
                "Name": "a6177cf4b46089356280f084dd2e272f673aa4a81accb53f031267fafcee6050",
                "Source":
"/var/lib/docker/volumes/a6177cf4b46089356280f084dd2e272f673aa4a81accb53f031267fafcee6050/_data",
                "Destination": "/data/db",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            }
...
            "Volumes": {
                "/data/configdb": {},
                "/data/db": {}
            },
...
```

En regardant le contenu du répertoire **/data/db**, on constate une arborescence classique de stockage de données de mongodb :

```
root@debian9:~# ls /var/lib/docker/volumes/a6177cf4b46089356280f084dd2e272f673aa4a81accb53f031267fafcee6050/_data
journal  local.0  local.ns  mongod.lock  storage.bson
```

Arrêtez et supprimez le conteneur **mongo2** :

```
root@debian9:~# docker stop mongo2
mongo2
root@debian9:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
ea239635e141        testcache           "more /tmp/moment"       6 hours ago         Exited (0) 6 hours ago
test1
```

```
21b0490a93dd        i2tch/mydocker      "/entrypoint.sh my..."   6 hours ago         Exited (137) 6 hours ago
myDocker
b9773e4aa06d        i2tch/mongodb2      "docker-entrypoint..."   7 hours ago         Exited (0) 10 seconds ago
mongo2
bdb4bc0f81de        i2tch/mongodb1      "docker-entrypoint..."   18 hours ago        Created
27017/tcp           mongo1
f5b45072b831        i2tch/mongodb       "bash"                   18 hours ago        Exited (137) 6 hours ago
mongo
9731a48f126a        nginx               "nginx -g 'daemon ..."   18 hours ago        Exited (0) 6 hours ago
cocky_gates
eacd70596e23        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 19 hours ago
adoring_yonath
cffb4456e9c4        ubuntu              "/bin/bash"              19 hours ago        Exited (0) 19 hours ago
i2tch
root@debian9:~# docker rm mongo2
mongo2
root@debian9:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
ea239635e141        testcache           "more /tmp/moment"       6 hours ago         Exited (0) 6 hours ago
test1
21b0490a93dd        i2tch/mydocker      "/entrypoint.sh my..."   6 hours ago         Exited (137) 6 hours ago
myDocker
bdb4bc0f81de        i2tch/mongodb1      "docker-entrypoint..."   18 hours ago        Created
27017/tcp           mongo1
f5b45072b831        i2tch/mongodb       "bash"                   18 hours ago        Exited (137) 6 hours ago
mongo
9731a48f126a        nginx               "nginx -g 'daemon ..."   18 hours ago        Exited (0) 6 hours ago
cocky_gates
eacd70596e23        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 19 hours ago
adoring_yonath
cffb4456e9c4        ubuntu              "/bin/bash"              19 hours ago        Exited (0) 19 hours ago
i2tch
```

Re-créez maintenant un conteneur à partir de l'image **i2tch/mongodb2** :

```
root@debian9:~# docker run -d --name mongo2 i2tch/mongodb2
a8382642c4e849337e12a60419b10f63ea21251dfcc2c6050284ca3eed7fa13d
root@debian9:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
a8382642c4e8        i2tch/mongodb2      "docker-entrypoint..."   12 seconds ago      Exited (100) 11 seconds ago
mongo2
ea239635e141        testcache           "more /tmp/moment"       6 hours ago         Exited (0) 6 hours ago
test1
21b0490a93dd        i2tch/mydocker      "/entrypoint.sh my..."   6 hours ago         Exited (137) 6 hours ago
myDocker
bdb4bc0f81de        i2tch/mongodb1      "docker-entrypoint..."   18 hours ago        Created
27017/tcp           mongo1
f5b45072b831        i2tch/mongodb       "bash"                   18 hours ago        Exited (137) 6 hours ago
mongo
9731a48f126a        nginx               "nginx -g 'daemon ..."   18 hours ago        Exited (0) 6 hours ago
cocky_gates
eacd70596e23        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 19 hours ago
adoring_yonath
cffb4456e9c4        ubuntu              "/bin/bash"              19 hours ago        Exited (0) 19 hours ago
i2tch
```

Utilisez de nouveau la commande **docker inspect** pour identifier le point de montage du répertoire **/data/db** :

```
root@debian9:~# docker inspect mongo2
...
        "Mounts": [
            {
                "Type": "volume",
                "Name": "76dcc0ccbe6604278cf8e8da0398a807f5d0719087f17c227c8504be24456d43",
                "Source":
"/var/lib/docker/volumes/76dcc0ccbe6604278cf8e8da0398a807f5d0719087f17c227c8504be24456d43/_data",
```

```
                "Destination": "/data/db",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            },
            {
                "Type": "volume",
                "Name": "3bf724ceb38ce0792469d7e403f05b6794f27e0aa72bda51a8ab75b2df5ae87c",
                "Source":
"/var/lib/docker/volumes/3bf724ceb38ce0792469d7e403f05b6794f27e0aa72bda51a8ab75b2df5ae87c/_data",
                "Destination": "/data/configdb",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            }
...
```

Important : Notez que le répertoire des données du précédent conteneur, **/var/lib/docker/volumes/a6177cf4b46089356280f084dd2e272f673aa4a81accb53f031267fafcee6050/_data** n'est pas le même que le conteneur courant **/var/lib/docker/volumes/76dcc0ccbe6604278cf8e8da0398a807f5d0719087f17c227c8504be24456d43/_data**.

Les conteneurs n'ayant pas été arrêtés avec l'option **-v**, on peut constater que les répertoires persistent dans **/var/lib/docker** :

```
root@debian9:~# ls -l /var/lib/docker/volumes/
total 52
drwxr-xr-x 3 root root  4096 Sep  7 09:43 3bf724ceb38ce0792469d7e403f05b6794f27e0aa72bda51a8ab75b2df5ae87c
drwxr-xr-x 3 root root  4096 Sep  6 16:07 46d11d005d05757609ff76159ce0992d210089c5247fa54b024706a20b0de501
drwxr-xr-x 3 root root  4096 Sep  7 09:43 76dcc0ccbe6604278cf8e8da0398a807f5d0719087f17c227c8504be24456d43
```

