

Version : **2024.01**

Dernière mise-à-jour : 2024/02/21 13:40

DOF604 - Gestion des Volumes, du Réseau et de la Supervision des Conteneurs

Contenu du Module

- **DOF604 - Gestion des Volumes, du Réseau et des Ressources**

- Contenu du Module
 - LAB #1 - Gestion des Volumes
 - 1.1 - Gestion Automatique par Docker
 - 1.2 - Gestion Manuelle d'un Volume
 - 1.3 - Gestion Manuelle d'un Bindmount
 - LAB #2 - Gestion du Réseau
 - 2.1 - L'Approche Réseau Docker
 - Bridge
 - Host
 - None
 - Liens
 - 2.2 - Lancer Wordpress dans un container
 - 2.3 - Gestion d'une Architecture de Microservices
 - LAB #3 - Superviser les Conteneurs
 - 3.1 - Les Journaux
 - 3.2 - Les Processus
 - 3.3 - L'Activité en Continu
-

LAB #1 - Gestion des Volumes

Lancez le conteneur **mongo2** :

```
root@debian11:~# docker start mongo2
mongo2
```

1.1 - Gestion Automatique de Volumes par Docker

Vérifiez que le processus est bien démarré dans le conteneur :

```
root@debian11:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
880733c6bdc3	i2tch/mongodb2	"docker-entrypoint.s..."	4 days ago	Up 43 seconds	27017/tcp	mongo2

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

```
root@debian11:~# docker inspect mongo2
...
  "Mounts": [
    {
      "Type": "volume",
      "Name": "537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703",
      "Source":
"/var/lib/docker/volumes/537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703/_data",
      "Destination": "/data/db",
      "Driver": "local",
      "Mode": "",
      "RW": true,
      "Propagation": ""
    },
  ],
```

```
{
  "Type": "volume",
  "Name": "58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59",
  "Source":
"/var/lib/docker/volumes/58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59/_data",
  "Destination": "/data/configdb",
  "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@debian11:~# ls
/var/lib/docker/volumes/537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703/_data
collection-0-2505194079268383602.wt  index-5-2505194079268383602.wt  storage.bson
collection-2-2505194079268383602.wt  index-6-2505194079268383602.wt  WiredTiger
collection-4-2505194079268383602.wt  journal                          WiredTigerLAS.wt
diagnostic.data                      _mdb_catalog.wt                 WiredTiger.lock
index-1-2505194079268383602.wt       mongod.lock                     WiredTiger.turtle
index-3-2505194079268383602.wt       sizeStorer.wt                  WiredTiger.wt
```

Arrêtez et supprimez le conteneur **mongo2** :

```
root@debian11:~# docker stop mongo2
mongo2
```

```
root@debian11:~# docker rm mongo2
mongo2
```

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

```
root@debian11:~# docker run -d --name mongo2 i2tch/mongodb2
1980be05ac73d70979f5e932f1a58b6526ae1001a335fd8ca010bbfaac48ca5e

root@debian11:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1980be05ac73	i2tch/mongodb2	"docker-entrypoint.s..."	5 seconds ago	Up 4 seconds	27017/tcp	mongo2

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

```
root@debian11:~# docker inspect mongo2
...
  "Mounts": [
    {
      "Type": "volume",
      "Name": "2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48",
      "Source":
"/var/lib/docker/volumes/2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48/_data",
      "Destination": "/data/configdb",
      "Driver": "local",
      "Mode": "",
      "RW": true,
      "Propagation": ""
    },
    {
      "Type": "volume",
      "Name": "4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605",
      "Source":
"/var/lib/docker/volumes/4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605/_data",
      "Destination": "/data/db",
```

```

    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  },
],
...

```



Important : Notez que le répertoire des données du précédent conteneur, **/var/lib/docker/volumes/537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703/_data** n'est pas le même que le conteneur courant **/var/lib/docker/volumes/4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605/_data**.

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

```

root@debian11:~# ls -l /var/lib/docker/volumes/
total 60
drwx-----x 3 root root 4096 Dec 15 14:39 2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48
drwx-----x 3 root root 4096 Dec 10 17:12 396ad783162131dcc92a649366ab79e24720bf866ce6803868e2ba2df8e90074
drwx-----x 3 root root 4096 Dec 15 14:39 4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605
drwx-----x 3 root root 4096 Dec 10 17:16 537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703
drwx-----x 3 root root 4096 Dec 10 17:16 58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59
drwx-----x 3 root root 4096 Dec 10 18:32 7d7b25232f6ec411dc9dfb888048eebbee919eb2cedc301969bc325a8a8d055e
brw----- 1 root root 8, 33 Dec 15 09:56 backingFsBlockDev
drwx-----x 3 root root 4096 Dec 10 17:12 f766fb3cd11eee28312b8804c1439c0f7b0a7d58a0ce4d61ba50af17b7630c8f
-rw----- 1 root root 65536 Dec 15 14:39 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 évidemment 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@debian11:~# docker stop mongo2
mongo2
```

```
root@debian11:~# docker rm mongo2
mongo2
```

```
root@debian11:~# docker run -d --name mongo2 -v persistent_data:/data/db i2tch/mongodb2
8b2207df56a727a8472b2949d7a83a1925e751ad0216fe98b6a3db83230c0988
```

```
root@debian11:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8b2207df56a7	i2tch/mongodb2	"docker-entrypoint.s..."	6 seconds ago	Up 6 seconds	27017/tcp	mongo2

```
root@debian11:~# docker logs mongo2
```

```
2023-12-15T13:45:40.845+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify -
-sslDisabledProtocols 'none'
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-
bit host=8b2207df56a7
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] db version v4.1.9
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] git version: a5fa363117062a20d6056c76e01edb3a08f71b7c
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] allocator: tcmalloc
```

```
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] modules: none
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] build environment:
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten]     distmod: ubuntu1804
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten]     distarch: x86_64
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten]     target_arch: x86_64
2023-12-15T13:45:40.847+0000 I CONTROL [initandlisten] options: { net: { bindIp: "*" } }
2023-12-15T13:45:40.847+0000 I STORAGE [initandlisten]
2023-12-15T13:45:40.847+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly
recommended with the WiredTiger storage engine
2023-12-15T13:45:40.847+0000 I STORAGE [initandlisten] **           See
http://dochub.mongodb.org/core/prodnotes-filesystem
2023-12-15T13:45:40.847+0000 I STORAGE [initandlisten] wiredtiger_open config:
create,cache_size=7485M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fa
st),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statisti
cs_log=(wait=0),verbose=(recovery_progress),
2023-12-15T13:45:41.621+0000 I STORAGE [initandlisten] WiredTiger message [1702647941:621009][1:0x7f381cfeaa40],
txn-recover: Set global recovery timestamp: (0,0)
2023-12-15T13:45:41.665+0000 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2023-12-15T13:45:41.708+0000 I STORAGE [initandlisten] Timestamp monitor starting
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten]
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten] ** NOTE: This is a development version (4.1.9) of
MongoDB.
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten] **           Not recommended for production.
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten]
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the
database.
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten] **           Read and write access to data and
configuration is unrestricted.
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten]
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten]
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled
is 'always'.
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten] **           We suggest setting it to 'never'
2023-12-15T13:45:41.727+0000 I CONTROL [initandlisten]
```

