Marquer comme terminé

author: - François Rioult lang: fr title: Bases de données avancées

subtitle: Sauvegarde d'un volume Docker

Les *volumes* de Docker permettent de faire des liaisons dynamiques entre le système de fichier de l'hôte (la VDI) et celui du container. Ainsi, lorsque le container écrit sur son système de fichiers, c'est en fait sur le système de fichiers de l'hôte que l'écriture a lieu.

Ce mécanismes de volumes permet de rendre persistantes des modifications du système de fichier du container. C'est notamment le cas lorsque le container fait tourner une base de données : il accède à son système de fichier local. Si les fichiers créés par la base de données sont sur un volume, les écritures ont en fait lieu sur le système de fichier de l'hôte.

La sauvegarde des volumes sur l'hôte permet donc d'assurer la pérennité des données du container.

Définition d'un volume sur un répertoire local

Lors de l'exécution d'un container, on indiquera la présence d'un volume, caractérisé par l'association entre un chemin sur le système de fichier local et un chemin dans le container\ :

- si on lance directement le container avec `docker run...`, on utilise l'option `-v` pour préciser l'association\ :

```
docker run -it -v ./backup:/backup-dir --rm bash:4.4
```

- ici, on lance un container nommé `testso` avec l'image `bash`, le volume fait la liaison entre le répertoire `./backup` de l'hôte et le répertoire `/backup-dir` du container

- si on utilise une pile de container, on précise le volume comme suit\ :

```
    volumes:
      - ./backup:/backup-dir
```

N.B. L'utilisation d'un volume local pour stocker la base de données est vivement déconseillée sur la VDI. Il est préférable de se tourner vers les solutions décrites ci-dessous (volume externe ou local à une pile). En effet, lorsqu'un container écrit des données dans un volume, il le fait en tant que `root`, éventuellement sans droit de lecture pour *les autres*. Sur le système de fichiers de l'hôte, les fichiers appartiendront également à `root`.

Si votre volume local est dans un sous-dossier de `~/Documents`, le système de fichier Samba qui gère ces dossiers refusera la création des fichiers du volume.

Si votre volume est dans `~`, les fichiers créés appartiendront à `root` et vous ne pourrez pas les relire / archiver.

Définition d'un volume externe

Docker permet de créer des volumes indépendamment d'un container\ :

```
$ docker volume create postgresdata
postgresdata
$ docker volume ls
local     postgresdata
$ docker volume inspect postgresdata
[
    {
        "CreatedAt": "2023-10-05T10:56:30+02:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/postgresdata/_data",
        "Name": "postgresdata",
        "Options": null,
        "Scope": "local"
    }
]
```

On constate que Docker gère le volume dans le point de montage `/var/lib/docker/volumes/postgresdata/_data`.

?

On pourra utiliser le volume externe\ :

- lors du lancement d'un container\ :

```
docker run -it -v docker-postgres_postgresdata:/var/lib/postgresql/data --rm bash:4.4
```

Docker comprend que le volume est externe car le chemin sur le système de fichier de l'hôte ne commence ni par `.` ni par `/` (ce n'est ni un chemin relatif ni absolu, c'est donc un volume externe).

- dans une pile de container :

```
services:
  postgres:
    ...
    volumes:
      - postgresdata:/var/lib/postgresql/data
    ...

volumes:
  postgresdata:
    external: true
```

Notez que dans la définition de la pile, le volume doit être déclaré dans la section adéquate.

Enfin, on peut détruire un volume externe par :

```
$ docker volume rm postgresdata
```

Définition d'un volume dans une pile de containers

Les volumes ne sont pas nécessairement externes, mais peuvent être locaux à une pile de container\ :

```
services:
  postgres:
    ...
    volumes:
      - postgresdata:/var/lib/postgresql/data
    ...

volumes:
  postgresdata:
```

La seule différence avec la version externe est que l'indication `external: true` n'est plus présente.