```
drwxr-xr-x 3 root root  4096 Sep  7 02:33 9c84c2d1a0db811a3c13dce354ba21169f3073513c8d025dd96c5f902364d44a
drwxr-xr-x 3 root root  4096 Sep  7 02:33 a6177cf4b46089356280f084dd2e272f673aa4a81accb53f031267fafcee6050
drwxr-xr-x 3 root root  4096 Sep  6 16:07 cc38fa97138adc55976aa16993d8920c5f7da922ad1b2a07936d30cc82d59f38
-rw------- 1 root root 32768 Sep  7 09:43 metadata.db
```

Important : Notez que non-seulement ceci représente une source de perte d'espace disque mais prouve aussi que les données ne sont pas persistantes entre deux instances d'un conteneur d'**i2tch/mongodb2**. Ceci crée bien évidement un problème important en production.

1.2 - Gestion Manuelle d'un Volume

Arrêtez et supprimez le conteneur **mongo2** puis re-créez un conteneur avec un volume spécifique pour contenir les données placées dans **/data/db** du conteneur par mongodb :

```
root@debian9:~# docker stop mongo2
mongo2
root@debian9:~# docker rm mongo2
mongo2
root@debian9:~# docker run -d --name mongo2 -v persistent_data:/data/db i2tch/mongodb2
3cf093d72b9e3739f2cb288e571244e494b7518292c31994ee012e3620bb0e98
root@debian9:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
3cf093d72b9e        i2tch/mongodb2      "docker-entrypoint..."   21 seconds ago      Up 20 seconds
27017/tcp           mongo2
ea239635e141        testcache           "more /tmp/moment"       6 hours ago         Exited (0) 6 hours ago
test1
21b0490a93dd        i2tch/mydocker      "/entrypoint.sh my..."   6 hours ago         Exited (137) 6 hours ago
myDocker
bdb4bc0f81de        i2tch/mongodb1      "docker-entrypoint..."   18 hours ago        Created
```

```
27017/tcp           mongo1
f5b45072b831        i2tch/mongodb       "bash"                   18 hours ago        Exited (137) 6 hours ago
mongo
9731a48f126a        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 6 hours ago
cocky_gates
eacd70596e23        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 19 hours ago
adoring_yonath
cffb4456e9c4        ubuntu              "/bin/bash"              19 hours ago        Exited (0) 19 hours ago
i2tch
root@debian9:~# docker logs mongo2
2017-09-07T08:53:12.523+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-
bit host=3cf093d72b9e
2017-09-07T08:53:12.524+0000 I CONTROL  [initandlisten] db version v3.0.15
2017-09-07T08:53:12.524+0000 I CONTROL  [initandlisten] git version: b8ff507269c382bc100fc52f75f48d54cd42ec3b
2017-09-07T08:53:12.524+0000 I CONTROL  [initandlisten] build info: Linux ip-10-166-66-3 3.2.0-4-amd64 #1 SMP
Debian 3.2.46-1 x86_64 BOOST_LIB_VERSION=1_49
2017-09-07T08:53:12.524+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-09-07T08:53:12.524+0000 I CONTROL  [initandlisten] options: { storage: { mmapv1: { smallFiles: true } } }
2017-09-07T08:53:12.535+0000 I JOURNAL  [initandlisten] journal dir=/data/db/journal
2017-09-07T08:53:12.535+0000 I JOURNAL  [initandlisten] recover : no journal files present, no recovery needed
2017-09-07T08:53:13.368+0000 I JOURNAL  [initandlisten] preallocateIsFaster=true 15.4
2017-09-07T08:53:14.410+0000 I JOURNAL  [initandlisten] preallocateIsFaster=true 19.36
2017-09-07T08:53:16.277+0000 I JOURNAL  [initandlisten] preallocateIsFaster=true 15.86
2017-09-07T08:53:16.277+0000 I JOURNAL  [initandlisten] preallocateIsFaster check took 3.742 secs
2017-09-07T08:53:16.277+0000 I JOURNAL  [initandlisten] preallocating a journal file /data/db/journal/prealloc.0
2017-09-07T08:53:19.930+0000 I JOURNAL  [initandlisten] preallocating a journal file /data/db/journal/prealloc.1
2017-09-07T08:53:23.035+0000 I JOURNAL  [initandlisten] preallocating a journal file /data/db/journal/prealloc.2
2017-09-07T08:53:25.889+0000 I JOURNAL  [durability] Durability thread started
2017-09-07T08:53:25.889+0000 I JOURNAL  [journal writer] Journal writer thread started
2017-09-07T08:53:26.016+0000 I INDEX    [initandlisten] allocating new ns file /data/db/local.ns, filling with
zeroes...
2017-09-07T08:53:26.246+0000 I STORAGE  [FileAllocator] allocating new datafile /data/db/local.0, filling with
zeroes...
2017-09-07T08:53:26.246+0000 I STORAGE  [FileAllocator] creating directory /data/db/_tmp
```

```
2017-09-07T08:53:26.256+0000 I STORAGE  [FileAllocator] done allocating datafile /data/db/local.0, size: 16MB,
took 0.002 secs
2017-09-07T08:53:26.299+0000 I NETWORK  [initandlisten] waiting for connections on port 27017
```

Notez que cette fois-ci, docker a créé un répertoire **persistent_data** dans le répertoire **/var/lib/docker/volumes/** :

```
root@debian9:~# ls -l /var/lib/docker/volumes/
total 68
drwxr-xr-x 3 root root  4096 Sep  7 09:43 3bf724ceb38ce0792469d7e403f05b6794f27e0aa72bda51a8ab75b2df5ae87c
drwxr-xr-x 3 root root  4096 Sep  6 16:07 46d11d005d05757609ff76159ce0992d210089c5247fa54b024706a20b0de501
drwxr-xr-x 3 root root  4096 Sep  7 09:46 511e23f818d5cf60f4333a3fe8fd2e4333c900dec6eee97f70448bfb0091184d
drwxr-xr-x 3 root root  4096 Sep  7 09:53 5ca72be4140ecf1271efe7342cf7cd58ce66fc3673d12c04b8503603b8cee66c
drwxr-xr-x 3 root root  4096 Sep  7 09:43 76dcc0ccbe6604278cf8e8da0398a807f5d0719087f17c227c8504be24456d43
drwxr-xr-x 3 root root  4096 Sep  7 02:33 9c84c2d1a0db811a3c13dce354ba21169f3073513c8d025dd96c5f902364d44a
drwxr-xr-x 3 root root  4096 Sep  7 02:33 a6177cf4b46089356280f084dd2e272f673aa4a81accb53f031267fafcee6050
drwxr-xr-x 3 root root  4096 Sep  6 16:07 cc38fa97138adc55976aa16993d8920c5f7da922ad1b2a07936d30cc82d59f38
-rw------- 1 root root 65536 Sep  7 09:53 metadata.db
drwxr-xr-x 3 root root  4096 Sep  7 09:46 persistent_data
```

Arrêtez et supprimez le conteneur **mongo2** puis re-créez un conteneur en utilisant le même volume spécifique pour contenir les données placées dans **/data/db** du conteneur par mongodb :