```
2023-12-15T13:45:41.728+0000 I STORAGE [initandlisten] createCollection: admin.system.version with provided
UUID: 1e149d35-bbda-46a7-bc8e-3a300abb052f
2023-12-15T13:45:41.815+0000 I INDEX [initandlisten] index build: done building index _id_ on ns
admin.system.version
2023-12-15T13:45:41.815+0000 I SHARDING [initandlisten] Marking collection admin.system.version as collection
version: <unsharded>
2023-12-15T13:45:41.815+0000 I COMMAND [initandlisten] setting featureCompatibilityVersion to 4.2
2023-12-15T13:45:41.819+0000 I SHARDING [initandlisten] Marking collection local.system.replset as collection
version: <unsharded>
2023-12-15T13:45:41.819+0000 I SHARDING [initandlisten] Marking collection admin.system.roles as collection
version: <unsharded>
2023-12-15T13:45:41.819+0000 I STORAGE [initandlisten] createCollection: local.startup_log with generated UUID:
759d58f7-7f35-441f-bd93-f090f4a14427
2023-12-15T13:45:41.856+0000 I INDEX [initandlisten] index build: done building index _id_ on ns
local.startup_log
2023-12-15T13:45:41.856+0000 I SHARDING [initandlisten] Marking collection local.startup_log as collection
version: <unsharded>
2023-12-15T13:45:41.856+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with
directory '/data/db/diagnostic.data'
2023-12-15T13:45:41.857+0000 I NETWORK [initandlisten] Listening on /tmp/mongodb-27017.sock
2023-12-15T13:45:41.857+0000 I NETWORK [initandlisten] Listening on 0.0.0.0
2023-12-15T13:45:41.857+0000 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as
collection version: <unsharded>
2023-12-15T13:45:41.857+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2023-12-15T13:45:41.857+0000 I STORAGE [LogicalSessionCacheRefresh] createCollection: config.system.sessions
with generated UUID: 6936ac77-1578-4a80-b907-f00774284e52
2023-12-15T13:45:41.894+0000 I INDEX [LogicalSessionCacheRefresh] index build: done building index _id_ on ns
config.system.sessions
2023-12-15T13:45:41.932+0000 I INDEX [LogicalSessionCacheRefresh] index build: starting on
config.system.sessions properties: { v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns:
"config.system.sessions", expireAfterSeconds: 1800 } using method: Hybrid
2023-12-15T13:45:41.932+0000 I INDEX [LogicalSessionCacheRefresh] build may temporarily use up to 500
megabytes of RAM
2023-12-15T13:45:41.932+0000 I INDEX [LogicalSessionCacheRefresh] index build: collection scan done. scanned 0
```

```
total records in 0 seconds
2023-12-15T13:45:41.932+0000 I INDEX    [LogicalSessionCacheRefresh] index build: inserted 0 keys from external
sorter into index in 0 seconds
2023-12-15T13:45:41.933+0000 I INDEX    [LogicalSessionCacheRefresh] index build: done building index
lsidTTLIndex on ns config.system.sessions
```

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

```
root@debian11:~# ls -l /var/lib/docker/volumes/
total 68
drwx-----x 3 root root 4096 Dec 15 14:39 2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48
drwx-----x 3 root root 4096 Dec 10 17:12 396ad783162131dcc92a649366ab79e24720bf866ce6803868e2ba2df8e90074
drwx-----x 3 root root 4096 Dec 15 14:39 4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605
drwx-----x 3 root root 4096 Dec 10 17:16 537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703
drwx-----x 3 root root 4096 Dec 10 17:16 58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59
drwx-----x 3 root root 4096 Dec 10 18:32 7d7b25232f6ec411dc9dfb888048eebbee919eb2cedc301969bc325a8a8d055e
brw----- 1 root root 8, 33 Dec 15 09:56 backingFsBlockDev
drwx-----x 3 root root 4096 Dec 10 17:12 f766fb3cd11eee28312b8804c1439c0f7b0a7d58a0ce4d61ba50af17b7630c8f
drwx-----x 3 root root 4096 Dec 15 14:45 fa3330b3415f534a7d0053ba95d74fe02765cde317c11cac9691c3132cec3d47
-rw----- 1 root root 65536 Dec 15 14:45 metadata.db
drwx-----x 3 root root 4096 Dec 15 14:45 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@debian11:~# docker stop mongo2
mongo2

root@debian11:~# docker rm mongo2
mongo2

root@debian11:~# docker run -d --name mongo2 -v persistent_data:/data/db i2tch/mongodb2
cc7cc8f3b43346fe47cc5107225b0b98851a73a9b2938530077ca7a3207581a0
```

```
root@debian11:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
cc7cc8f3b433	i2tch/mongodb2	"docker-entrypoint.s..."	18 seconds ago	Up 18 seconds	27017/tcp	mongo2

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

```
root@debian11:~# docker inspect mongo2
```

```
...
  "Mounts": [
    {
      "Type": "volume",
      "Name": "persistent_data",
      "Source": "/var/lib/docker/volumes/persistent_data/_data",
      "Destination": "/data/db",
      "Driver": "local",
      "Mode": "z",
      "RW": true,
      "Propagation": ""
    },
    {
      "Type": "volume",
      "Name": "cef4abb286a1b519e6cdfb4a2327659cd124dc1513ed55288c014e61a8bf27b7",
      "Source":
"/var/lib/docker/volumes/cef4abb286a1b519e6cdfb4a2327659cd124dc1513ed55288c014e61a8bf27b7/_data",
      "Destination": "/data/configdb",
      "Driver": "local",
      "Mode": "",
      "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@debian11:~# docker volume create myvolume
myvolume
```