Le volume est alors nommé `<répertoire de la pile>_postgresdata`\ :

```
$ docker volume ls
DRIVER    VOLUME NAME
local     86c2419de77b79270ae9d6a15844111530ac823258a86ebd3637abbcf6bbdbe5
local     bf41994596b071536e1997267c3d8ee7f19bcacbd700e3661291997a75c74d33
local     docker-postgres_postgresdata
```

Sauvegarde d'un volume

Pour sauvegarder un volume, il faut\ :

- instancier un container avec ce volume et y générer un fichier d'archive que l'on souhaite récupérer en local\ : il faut donc un autre volume pour récupérer le fichier d'archive. Une image de `bash` suffira pour ces opérations

```
# créer un répertoire de backup
mkdir backup
# exécuter une image de bash avec deux volumes : l'un est postgresdata,
# l'autre sert au backup. L'option --rm permet de détruire le container
# lorsque sa tâche est terminée
docker run --rm -v docker-postgres_postgresdata:/var/lib/postgresql/data -v ./backup:/backup-dir bash:4.4 -c "cd /var/lib/
postgresql/data && tar cvf /backup-dir/postgres.tar ."
```

On dispose donc dans le répertoire `./backup` d'un fichier d'archive `postgres.tar`.

N.B. Il est prudent de préalablement d'éteindre la pile de containers dont on sauvegarde le volume. Cependant, le backup peut être effectué pendant que la pile tourne. Il faudra cependant stopper le container (pour éviter que le gestionnaire de données effectue des opérations)\ :

```
docker stop postgres-dev
```

On peut ensuite procéder au backup, puis relancer le container\ :

?

```
docker start postgres-dev
```

Restauration d'un volume

La restauration est l'opération inverse de la sauvegarde\ : on instancie un container `bash` avec deux volumes, celui à restaurer et un volume permettant d'accéder à l'archive\ :

```
docker run --rm -v docker-postgres_postgresdata:/var/lib/postgresql/data -v ./backup:/backup-dir bash:4.4 -c "cd /var/lib/
postgresql/data && tar xvf /backup-dir/postgres.tar"
```

On a supposé ici que le volume `docker-postgres_postgresdata` n'existait pas.

Contrôle d'un volume

Pour contrôler ce qui se passe dans un volume, il est possible d'instancier une image `bash` intégrant le volume à tester\ :

```
$ docker run -it -v docker-postgres_postgresdata:/var/lib/postgresql/data --rm bash:4.4
bash-4.4# ls
bin    dev    home   media  opt    root   sbin   sys    usr
data   etc    lib    mnt    proc   run    srv    tmp    var
bash-4.4# cd data/db/
bash-4.4# ls
...
```

Récapitulatif sauvegarde / restauration d'un volume

On considère disposer d'un volume `graphile_datadir` :

```
# démarrer la pile de container
docker-compose -f stack.yml up -d
# insérer les données
docker exec -i postgres-dev psql -U postgres -d rioultf < commerce.sql
# éteindre la pile de container
docker-compose -f stack.yml down

# créer un répertoire de backup
mkdir backup
# exécuter une image de bash avec deux volumes : l'un est `graphile_datadir`,
# l'autre sert au backup. L'option --rm permet de détruire le container
# lorsque sa tâche est terminée
docker run --rm -v graphile_datadir:/var/lib/postgresql/data -v ./backup:/backup-dir bash:4.4 -c "cd /var/lib/postgresql/data
&& tar cvf /backup-dir/postgres.tar ."

# détruire le volume
docker volume rm graphile_datadir

# défaire le tar dans le volume
docker run --rm -v graphile_datadir:/var/lib/postgresql/data -v ./backup:/backup-dir bash:4.4 -c "cd /var/lib/postgresql/data
&& tar xvf /backup-dir/postgres.tar"

# lancer un système minimal avec les volumes
docker run -it -v graphile_datadir:/var/lib/postgresql/data --rm bash:4.4
```

Modifié le: lundi 20 novembre 2023, 17:51

◄ Séance Docker + Graphile

Choisir un élément

Aller à...

Bréviaire Docker ►