```
root@debian9:~# docker stop mongo2
mongo2
root@debian9:~# docker rm mongo2
mongo2
root@debian9:~# docker run -d --name mongo2 -v persistent_data:/data/db i2tch/mongodb2
ad672c3038245c25a36162d05820c21f7250557ac342582d0908d3ca33799e37
root@debian9:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
ad672c303824        i2tch/mongodb2      "docker-entrypoint..."   24 seconds ago      Up 22 seconds
27017/tcp           mongo2
ea239635e141        testcache           "more /tmp/moment"       6 hours ago         Exited (0) 6 hours ago
```

```
test1
21b0490a93dd        i2tch/mydocker      "/entrypoint.sh my..."   6 hours ago         Exited (137) 6 hours ago
myDocker
bdb4bc0f81de        i2tch/mongodb1      "docker-entrypoint..."   18 hours ago        Created
27017/tcp           mongo1
f5b45072b831        i2tch/mongodb       "bash"                   18 hours ago        Exited (137) 6 hours ago
mongo
9731a48f126a        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 6 hours ago
cocky_gates
eacd70596e23        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 19 hours ago
adoring_yonath
cffb4456e9c4        ubuntu              "/bin/bash"              19 hours ago        Exited (0) 19 hours ago
i2tch
```

Encore une fois, cherchez le point de montage de **/data/db** grâce à l'utilisation de la commande **docker inspect** :

```
root@debian9:~# docker inspect mongo2
...
        "Mounts": [
            {
                "Type": "volume",
                "Name": "6cefc73cef475279dfe20e25421fa358e6aa995b5c175b9f2c7a9b86163661e5",
                "Source":
"/var/lib/docker/volumes/6cefc73cef475279dfe20e25421fa358e6aa995b5c175b9f2c7a9b86163661e5/_data",
                "Destination": "/data/configdb",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
            },
            {
                "Type": "volume",
                "Name": "persistent_data",
                "Source": "/var/lib/docker/volumes/persistent_data/_data",
```

```
                "Destination": "/data/db",
                "Driver": "local",
                "Mode": "z",
                "RW": true,
                "Propagation": ""
            }
...
```

Important : Notez ici que l'utilisation du même répertoire entre les deux conteneurs rend les données persistantes et évite la création de volumes orphelins. Pour plus d'information sur les volumes, consultez la page : **https://docs.docker.com/storage/volumes/**.

Pour créer un volume à utiliser avec un conteneur utilisez la commande docker volume **create** :

```
root@debian9:~# docker volume create myvolume
myvolume
```

Pour lister les volumes, utilisez la commande docker volume **ls** :

```
root@debian9:~# docker volume ls
DRIVER              VOLUME NAME
local               myvolume
```

Notez maintenant l'emplacement physique du volume créé :

```
root@debian9:~# docker volume inspect myvolume
[
    {
        "CreatedAt": "2021-04-15T09:35:21+02:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",
```

```
        "Name": "myvolume",
        "Options": {},
        "Scope": "local"
    }
]
```

Créez un fichier témoin dans le répertoire **/var/lib/docker/volumes/myvolume/_data/** :

```
root@debian9:~# touch /var/lib/docker/volumes/myvolume/_data/test-file
```

Démarrez maintenant un conteneur qui utilise ce volume :

```
root@debian9:~# docker run -it --name ubuntu-volume --mount source=myvolume,target=/myvolume ubuntu bash
root@673f9c8bc837:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  myvolume  opt  proc  root  run  sbin  srv  sys
tmp  usr  var
```

Important : Notez l'utilisation de l'option **–mount** au lieu de l'option **–volume** ou **-v**. Introduit en Docker version 17.06, Docker recommende l'utilisation de l'option –mount plutôt que l'option -v..

Notez que le fichier témoin **test-file** est disponible dans le conteneur :

```
root@673f9c8bc837:/# cd myvolume/
root@673f9c8bc837:/myvolume# ls
test-file
```

Créez un deuxième fichier témoin dans le répertoire **/myvolume** du conteneur et quittez celui-ci :

```
root@673f9c8bc837:/myvolume# touch container_volume
root@673f9c8bc837:/myvolume# exit
```

Contrôlez maintenant le contenu du répertoire **/var/lib/docker/volumes/myvolume/_data/** :

```
root@debian9:~# ls -l /var/lib/docker/volumes/myvolume/_data/
total 0
-rw-r--r-- 1 root root 0 avril 15 10:22 container_volume
-rw-r--r-- 1 root root 0 avril 15 10:16 test-file
```

Important : Notez que les deux fichiers témoins sont visibles.

```
root@debian9:~# docker rm ubuntu-volume
ubuntu-volume
root@debian9:~# ls -l /var/lib/docker/volumes/myvolume/_data/
total 0
-rw-r--r-- 1 root root 0 avril 15 10:22 container_volume
-rw-r--r-- 1 root root 0 avril 15 10:16 test-file
```

Important : Notez que les deux fichiers témoins sont toujours visibles.

Créez maintenant un deuxième conteneur en spécifiant un volume qui n'existe pas :

```
root@debian9:~# docker run -it --rm --name ubuntu-volume --mount source=myvolume1,target=/myvolume1 ubuntu bash
root@b1476960de63:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  myvolume1  opt  proc  root  run  sbin  srv  sys
tmp  usr  var
root@b1476960de63:/# cd myvolume1
root@b1476960de63:/myvolume1# touch file_myvolume1
root@b1476960de63:/myvolume1# exit
exit
```

Notez que Docker a automatiquement créé le volume :

```
root@debian9:~# docker volume ls
DRIVER              VOLUME NAME
local               myvolume
local               myvolume1
root@debian9:~# ls -l /var/lib/docker/volumes/myvolume1/_data/
total 0
-rw-r--r-- 1 root root 0 avril 15 12:06 file_myvolume1
```

Un autre type de volume utilisable avec Docker est le **Bindmount**. Un Bindmount :

- dépend de la structure de l'arborescence de l'hôte Docker,
- ne peut pas être contrôler par la CLI Docker.

Pour créer un Bindmount, commencez par créer le répertoire **bindmount** dans **/root** :

```
root@debian9:~# mkdir bindmount
root@debian9:~# touch bindmount/test_bind
```

Montez le Bindmount à l'intérieur d'un conteneur :

```
root@debian9:~# docker run -it --name ubuntu-volume --mount type=bind,source=/root/bindmount,target=/bindmount
ubuntu bash
root@7b13fe558984:/# ls
bin  bindmount  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys
tmp  usr  var
root@7b13fe558984:/# cd bindmount
root@7b13fe558984:/bindmount# ls
test_bind
root@7b13fe558984:/bindmount# touch container_bind
root@7b13fe558984:/bindmount# ls
container_bind  test_bind
root@7b13fe558984:/bindmount# exit
```