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

```
root@debian11:~# docker volume ls
DRIVER      VOLUME NAME
local       2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48
local       4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605
local       7d7b25232f6ec411dc9dfb888048eebb9e919eb2cedc301969bc325a8a8d055e
local       396ad783162131dcc92a649366ab79e24720bf866ce6803868e2ba2df8e90074
local       537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703
local       58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59
local       cef4abb286a1b519e6cdfb4a2327659cd124dc1513ed55288c014e61a8bf27b7
local       f766fb3cd11eee28312b8804c1439c0f7b0a7d58a0ce4d61ba50af17b7630c8f
local       fa3330b3415f534a7d0053ba95d74fe02765cde317c11cac9691c3132cec3d47
local       myvolume
local       persistent_data
```

Notez maintenant l'emplacement physique du volume créé :

```
root@debian11:~# docker volume inspect myvolume
[
  {
```

```
    "CreatedAt": "2023-12-15T14:50:18+01:00",  
    "Driver": "local",  
    "Labels": null,  
    "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",  
    "Name": "myvolume",  
    "Options": null,  
    "Scope": "local"  
  }  
]
```

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

```
root@debian11:~# touch /var/lib/docker/volumes/myvolume/_data/test-file
```

Démarrez maintenant un conteneur qui utilise ce volume :

```
root@debian11:~# docker run -it --name ubuntu-volume --mount source=myvolume,target=/myvolume ubuntu bash
```



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

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

```
root@ff76d3820051:/# cd myvolume/  
root@ff76d3820051:/myvolume# ls  
test-file
```

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

```
root@ff76d3820051:/myvolume# touch container_volume
```

```
root@ff76d3820051:/myvolume# exit
exit
root@debian11:~#
```

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

```
root@debian11:~# ls -l /var/lib/docker/volumes/myvolume/_data/
total 0
-rw-r--r-- 1 root root 0 Dec 15 14:55 container_volume
-rw-r--r-- 1 root root 0 Dec 15 14:51 test-file
```



Important : Notez que les deux fichiers témoins sont visibles.

```
root@debian11:~# docker rm ubuntu-volume
ubuntu-volume

root@debian11:~# ls -l /var/lib/docker/volumes/myvolume/_data/
total 0
-rw-r--r-- 1 root root 0 Dec 15 14:55 container_volume
-rw-r--r-- 1 root root 0 Dec 15 14:51 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@debian11:~# docker run -it --rm --name ubuntu-volume --mount source=myvolume1,target=/myvolume1 ubuntu bash
root@5547f3231534:/# ls
```

```
bin   dev   home  lib32  libx32  mnt      opt   root  sbin  sys  usr
boot  etc   lib   lib64  media   myvolume1  proc  run   srv   tmp  var
```

```
root@5547f3231534:/# cd myvolume1
```

```
root@5547f3231534:/myvolume1# touch file_myvolume1
```

```
root@5547f3231534:/myvolume1# exit
```

```
exit
```

```
root@debian11:~#
```

Notez que Docker a automatiquement créé le volume :

```
root@debian11:~# docker volume ls
```

```
DRIVER      VOLUME NAME
```

```
local      2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48
local      4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605
local      7d7b25232f6ec411dc9dfb888048eebbe919eb2cedc301969bc325a8a8d055e
local      396ad783162131dcc92a649366ab79e24720bf866ce6803868e2ba2df8e90074
local      537cc5d0f0f0aa0af9dd959b45fc9fcbe8467a868b9d61919991366a2813f703
local      58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59
local      cef4abb286a1b519e6cdfb4a2327659cd124dc1513ed55288c014e61a8bf27b7
local      f766fb3cd11eee28312b8804c1439c0f7b0a7d58a0ce4d61ba50af17b7630c8f
local      fa3330b3415f534a7d0053ba95d74fe02765cde317c11cac9691c3132cec3d47
local      myvolume
local      myvolume1
local      persistent_data
```

```
root@debian11:~# ls -l /var/lib/docker/volumes/myvolume1/_data/
```

```
total 0
```

```
-rw-r--r-- 1 root root 0 Dec 15 15:01 file_myvolume1
```

1.3 - Gestion Manuelle d'un Bindmount

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 contrôlé par la CLI Docker.

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

```
root@debian11:~# mkdir bindmount
root@debian11:~# touch bindmount/test_bind
```

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

```
root@debian11:~# docker run -it --name ubuntu-volume --mount type=bind,source=/root/bindmount,target=/bindmount
ubuntu bash

root@1cd3cc50e6c0:/# ls
bin      boot  etc   lib   lib64  media  opt   root  sbin  sys  usr
bindmount dev   home  lib32 libx32  mnt    proc  run   srv   tmp  var

root@1cd3cc50e6c0:/# cd bindmount/

root@1cd3cc50e6c0:/bindmount# ls
test_bind

root@1cd3cc50e6c0:/bindmount# touch container_bind

root@1cd3cc50e6c0:/bindmount# ls
container_bind  test_bind

root@1cd3cc50e6c0:/bindmount# exit
exit
```

```
root@debian11:~#
```

Contrôlez la présence du fichier témoin :

```
root@debian11:~# ls -l bindmount/
total 0
-rw-r--r-- 1 root root 0 Dec 15 15:27 container_bind
-rw-r--r-- 1 root root 0 Dec 15 15:25 test_bind

root@debian11:~# docker rm ubuntu-volume
ubuntu-volume

root@debian11:~# ls -l bindmount/
total 0
-rw-r--r-- 1 root root 0 Dec 15 15:27 container_bind
-rw-r--r-- 1 root root 0 Dec 15 15:25 test_bind
```

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

```
root@debian11:~# docker volume ls
DRIVER      VOLUME NAME
local       2ec1edeca3acd91aede62091e0d96252eb31e403a00fef324ca2244b2952bb48
local       4abe232050675d853d0a4d8beefe31f884e1252c985828c3be47a983aac58605
local       7d7b25232f6ec411dc9dfb888048eebb919eb2cedc301969bc325a8a8d055e
local       396ad783162131dcc92a649366ab79e24720bf866ce6803868e2ba2df8e90074
local       537cc5d0f0f0aa0af9dd959b45fc9fcbce8467a868b9d61919991366a2813f703
local       58795fb69d54b87b11fcb6ab752a1fa10736e5aa37c8a14b1c36db1306853a59
local       cef4abb286a1b519e6cdfb4a2327659cd124dc1513ed55288c014e61a8bf27b7
local       f766fb3cd11eee28312b8804c1439c0f7b0a7d58a0ce4d61ba50af17b7630c8f
local       fa3330b3415f534a7d0053ba95d74fe02765cde317c11cac9691c3132cec3d47
local       myvolume
local       myvolume1
local       persistent_data
```