```
exit
```

Conrôlez la présence du fichier témoin :

```
root@debian9:~# ls -l bindmount/
total 0
-rw-r--r-- 1 root root 0 avril 15 10:32 container_bind
-rw-r--r-- 1 root root 0 avril 15 10:32 test_bind
root@debian9:~# docker rm ubuntu-volume
ubuntu-volume
root@debian9:~# ls -l bindmount/
total 0
-rw-r--r-- 1 root root 0 avril 15 10:32 container_bind
-rw-r--r-- 1 root root 0 avril 15 10:32 test_bind
```

Notez que la CLI Docker n'a pas de connaissance de ce point de montage :

```
root@debian9:~# docker volume ls
DRIVER              VOLUME NAME
local               myvolume
```

LAB #2 - Gestion du Réseau

2.1 - L'Approche Réseau Docker

Docker fournit trois réseaux par défaut :

```
root@debian9:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
495b3db75b0d        bridge              bridge              local
e1ed4de2f947        host                host                local
```

```
6bda460c97c6        none                null                local
```

Bridge

Ce type de réseau est limité aux conteneurs d'un hôte unique exécutant Docker. Les conteneurs ne peuvent communiquer qu'entre eux et ils ne sont pas accessibles depuis l'extérieur. Pour que les conteneurs sur le réseau puissent communiquer ou être accessibles du monde extérieur, il faut configurer le mappage de port.

Par défaut Docker fonctionne en mode **Pont** ou (*Bridge*) et crée une interface intermédiaire à cet effet appelé **docker0** :

```
root@debian9:~# ip addr show docker0
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:38:f1:e7:ee 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
```

Démarrez un conteneur dénommé **resotest** à partir d'une image de CentOS :

```
root@debian9:~# docker run -itd --name=resotest centos
2169360fcbfdbd6e68ea969a95edeb6fc42603c23ee42f03ceec286276519855
```

Lancez ensuite la commande **docker network inspect bridge** à partir de la machine virtuelle hôte de Debian_9 :

```
root@debian9:~# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "495b3db75b0d4bfcfc6da7c3e2af5f6addcdc227aa8b69b1e59a998be1819d12",
        "Created": "2017-09-07T07:44:49.942615596+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
```

```
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "2169360fcbfdbd6e68ea969a95edeb6fc42603c23ee42f03ceec286276519855": {
                "Name": "resotest",
                "EndpointID": "fc74e519d69b9a2112be959c92cda22b67671b52efbbd36fadf66097ccbb1271",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
            },
            "ad672c3038245c25a36162d05820c21f7250557ac342582d0908d3ca33799e37": {
                "Name": "mongo2",
                "EndpointID": "adc15132fb73b57ab14e960feeff1b965321ada411be8535b715b103b941d8cc",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
```

```
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Important : Notez ici que les conteneurs **mongo2** et **resotest** ne disposent pas de la même adresse que l'interface **docker0** de la machine hôte. Cependant les adresses se trouvent dans le même segment - **172.17.0.0/16** indiqué par la sortie **“Subnet”: “172.17.0.0/16”**.

Vous pouvez déconnecter un conteneur du réseau en utilisant la commande suivante :

```
root@debian9:~# docker network disconnect bridge resotest
root@debian9:~# docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "495b3db75b0d4bfcfc6da7c3e2af5f6addcdc227aa8b69b1e59a998be1819d12",
        "Created": "2017-09-07T07:44:49.942615596+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
```

```
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "ad672c3038245c25a36162d05820c21f7250557ac342582d0908d3ca33799e37": {
                "Name": "mongo2",
                "EndpointID": "adc15132fb73b57ab14e960feeff1b965321ada411be8535b715b103b941d8cc",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Créez maintenant votre propre réseau ponté appelé **my-bridged-network** :

```
root@debian9:~# docker network create -d bridge --subnet 172.25.0.0/16 --gateway 172.25.0.1 my-bridged-network
ceb7ba7493933c55d181bc92b1f799ca07bfe84b168d52a6ac648c1a906093f3
root@debian9:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
495b3db75b0d        bridge              bridge              local
e1ed4de2f947        host                host                local
ceb7ba749393        my-bridged-network   bridge              local
6bda460c97c6        none                null                local
```

Bien évidement, ce réseau est actuellement vide :

```
root@debian9:~# docker network inspect my-bridged-network
[
    {
        "Name": "my-bridged-network",
        "Id": "ceb7ba7493933c55d181bc92b1f799ca07bfe84b168d52a6ac648c1a906093f3",
        "Created": "2017-09-07T10:03:17.063730665+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.25.0.0/16",
                    "Gateway": "172.25.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
```

```
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Lancez maintenant deux conteneurs et consultez les informations concernant le réseau :

```
root@debian9:~# docker run -itd --name=centos1 centos
9f36a628c72b383edfd4dc13ee4e4b2eaf5be0078d780f0334fcb8be0d977d0e

root@debian9:~# docker run -itd --name=centos2 centos
aaed3bc8e404ee1bccd6c87b39de32332940b5391514691fc70188edb17c1d7c

root@debian9:~# docker inspect --format='{{json .NetworkSettings.Networks}}'  centos1
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"495b3db75b0d4bfcfc6da7c3e2af5f6addcdc227aa8
b69b1e59a998be1819d12","EndpointID":"d7b87875688b45258fc867b6bb8b0a0592f5c5fa16857fe136e55b87b6698219","Gateway":
"172.17.0.1","IPAddress":"172.17.0.3","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixL
en":0,"MacAddress":"02:42:ac:11:00:03","DriverOpts":null}}

root@debian9:~# docker inspect --format='{{json .NetworkSettings.Networks}}'  centos2
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"495b3db75b0d4bfcfc6da7c3e2af5f6addcdc227aa8
b69b1e59a998be1819d12","EndpointID":"2bfe090dccef89495d437d8deba5765996a917544ab7fde28ef5199f4e907eb1","Gateway":
"172.17.0.1","IPAddress":"172.17.0.4","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixL
en":0,"MacAddress":"02:42:ac:11:00:04","DriverOpts":null}}

root@debian9:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos1
172.17.0.3
```

```
root@debian9:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos2
172.17.0.4
```

Mettez le conteneur **centos1** dans le réseau **my-bridged-network** :

```
root@debian9:~# docker network connect my-bridged-network centos1

root@debian9:~# docker network inspect my-bridged-network
[
    {
        "Name": "my-bridged-network",
        "Id": "ceb7ba7493933c55d181bc92b1f799ca07bfe84b168d52a6ac648c1a906093f3",
        "Created": "2017-09-07T10:03:17.063730665+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.25.0.0/16",
                    "Gateway": "172.25.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
```

```
            "9f36a628c72b383edfd4dc13ee4e4b2eaf5be0078d780f0334fcb8be0d977d0e": {
                "Name": "centos1",
                "EndpointID": "71e10e4e34ce8c42ef029e302f6ed372357f6fde8fd87fc2cbc1b14c2bdf6bb5",
                "MacAddress": "02:42:ac:19:00:02",
                "IPv4Address": "172.25.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]

root@debian9:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos1
172.17.0.3172.25.0.2
```

Important : Notez que le conteneur **centos1** se trouve dans deux réseaux.

Faites la même chose pour le conteneur **centos2** :

```
root@debian9:~# docker network connect my-bridged-network centos2

root@debian9:~# docker network inspect my-bridged-network
[
    {
        "Name": "my-bridged-network",
        "Id": "ceb7ba7493933c55d181bc92b1f799ca07bfe84b168d52a6ac648c1a906093f3",
        "Created": "2017-09-07T10:03:17.063730665+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
```

```
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.25.0.0/16",
                    "Gateway": "172.25.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "9f36a628c72b383edfd4dc13ee4e4b2eaf5be0078d780f0334fcb8be0d977d0e": {
                "Name": "centos1",
                "EndpointID": "71e10e4e34ce8c42ef029e302f6ed372357f6fde8fd87fc2cbc1b14c2bdf6bb5",
                "MacAddress": "02:42:ac:19:00:02",
                "IPv4Address": "172.25.0.2/16",
                "IPv6Address": ""
            },
            "aaed3bc8e404ee1bccd6c87b39de32332940b5391514691fc70188edb17c1d7c": {
                "Name": "centos2",
                "EndpointID": "34f533622f134b995097f1d3e6ce935158c1e5644201f896b42336738a81819c",
                "MacAddress": "02:42:ac:19:00:03",
                "IPv4Address": "172.25.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
```

```
        "Labels": {}
    }
]

root@debian9:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos2
172.17.0.4172.25.0.3
```

Connectez-vous au conteneur **centos1** en lançant bash :

```
root@debian9:~# docker exec -it centos1 bash
```

Vérifiez que la connectivité fonctionne :

```
[root@9f36a628c72b /]# ping 172.25.0.3
PING 172.25.0.3 (172.25.0.3) 56(84) bytes of data.
64 bytes from 172.25.0.3: icmp_seq=1 ttl=64 time=0.100 ms
64 bytes from 172.25.0.3: icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from 172.25.0.3: icmp_seq=3 ttl=64 time=0.050 ms
^C
--- 172.25.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.050/0.066/0.100/0.025 ms
```

Les options possibles au niveau de la gestion du réseau sont vaste. Voici deux exemples supplémentaires.

Il est possible d'ajouter une adresse d'un serveur DNS au lancement d'un conteneur :

```
[root@9f36a628c72b /]# exit
exit
root@debian9:~# docker stop mongo2
mongo2
root@debian9:~# docker rm mongo2
mongo2
root@debian9:~# docker run -it --name mongo2 --dns 8.8.8.8 i2tch/mongodb2 bash
```

```
root@735599480b45:/# cat /etc/resolv.conf
search home
nameserver 8.8.8.8
root@735599480b45:/#
```

ou de passer une entrée pour le fichier **/etc/hosts** :

```
root@735599480b45:/# exit
exit
root@debian9:~# docker stop mongo2
mongo2
root@debian9:~# docker rm mongo2
mongo2
root@debian9:~# docker run -it --name mongo2 --add-host mickeymouse:127.0.0.1 i2tch/mongodb2 bash
root@718e7eab814f:/# cat /etc/hosts
127.0.0.1   localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1   mickeymouse
172.17.0.2  718e7eab814f
```

Host

Ce type de réseau est utilisé dans le cas où le réseau ne doit pas être isolé de l'hôte tout en isolant les autres aspects du conteneur. Les conteneurs utilisent la même interface que l'hôte en prenant la même adresse IP que la machine hôte.

Dans le cas de la machine virtuelle, l'adresse IP de l'interface connectée au réseau local est **10.0.2.60** :

```
root@debian9:~# ip addr show ens18
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
```

```
    link/ether 08:00:27:2e:77:01 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.60/24 brd 10.0.2.255 scope global dynamic ens18
       valid_lft 83772sec preferred_lft 83772sec
    inet6 fe80::a00:27ff:fe2e:7701/64 scope link
       valid_lft forever preferred_lft forever
```

Démarrez un conteneur à partir de l'image **centos** dans un réseau de type **host** :

```
root@debian9:~# docker run -it --rm --network host --name centos3 centos bash
[root@debian9 /]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:2e:77:01 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.60/24 brd 10.0.2.255 scope global dynamic ens18
       valid_lft 82102sec preferred_lft 82102sec
    inet6 fe80::a00:27ff:fe2e:7701/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:38:f1:e7:ee 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::42:38ff:fef1:e7ee/64 scope link
       valid_lft forever preferred_lft forever
[root@debian9 /]# hostname
debian9
[root@debian9 /]# exit
```

Le but de ce type de réseau est de permettre l'accès à des services dans le conteneur directement à partir de l'adresse IP de l'hôte Docker. Par exemple, un nginx dans le conteneur pourrait être joint directement sur 10.0.2.60:80 **sans** avoir besoin de passer par l'exposition du port.

Pour cette raison, dans le cas de l'option **-p** utilisé dans la cas du réseau **host**, cette option n'est pas prise en compte et produit l'avertissement **WARNING: Published ports are discarded when using host network mode**. L'utilité majeure donc du réseau **host** se trouve dans le cas où de multiples ports dans le conteneur doivent être joignables.

Important : Notez que le réseau de type **host** ne fonctionne que sous Linux. Il est donc incompatible avec Docker Desktop pour Mac, Docker Desktop pour Windows et Docker EE pour Windows Server.

None

Ce type de réseau est utilisé principalement dans le cas de l'utilisation d'un plugin réseau disponible dans le **Docker Hub**.