LAB #2 - Gestion du Réseau

2.1 - L'Approche Réseau Docker

Docker fournit trois réseaux par défaut :

```
root@debian11:~# docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
2473b0d9324a    bridge    bridge      local
b0a285caf920    host      host        local
a2da9933cdce    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@debian11:~# ip addr show docker0
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:07:c9:88:32 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::42:7ff:fec9:8832/64 scope link
        valid_lft forever preferred_lft forever
```

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

```
root@debian11:~# docker run -itd --name=resotest centos
```

```
2126924504d8dedb920728cc7c2a6c73e87f8c5c3d13c17c4fcc2bfe8ff93cc9
```

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

```
root@debian11:~# docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "2473b0d9324a421018cdf501060801d34c599991bed76751ae328bc68126a180",
    "Created": "2023-12-15T09:56:50.183135221+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": {
      "2126924504d8dedb920728cc7c2a6c73e87f8c5c3d13c17c4fcc2bfe8ff93cc9": {
        "Name": "resotest",
        "EndpointID": "e9ae0ba15d4588571fe77a9a8e1564e92620f9532bed8ee38d060b954116b20c",
```

```
    "MacAddress": "02:42:ac:11:00:03",
    "IPv4Address": "172.17.0.3/16",
    "IPv6Address": ""
  },
  "cc7cc8f3b43346fe47cc5107225b0b98851a73a9b2938530077ca7a3207581a0": {
    "Name": "mongo2",
    "EndpointID": "d9f30326f473f12a45f6aacf97cee0f12c9c545e45ed7808b0cba809fa48ae9a",
    "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@debian11:~# docker network disconnect bridge resotest
```

```
root@debian11:~# docker network inspect bridge
```

```
[
  {
    "Name": "bridge",
    "Id": "2473b0d9324a421018cdf501060801d34c599991bed76751ae328bc68126a180",
    "Created": "2023-12-15T09:56:50.183135221+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": {
      "cc7cc8f3b43346fe47cc5107225b0b98851a73a9b2938530077ca7a3207581a0": {
        "Name": "mongo2",
        "EndpointID": "d9f30326f473f12a45f6aacf97cee0f12c9c545e45ed7808b0cba809fa48ae9a",
        "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@debian11:~# docker network create -d bridge --subnet 172.25.0.0/16 --gateway 172.25.0.1 my-bridged-network
4d26c1192dd7b25c3c787ef41b7bfb94d0eb989d230f33906db6d54ed9c128cc
```

```
root@debian11:~# docker network ls
NETWORK ID        NAME                DRIVER            SCOPE
2473b0d9324a     bridge             bridge            local
b0a285caf920     host               host              local
4d26c1192dd7     my-bridged-network bridge            local
a2da9933cdce     none               null              local
```

Bien évidemment, ce réseau est actuellement vide :

```
root@debian11:~# docker network inspect my-bridged-network
[
  {
    "Name": "my-bridged-network",
    "Id": "4d26c1192dd7b25c3c787ef41b7bfb94d0eb989d230f33906db6d54ed9c128cc",
```

```
    "Created": "2023-12-15T15:34:02.824127656+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@debian11:~# docker run -itd --name=centos1 centos
cb2875ab1059e66308228d9179b810db748ad287453cf758206a7025f57b0176

root@debian11:~# docker run -itd --name=centos2 centos
fc417b22a20d3f9de674889962452bfe453ced1fc389410b225055f865cb817f
```

```
root@debian11:~# docker inspect --format='{{json .NetworkSettings.Networks}}' centos1
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"2473b0d9324a421018cdf501060801d34c599991bed76751ae328bc68126a180","EndpointID":"a0e5d67f4652be807583fba65a76bf1a0462200aa198c92ab29845f88c45d559","Gateway":"172.17.0.1","IPAddress":"172.17.0.3","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:11:00:03","DriverOpts":null}}

root@debian11:~# docker inspect --format='{{json .NetworkSettings.Networks}}' centos2
{"bridge":{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"2473b0d9324a421018cdf501060801d34c599991bed76751ae328bc68126a180","EndpointID":"15ed4b05703524f64015b12683ea01729dd26a7dfe68ab638383ea075d8d1428","Gateway":"172.17.0.1","IPAddress":"172.17.0.4","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:11:00:04","DriverOpts":null}}

root@debian11:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos1
172.17.0.3

root@debian11:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos2
172.17.0.4
```

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

```
root@debian11:~# docker network connect my-bridged-network centos1

root@debian11:~# docker network inspect my-bridged-network
[
  {
    "Name": "my-bridged-network",
    "Id": "4d26c1192dd7b25c3c787ef41b7bfb94d0eb989d230f33906db6d54ed9c128cc",
    "Created": "2023-12-15T15:34:02.824127656+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": {
      "cb2875ab1059e66308228d9179b810db748ad287453cf758206a7025f57b0176": {
        "Name": "centos1",
        "EndpointID": "a7de8c07d195168c20548b33b506073caa03c16770f330a2e576aedcda25662c",
        "MacAddress": "02:42:ac:19:00:02",
        "IPv4Address": "172.25.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
```

```
root@debian11:~# 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@debian11:~# docker network connect my-bridged-network centos2

root@debian11:~# docker network inspect my-bridged-network
[
  {
    "Name": "my-bridged-network",
    "Id": "4d26c1192dd7b25c3c787ef41b7bfb94d0eb989d230f33906db6d54ed9c128cc",
    "Created": "2023-12-15T15:34:02.824127656+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": {
      "cb2875ab1059e66308228d9179b810db748ad287453cf758206a7025f57b0176": {
        "Name": "centos1",
        "EndpointID": "a7de8c07d195168c20548b33b506073caa03c16770f330a2e576aedcda25662c",
```

```
        "MacAddress": "02:42:ac:19:00:02",
        "IPv4Address": "172.25.0.2/16",
        "IPv6Address": ""
    },
    "fc417b22a20d3f9de674889962452bfe453ced1fc389410b225055f865cb817f": {
        "Name": "centos2",
        "EndpointID": "8467b7d4233dbf855d0538dbc9b1fe718874434913baae403391b3da81ccb92b",
        "MacAddress": "02:42:ac:19:00:03",
        "IPv4Address": "172.25.0.3/16",
        "IPv6Address": ""
    }
},
"Options": {},
"Labels": {}
}
]
```

```
root@debian11:~# 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@debian11:~# docker exec -it centos1 bash
[root@cb2875ab1059 /]#
```