Il est donc possible de lancer un conteneur totalement étanche grâce au réseau **none** :

```
root@718e7eab814f:/# exit
exit
root@debian9:~# docker stop mongo2
mongo2
root@debian9:~# docker rm mongo2
mongo2
root@debian9:~# docker run -it --name mongo2 --network none i2tch/mongodb2 bash
root@332aa9930f30:/#
```

Liens

Le mécanisme des liens entre conteneurs est très puissant et permet d'atteindre un autre conteneur facilement à condition que les deux conteneurs soient dans le même réseau. Créez donc un conteneur dénommé **centos3** qui est lié au conteneur **centos2** qu'il connait aussi sous l'alias **alias** :

```
root@332aa9930f30:/# exit
exit

root@debian9:~# docker run -itd --name centos3 --link centos2:alias centos
6a315259b2946c3bf2bb69f608cbe910d87edaadedb4f805e7a4dbf6af1eb916

root@debian9:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
6a315259b294        centos              "/bin/bash"              33 seconds ago      Up 32 seconds
centos3
332aa9930f30        i2tch/mongodb2      "docker-entrypoint..."   3 minutes ago       Exited (127) 39 seconds ago
mongo2
aaed3bc8e404        centos              "/bin/bash"              16 minutes ago      Up 16 minutes
centos2
9f36a628c72b        centos              "/bin/bash"              16 minutes ago      Up 16 minutes
centos1
2169360fcbfd        centos              "/bin/bash"              20 minutes ago      Up 20 minutes
resotest
ea239635e141        testcache           "more /tmp/moment"       7 hours ago         Exited (0) 7 hours ago
test1
21b0490a93dd        i2tch/mydocker      "/entrypoint.sh my..."   7 hours ago         Exited (137) 6 hours ago
myDocker
bdb4bc0f81de        i2tch/mongodb1      "docker-entrypoint..."   18 hours ago        Created
27017/tcp           mongo1
f5b45072b831        i2tch/mongodb       "bash"                   19 hours ago        Exited (137) 6 hours ago
mongo
9731a48f126a        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 6 hours ago
cocky_gates
eacd70596e23        nginx               "nginx -g 'daemon ..."   19 hours ago        Exited (0) 19 hours ago
adoring_yonath
cffb4456e9c4        ubuntu              "/bin/bash"              20 hours ago        Exited (0) 20 hours ago
i2tch
```

```
root@debian9:~# docker exec -it centos3 bash

[root@6a315259b294 /]# ping centos2
PING alias (172.17.0.4) 56(84) bytes of data.
64 bytes from alias (172.17.0.4): icmp_seq=1 ttl=64 time=0.116 ms
64 bytes from alias (172.17.0.4): icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from alias (172.17.0.4): icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from alias (172.17.0.4): icmp_seq=4 ttl=64 time=0.070 ms
^C
--- alias ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.068/0.080/0.116/0.023 ms

[root@6a315259b294 /]# cat /etc/hosts
127.0.0.1   localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.4  alias aaed3bc8e404 centos2
172.17.0.2  6a315259b294

[root@6a315259b294 /]# exit
exit

root@debian9:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos3
172.17.0.2
```

Notez cependant qu le lien est unidirectionnel :

```
root@debian9:~# docker exec -it centos2 bash

[root@aaed3bc8e404 /]# ping centos3
```

```
ping: centos3: Name or service not known

[root@aaed3bc8e404 /]# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.054 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.071 ms
^C
--- 172.17.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.035/0.052/0.071/0.015 ms

[root@aaed3bc8e404 /]#
```

Dans le cas ci-dessus, **centos2** peut atteindre **centos3** en utilisant l'adresse IP **172.17.0.2** car **centos2** se trouve dans les deux réseaux avec les adresses IP 172.17.0.4 et 172.25.0.3 :

```
[root@aaed3bc8e404 /]# exit
exit
root@debian9:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos2
172.17.0.4172.25.0.3
```

2.2 - Lancer Wordpress dans un container

Créez le répertoire ~/wordpress et placez-vous dedans :

```
root@debian9:~# mkdir ~/wordpress && cd ~/wordpress
```

Créez un conteneur dénommé **wordpressdb** à partir de l'image **mariadb:latest** :

```
root@debian9:~/wordpress# docker run -e MYSQL_ROOT_PASSWORD=fenestros -e MYSQL_DATABASE=wordpress --name
```

```
wordpressdb -v "$PWD/database":/var/lib/mysql -d mariadb:latest
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
f2b6b4884fc8: Pull complete
26d8bdca4f3e: Pull complete
74f09e820cce: Pull complete
5390f1fe4554: Pull complete
3d3f1706a741: Pull complete
2942f66426ea: Pull complete
97ee11d39c75: Pull complete
590c46ef722b: Pull complete
32eb4b9666e5: Pull complete
fc883f98a064: Pull complete
bb8bee61bc1e: Pull complete
Digest: sha256:6135f5b851e7fe263dcf0edf3480cdab1ab28c4287e867c5d83fbe967412ea14
Status: Downloaded newer image for mariadb:latest
67831dacf002bdc21dc79b0e8483f538235d00ddd2e8aae175ef3ebf189ae14d
```

Vérifiez que le conteneur fonctionne :

```
root@debian9:~/wordpress# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
NAMES
67831dacf002        mariadb:latest      "docker-entrypoint.s…"   About a minute ago   Up 45 seconds
3306/tcp            wordpressdb
```

Créez un conteneur appellé **wordpress** lié au conteneur wordpressdb :

```
root@debian9:~/wordpress# docker run -e WORDPRESS_DB_USER=root -e WORDPRESS_DB_PASSWORD=fenestros --name
wordpress --link wordpressdb:mysql -p 10.0.2.60:80:80 -v "$PWD/html":/var/www/html -d wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
2a72cbf407d6: Pull complete
273cd543cb15: Pull complete
```

```
ec5ac8875de7: Pull complete
9106e19b56c1: Pull complete
ee2f70ac7c7d: Pull complete
7257ad6985e8: Pull complete
18f5c2055da2: Pull complete
85293a6fdd80: Pull complete
9e797eeb0c14: Pull complete
f16178842884: Pull complete
13899c06d3f8: Pull complete
70c27fe4c3c5: Pull complete
d32c8ad2d9d7: Pull complete
07fe445494e6: Pull complete
63b8de7b32fe: Pull complete
e4b721952e22: Pull complete
d9ede6dd6f74: Pull complete
0af4f74bfd92: Pull complete
e4e7c47b969f: Pull complete
69aff47f3112: Pull complete
Digest: sha256:201d004f55669dd2c0884f00fc44145fb0da8cafa465bf22cbaacecaf81138d4
Status: Downloaded newer image for wordpress:latest
9eb2f7fbfbd25307ed2f463c7eb3bef40bfa556174e68750bb76b8d032546129
```

Vérifiez que le conteneur fonctionne :

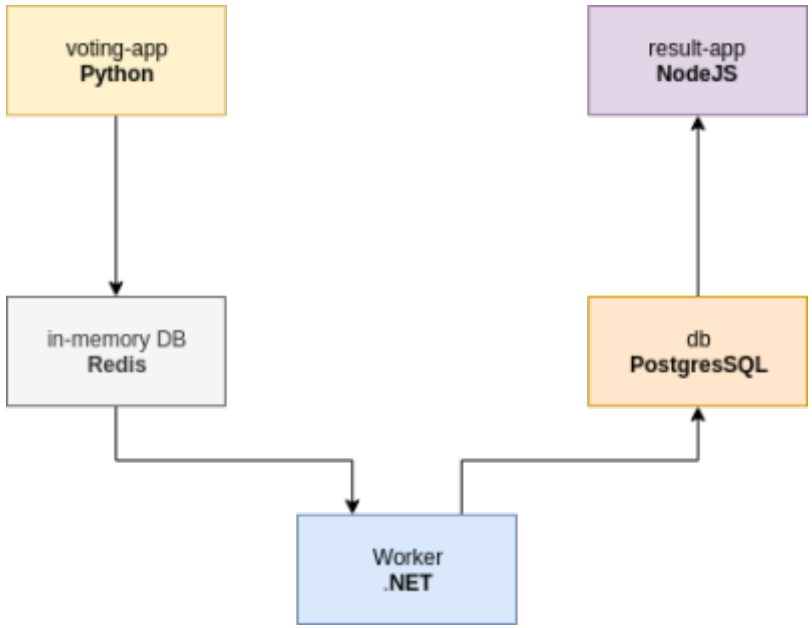
```
root@debian9:~/wordpress# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
9eb2f7fbfbd2        wordpress           "docker-entrypoint.s…"   2 minutes ago       Up About a minute
10.0.2.60:80->80/tcp   wordpress
67831dacf002        mariadb:latest      "docker-entrypoint.s…"   9 minutes ago       Up 8 minutes        3306/tcp
wordpressdb
```

Vérifiez que le Wordpress fonctionne :

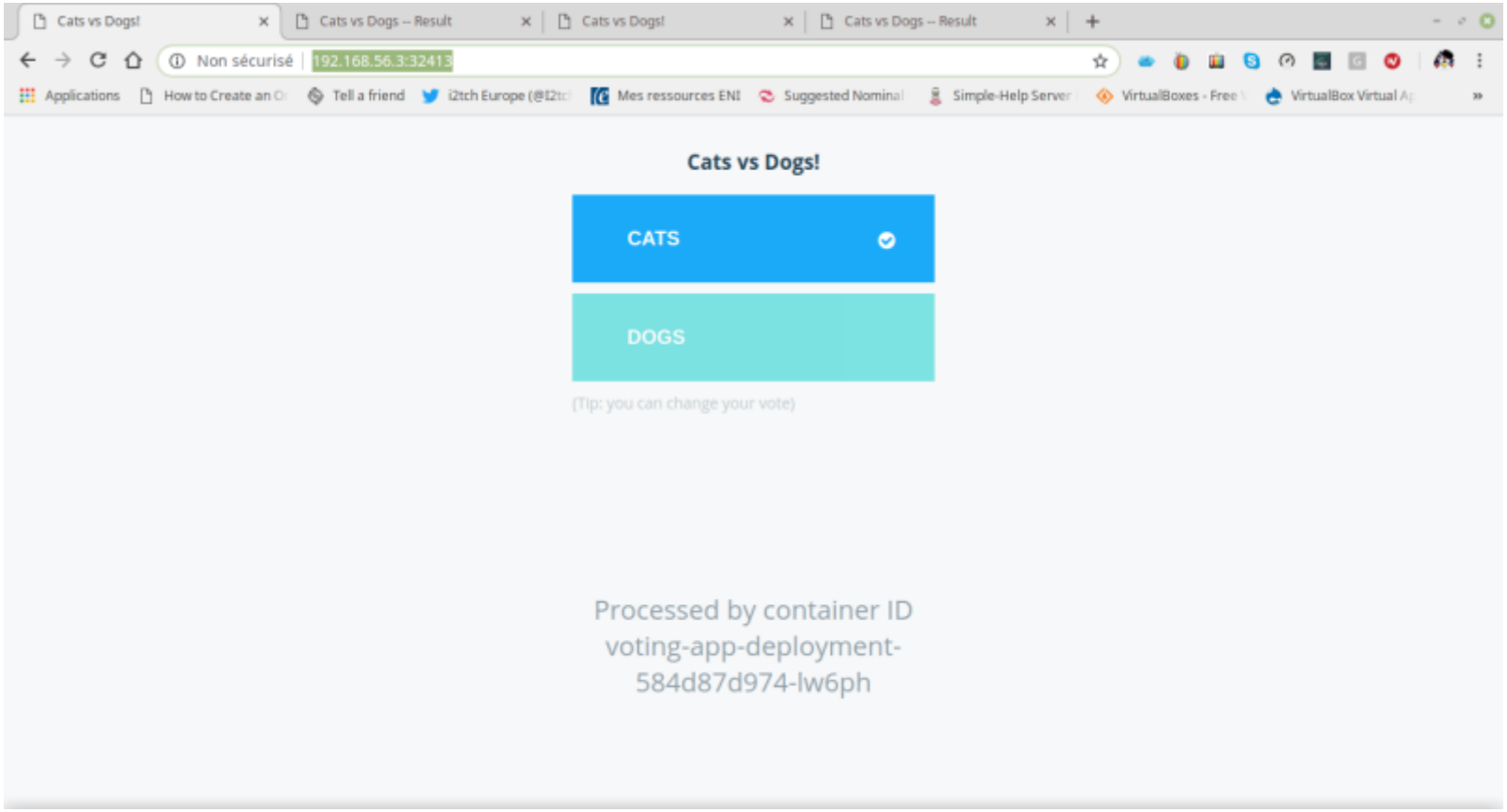
```
root@debian9:~/wordpress# lynx --dump http://10.0.2.60
   [1]WordPress
   Select a default language [English (United States)________]

   Continue

References

   1. https://wordpress.org/

root@debian9:~/wordpress# docker inspect wordpress | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.3",
                    "IPAddress": "172.17.0.3",
root@debian9:~/wordpress# lynx --dump http://172.17.0.3
   [1]WordPress
   Select a default language [English (United States)________]

   Continue

References

   1. https://wordpress.org/
```

2.3 - Gestion d'une Architecture de Microservices

Vous allez mettre en place une application simple sous forme de microservices, développé par Docker et appelé **demo-voting-app**, :