Vérifiez que la connectivité fonctionne :

```
[root@cb2875ab1059 /]# 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=0.140 ms
64 bytes from 172.25.0.3: icmp_seq=2 ttl=64 time=0.099 ms
^C
--- 172.25.0.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
```

```
rtt min/avg/max/mdev = 0.099/0.119/0.140/0.023 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@cb2875ab1059 /]# exit
exit

root@debian11:~# docker stop mongo2
mongo2

root@debian11:~# docker rm mongo2
mongo2

root@debian11:~# docker run -it --name mongo2 --dns 8.8.8.8 i2tch/mongodb2 bash

root@aa2717305397:/# cat /etc/resolv.conf
nameserver 8.8.8.8
```

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

```
root@aa2717305397:/# exit
exit

root@debian11:~# docker stop mongo2
mongo2

root@debian11:~# docker rm mongo2
mongo2

root@debian11:~# docker run -it --name mongo2 --add-host mickeymouse:127.0.0.1 i2tch/mongodb2 bash
root@519423c7fb32:/# 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     519423c7fb32

root@519423c7fb32:/# exit
exit
root@debian11:~#
```

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 **10.0.2.46** :

```
root@debian11:~# ip addr show ens18
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 56:a3:fd:18:02:6d brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 10.0.2.46/24 brd 10.0.2.255 scope global noprefixroute ens18
        valid_lft forever preferred_lft forever
    inet6 fe80::54a3:fdff:fe18:26d/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

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

```
root@debian11:~# docker run -it --rm --network host --name centos3 centos bash

[root@debian11 /]# ip a
```

```
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: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 56:a3:fd:18:02:6d brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 10.0.2.46/24 brd 10.0.2.255 scope global noprefixroute ens18
        valid_lft forever preferred_lft forever
    inet6 fe80::54a3:fdff:fe18:26d/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:07:c9:88:32 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::42:7ff:fec9:8832/64 scope link
        valid_lft forever preferred_lft forever
102: br-4d26c1192dd7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:2d:69:ac:d5 brd ff:ff:ff:ff:ff:ff
    inet 172.25.0.1/16 brd 172.25.255.255 scope global br-4d26c1192dd7
        valid_lft forever preferred_lft forever
    inet6 fe80::42:2dff:fe69:acd5/64 scope link
        valid_lft forever preferred_lft forever
104: vethc5ca04a@if103: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group
default
    link/ether 42:98:9a:1c:41:76 brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::4098:9aff:fe1c:4176/64 scope link
        valid_lft forever preferred_lft forever
106: veth6a46250@if105: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group
default
    link/ether 5e:9d:9a:86:23:b0 brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet6 fe80::5c9d:9aff:fe86:23b0/64 scope link
```

```
    valid_lft forever preferred_lft forever
108: vethc5ccfca@if107: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-4d26c1192dd7 state UP
group default
    link/ether f2:37:cb:08:ff:8f brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::f037:cbff:fe08:ff8f/64 scope link
        valid_lft forever preferred_lft forever
110: vetha87ff61@if109: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-4d26c1192dd7 state UP
group default
    link/ether 2e:e0:2d:5c:5d:c7 brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet6 fe80::2ce0:2dff:fe5c:5dc7/64 scope link
        valid_lft forever preferred_lft forever

[root@debian11 /]# hostname
debian11

[root@debian11 /]# exit
exit
root@debian11:~#
```

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 10.0.2.46:80 **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@debian11:~# docker stop mongo2
mongo2
root@debian11:~# docker rm mongo2
mongo2
root@debian11:~# docker run -it --name mongo2 --network none i2tch/mongodb2 bash
root@5bfbf0306ad7:/#
```

===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 connaît aussi sous l'alias **alias** :

<code>

```
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
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 centos2	centos	"/bin/bash"	16 minutes ago	Up 16 minutes
9f36a628c72b centos1	centos	"/bin/bash"	16 minutes ago	Up 16 minutes
2169360fcbfd resotest	centos	"/bin/bash"	20 minutes ago	Up 20 minutes
ea239635e141 test1	testcache	"more /tmp/moment"	7 hours ago	Exited (0) 7 hours ago
21b0490a93dd myDocker	i2tch/mydocker	"/entrypoint.sh my..."	7 hours ago	Exited (137) 6 hours ago
bdb4bc0f81de 27017/tcp	i2tch/mongodb1 mongo1	"docker-entrypoint..."	18 hours ago	Created
f5b45072b831 mongo	i2tch/mongodb	"bash"	19 hours ago	Exited (137) 6 hours ago
9731a48f126a cocky_gates	nginx	"nginx -g 'daemon ...'"	19 hours ago	Exited (0) 6 hours ago
eacd70596e23 adoring_yonath	nginx	"nginx -g 'daemon ...'"	19 hours ago	Exited (0) 19 hours ago
cffb4456e9c4 i2tch	ubuntu	"/bin/bash"	20 hours ago	Exited (0) 20 hours ago

```
root@debian11:~# docker exec -it centos3 bash
```

```
[root@57e92a8b25d7 /]# 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=0.146 ms
64 bytes from alias (172.17.0.4): icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from alias (172.17.0.4): icmp_seq=3 ttl=64 time=0.081 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 3079ms
rtt min/avg/max/mdev = 0.070/0.096/0.146/0.030 ms
```

```
[root@57e92a8b25d7 /]# 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 fc417b22a20d centos2
172.17.0.2     57e92a8b25d7

[root@57e92a8b25d7 /]# exit
exit
root@debian11:~#

root@debian11:~# docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' centos3
172.17.0.2
```

Notez cependant qu le lien est unidirectionnel :

```
root@debian11:~# docker exec -it centos2 bash

[root@fc417b22a20d /]# ping centos3
ping: centos3: Name or service not known

[root@fc417b22a20d /]# 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=0.097 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.125 ms
^C
--- 172.17.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2029ms
rtt min/avg/max/mdev = 0.082/0.101/0.125/0.019 ms
```

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@fc417b22a20d /]# exit
exit

root@debian11:~# 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@debian11:~# mkdir ~/wordpress && cd ~/wordpress
```

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

```
root@debian11:~/wordpress# docker run -e MYSQL_ROOT_PASSWORD=fenestros -e MYSQL_DATABASE=wordpress --name
wordpressdb -v "$PWD/database":/var/lib/mysql -d mysql:latest
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
e9f2695d7e5b: Pull complete
80c6055edb33: Pull complete
c646ab461d8b: Pull complete
012006c6a591: Pull complete
929d5fa34b95: Pull complete
17e0243877fa: Pull complete
1850b459cd2f: Pull complete
8dceaed53baf: Pull complete
197b834ealcd: Pull complete
8df78c25b227: Pull complete
Digest: sha256:ceb98918916bd5261b3e9866ac8271d75d276b8a4db56f1dc190770342a77a9b
```

```
Status: Downloaded newer image for mysql:latest
db3732939266ed8a112857db9c970ca39571785e62db74175bda9be5a0f9d726
```