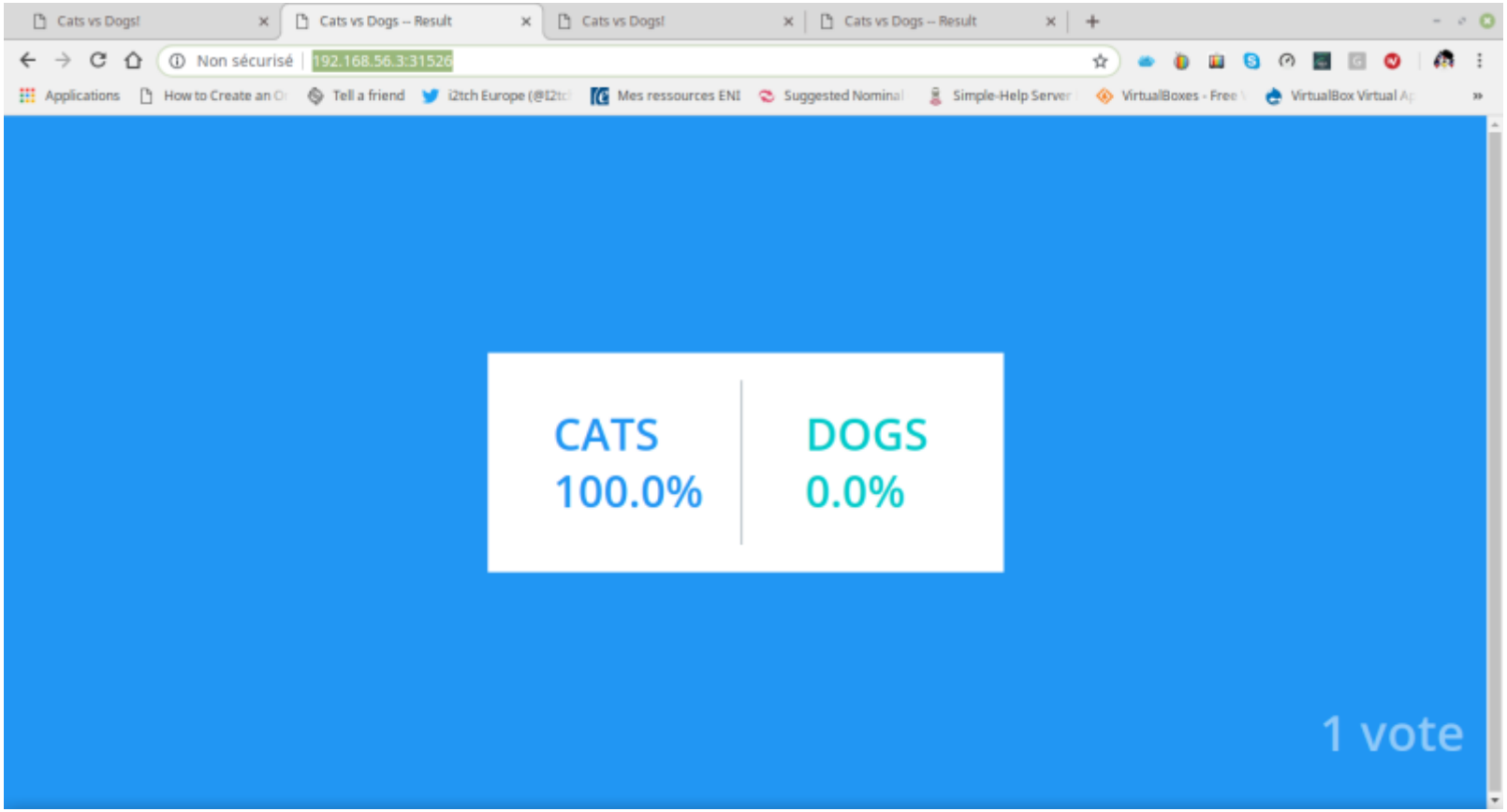


Dans cette application le conteneur **voting-app** permet de voter pour des **chats** ou des **chiens**. Cette application tourne sous Python et fournit une interface HTML :



Lors de la vote, le résultat de celle-ci est stocké dans **Redis** dans une base de données en mémoire. Le résultat est ensuite passé au conteneur **Worker** qui tourne sous .NET et qui met à jour la base de données persistante dans le conteneur **db** qui tourne sous PostgreSQL.

L'application **result-app** qui tourne sous NodeJS lit ensuite la table dans la base de données PostgreSQL et affiche le résultat sous forme HTML :



Cette application peut être mise en place sous docker avec les commandes suivantes :

```
docker run -d --name=redis redis
docker run -d --name=db -e POSTGRES_PASSWORD=postgres -e POSTGRES_USER=postgres postgres:9.4
docker run -d --name=vote -p 5000:80 --link redis:redis dockersamples/examplevotingapp_vote
docker run -d --name=result -p 5001:80 --link db:db dockersamples/examplevotingapp_result
docker run -d --name=worker --link db:db --link redis:redis dockersamples/examplevotingapp_worker
```

Cette solution utilise un réseau de type Bridge. Ce type de réseau est limité aux conteneurs d'un hôte unique exécutant Docker. Les conteneurs ne peuvent communiquer qu'entre eux et ils ne sont pas accessibles depuis l'extérieur. Pour que les conteneurs sur le réseau puissent communiquer ou être accessibles du monde extérieur, il faut configurer le mappage de port.

LAB #3 - Superviser les Conteneurs

3.1 - Les Journaux

Consultez les logs d'un conteneur :

```
root@debian9:~# docker logs mongo2
root@332aa9930f30:/# ip addr
bash: ip: command not found
root@332aa9930f30:/# ip address
bash: ip: command not found
root@332aa9930f30:/# ifconfig
bash: ifconfig: command not found
root@332aa9930f30:/# ls
bin  boot  core  data  dev  docker-entrypoint-initdb.d  entrypoint.sh  etc  home  lib  lib64  media  mnt  opt
proc  root  run    sbin  selinux  srv  sys  tmp  usr  var
root@332aa9930f30:/# which ip
root@332aa9930f30:/# which ifconfig
root@332aa9930f30:/# docker run -itd --name centos3 --link centos2:alias centos
bash: docker: command not found
root@332aa9930f30:/# exit
exit
```

3.2 - Les Processus

Consultez les processus d'un conteneur :

```
root@debian9:~# docker top centos3
UID                 PID                 PPID                C                   STIME               TTY
TIME                CMD
root                31073               31060               0                   10:20               pts/0
```

```
00:00:00            /bin/bash
```

3.3 - L'Activité en Continu

Pour voir l'activité d'un conteneur, utilisez la commande suivante :

```
root@debian9:~# docker stats centos3
CONTAINER           CPU %               MEM USAGE / LIMIT   MEM %               NET I/O             BLOCK I/O
PIDS
centos3             0.00%               0B / 0B             0.00%               4.37kB / 952B       61.4kB / 0B
0
```