Vérifiez que le conteneur fonctionne :

```
root@debian11:~/wordpress# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
db3732939266   mysql:latest   "docker-entrypoint.s..." About a minute ago Up 52 seconds 3306/tcp, 33060/tcp
wordpressdb
57e92a8b25d7   centos         "/bin/bash"             12 minutes ago Up 12 minutes
centos3
fc417b22a20d   centos         "/bin/bash"             25 minutes ago Up 25 minutes
centos2
cb2875ab1059   centos         "/bin/bash"             26 minutes ago Up 25 minutes
centos1
2126924504d8   centos         "/bin/bash"             29 minutes ago Up 29 minutes
resotest
```

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

```
root@debian11:~/wordpress# docker run -e WORDPRESS_DB_USER=root -e WORDPRESS_DB_PASSWORD=fenestros --name
wordpress --link wordpressdb:mysql -p 10.0.2.46:80:80 -v "$PWD/html":/var/www/html -d wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
1f7ce2fa46ab: Already exists
48824c101c6a: Pull complete
249ff3a7bbe6: Pull complete
aa5d47f22b64: Pull complete
851cb5d3b62c: Pull complete
090f07e09d3e: Pull complete
74f97600920f: Pull complete
f48a9f994636: Pull complete
108b4c091efa: Pull complete
```

```

94f753607622: Pull complete
5d0ec11ef45d: Pull complete
87757e6fac28: Pull complete
899a04597fc2: Pull complete
44506e60b7c1: Pull complete
305ecc1d68f5: Pull complete
a4e6cb47406c: Pull complete
8d4e2943ab66: Pull complete
cab275157cee: Pull complete
b12b496c1035: Pull complete
5bc81c9fd938: Pull complete
e737031fb816: Pull complete
Digest: sha256:3a2a8b925c86967a43027ec3ba146e1859de1fa0f0f535dd9b40f4d39f8b9caa
Status: Downloaded newer image for wordpress:latest
63fec083f4d6bb6a17563d9c6b4aefce2430abea6a2172997038c8f6edabab78

```

Vérifiez que le conteneur fonctionne :

```

root@debian11:~/wordpress# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
63fec083f4d6   wordpress     "docker-entrypoint.s..." About a minute ago Up About a minute
10.0.2.46:80->80/tcp   wordpress
db3732939266   mysql:latest  "docker-entrypoint.s..." 6 minutes ago  Up 6 minutes  3306/tcp,
33060/tcp      wordpressdb
57e92a8b25d7   centos        "/bin/bash"             18 minutes ago Up 18 minutes
centos3
fc417b22a20d   centos        "/bin/bash"             31 minutes ago Up 31 minutes
centos2
cb2875ab1059   centos        "/bin/bash"             31 minutes ago Up 31 minutes
centos1
2126924504d8   centos        "/bin/bash"             35 minutes ago Up 35 minutes
resotest

```

Vérifiez que le Wordpress fonctionne :

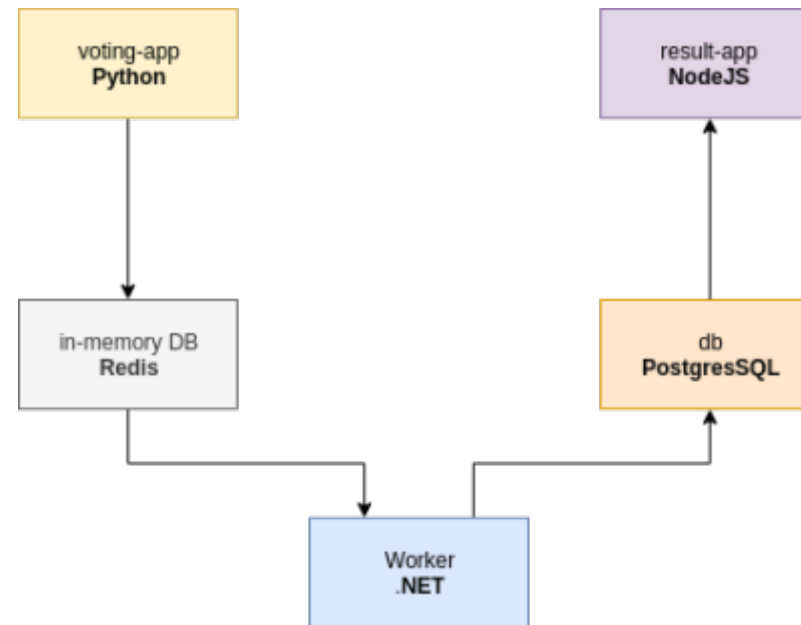
```
root@debian11:~/wordpress# lynx --dump http://10.0.2.46
WordPress
Select a default language [English (United States)_____]

Continue
root@debian11:~/wordpress#

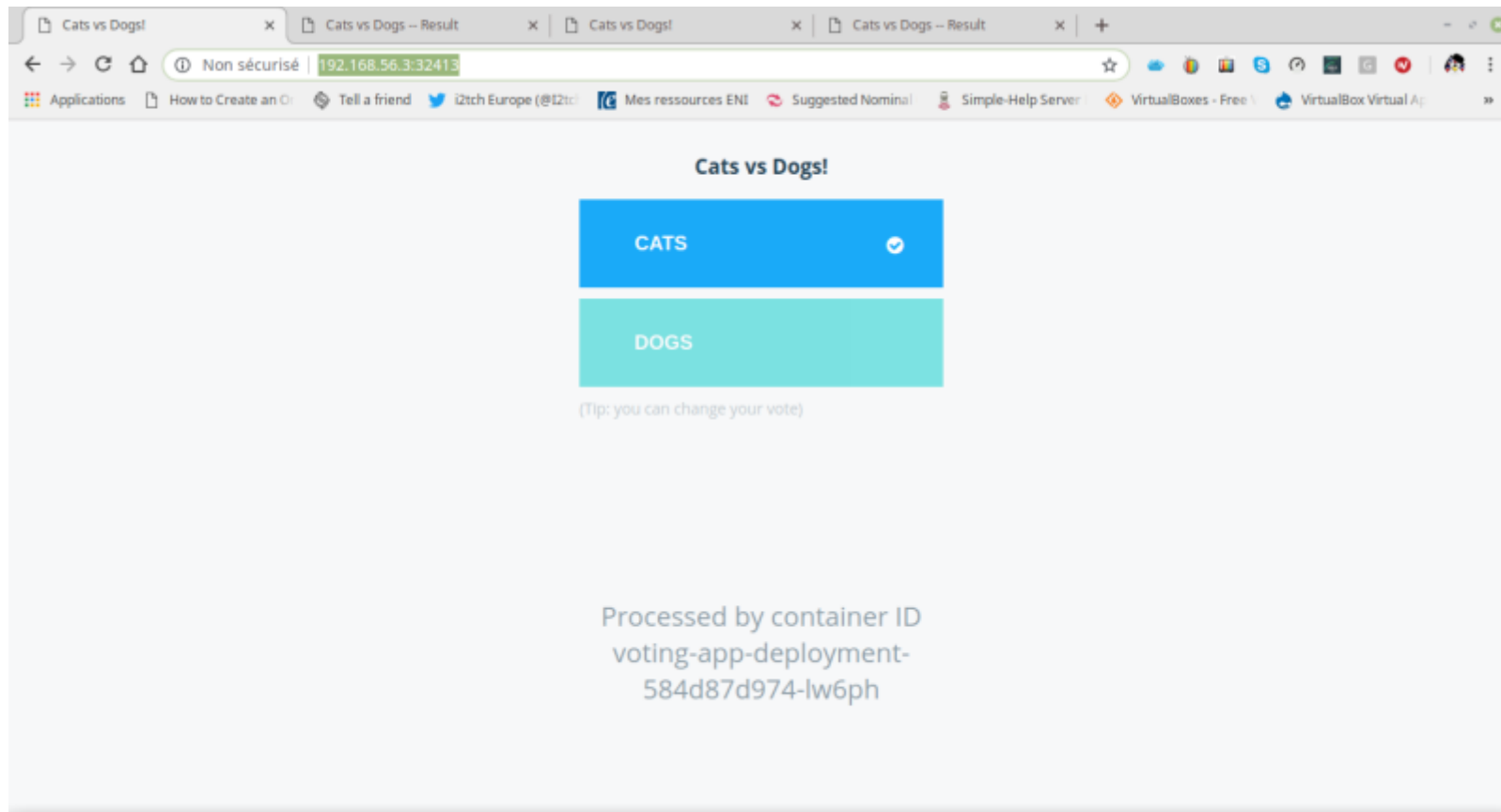
root@debian11:~/wordpress# cd -
/root
root@debian11:~#
```

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**, :

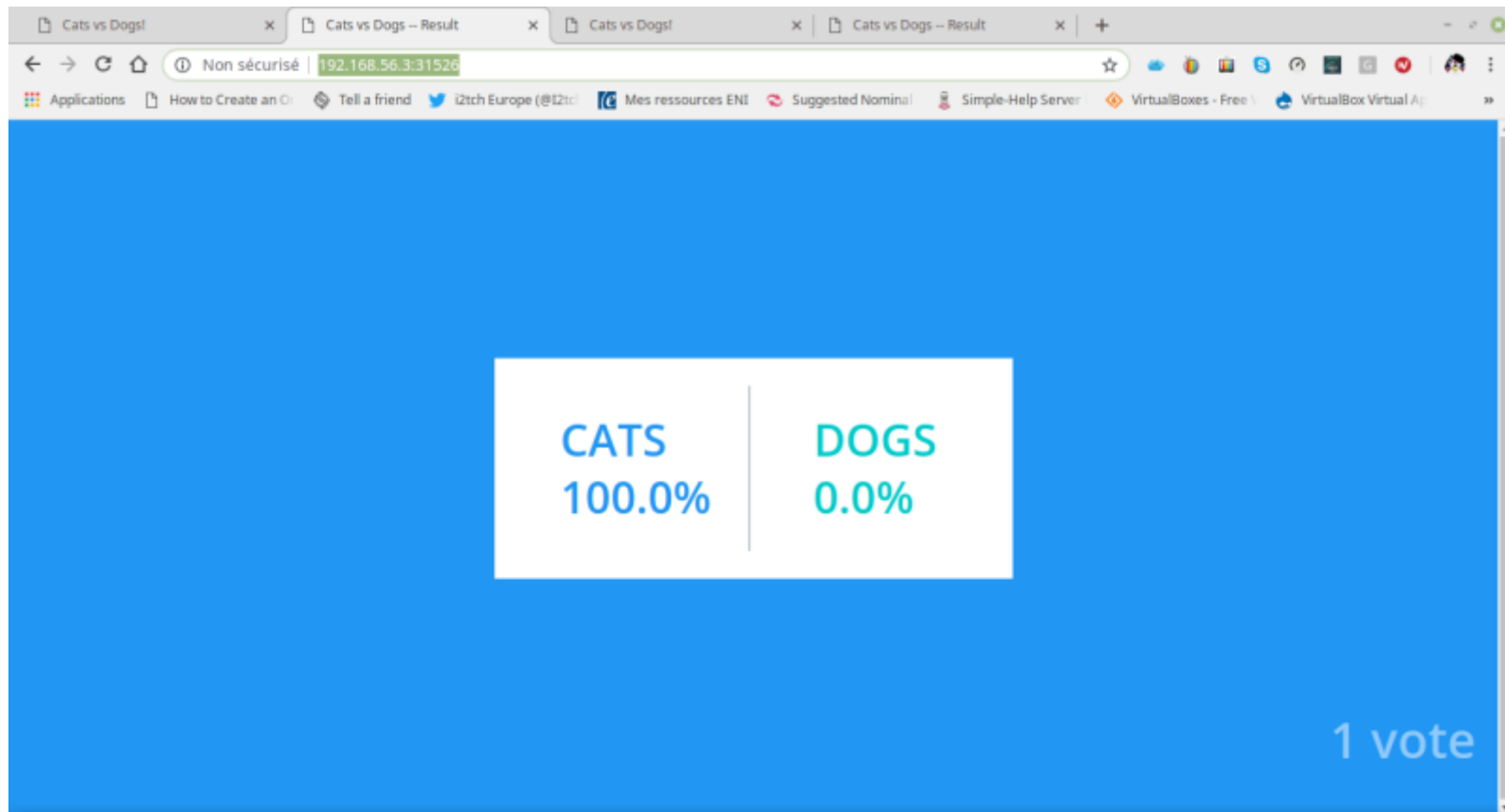


Dans cette application le conteneur **voting-app** permet de voter pour des **chats** ou des **chiens**. Cette application tourne sous Python et fournit une interface HTML :



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 :

```
root@debian11:~# docker run -d --name=redis redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
1f7ce2fa46ab: Already exists
4827e9d1e197: Pull complete
5845062cfda9: Pull complete
44d659adcf8b: Pull complete
b6962d83313d: Pull complete
5d29cf86ecab: Pull complete
```

```
4f4fb700ef54: Pull complete
3a2d9f90268c: Pull complete
Digest: sha256:396b0f027ba2f33bf385771a621b58c1fd834fd2c522c35c98fd24fc17863c2f
Status: Downloaded newer image for redis:latest
227554e3e4c198272cbf21dd468e7cf78d4a75ad5ed000a9df364aa98726bc86
```

```
root@debian11:~# docker run -d --name=db -e POSTGRES_PASSWORD=postgres -e POSTGRES_USER=postgres postgres:9.4
Unable to find image 'postgres:9.4' locally
9.4: Pulling from library/postgres
619014d83c02: Pull complete
7ec0fe6664f6: Pull complete
9ca7ba8f7764: Pull complete
9e1155d037e2: Pull complete
febcbf7f8870: Pull complete
8c78c79412b5: Pull complete
5a35744405c5: Pull complete
27717922e067: Pull complete
36f0c5255550: Pull complete
dbf0a396f422: Pull complete
ec4c06ea33e5: Pull complete
e8dd33eba6d1: Pull complete
51c81b3b2c20: Pull complete
2a03dd76f5d7: Pull complete
Digest: sha256:42a7a6a647a602efa9592edd1f56359800d079b93fa52c5d92244c58ac4a2ab9
Status: Downloaded newer image for postgres:9.4
5083545dcbf88ed9d1e605d306fe8dba86df1c130fcc843e7fba30eadd524545
```

```
root@debian11:~# docker run -d --name=vote -p 5000:80 --link redis:redis dockersamples/examplevotingapp_vote
Unable to find image 'dockersamples/examplevotingapp_vote:latest' locally
latest: Pulling from dockersamples/examplevotingapp_vote
a378f10b3218: Pull complete
c11bdfacfd25: Pull complete
64fc9a66a5d8: Pull complete
5146634606ba: Pull complete
```

```
479ce1f6823a: Pull complete
070425b38bdc: Pull complete
ce42fc94cbff: Pull complete
6bad37ec452b: Pull complete
edf50a17349a: Pull complete
db9bdfb7847f: Pull complete
Digest: sha256:797919beacc239d80f6c568e170ad4be0a6afd0ff0567e89d45f1dc3350b87f7
Status: Downloaded newer image for dockersamples/examplevotingapp_vote:latest
81e6fcb9f6920c048b3062e3da8e7e48b0475e5de3059ff3e5e63cbf73cb5fe6
```

```
root@debian11:~# docker run -d --name=result -p 5001:80 --link db:db dockersamples/examplevotingapp_result
Unable to find image 'dockersamples/examplevotingapp_result:latest' locally
latest: Pulling from dockersamples/examplevotingapp_result
a378f10b3218: Already exists
bc194d4002b7: Pull complete
231a505b2fbc: Pull complete
71731700a241: Pull complete
9c2ee871f3d2: Pull complete
a5ec303d8450: Pull complete
0548d3f3cdbc: Pull complete
c33ac9356c9f: Pull complete
495a50ede288: Pull complete
66140bd7f458: Pull complete
4d77129208cd: Pull complete
Digest: sha256:0b8fe15d93c08b9b90ad2eba02af526c1bee8bc9fab162a6b93b3186aa0a5faf
Status: Downloaded newer image for dockersamples/examplevotingapp_result:latest
33a264a36bdc63ba7c0a4e3412e437d20357b1142a02e26b6f6ccfb4aaab6cf2
```

```
root@debian11:~# docker run -d --name=worker --link db:db --link redis:redis
dockersamples/examplevotingapp_worker
Unable to find image 'dockersamples/examplevotingapp_worker:latest' locally
latest: Pulling from dockersamples/examplevotingapp_worker
e67fdae35593: Pull complete
0ab66724116f: Pull complete
```

```

14ccddebblbc: Pull complete
5e265b51b431: Pull complete
9ac34f7bda15: Pull complete
17081859cc14: Pull complete
Digest: sha256:bfa42cb2a0200cef7d384635225ca670f08c063341fc401bd27bae67ba6afc04
Status: Downloaded newer image for dockersamples/examplevotingapp_worker:latest
cf27f30654d2c527f30c1ed4b80a517ab589dc1579c30af7bd4e53eba746354a

```

```
root@debian11:~# docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
cf27f30654d2	dockersamples/examplevotingapp_worker		"dotnet Worker.dll"	7 seconds ago	Up 3 seconds
33a264a36bdc	dockersamples/examplevotingapp_result		"/usr/bin/tini -- no..."	31 seconds ago	Up 26
81e6fcb9f692	dockersamples/examplevotingapp_vote		"unicorn app:app -b..."	55 seconds ago	Up 50
5083545dcbf8	postgres:9.4	db	"docker-entrypoint.s..."	About a minute ago	Up About a
227554e3e4c1	redis	redis	"docker-entrypoint.s..."	About a minute ago	Up About a
63fec083f4d6	wordpress		"docker-entrypoint.s..."	43 minutes ago	Up 42
db3732939266	mysql:latest		"docker-entrypoint.s..."	48 minutes ago	Up 48
57e92a8b25d7	centos	centos3	"/bin/bash"	59 minutes ago	Up 59
fc417b22a20d	centos	centos2	"/bin/bash"	About an hour ago	Up About an
cb2875ab1059	centos	centos1	"/bin/bash"	About an hour ago	Up About an
2126924504d8	centos	resotest	"/bin/bash"	About an hour ago	Up About an

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@debian11:~# docker logs mongo2
root@5bfbf0306ad7:/# ip a
bash: ip: command not found
root@5bfbf0306ad7:/# ip addr
bash: ip: command not found
root@5bfbf0306ad7:/# exit
exit
```

3.2 - Les Processus

Consultez les processus d'un conteneur :

```
root@debian11:~# docker top centos3
```

UID	PID	PPID	C	STIME	TTY
TIME	CMD				
root	818263	818243	0	15:49	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@debian11:~# docker stats centos3
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT   MEM %     NET I/O       BLOCK I/O    PIDS
57e92a8b25d7   centos3   0.00%     880KiB / 15.62GiB    0.01%     4.72kB / 854B   0B / 4.1kB   1
^C
root@debian11:~#
```