Virtualisation de données: Jouer avec Oracle 12c sur les containers de Docker

Franck Pachot, dbi-services
Traduit par Pierre Ochsenbein & Vincent Matthey

Cette année sera consacré à la virtualisation de données. Nous ne pouvons pas continuer à multiplier le stockage de nos base de données en fonction de chaque environnement (exemple : production, développement, test) que nous avons à mettre à disposition. Nous avons besoin de plus d'agilité dans cet approvisionnement. Il y a d'excellents outils (comme Delphix, sur lequel j'ai écrit dans la newsletter précédente) et des alternatives, sur lesquelles j'écrirai prochainement (Snapshot Standby Snap Clone, etc). Docker est un outil possédant une grande agilité, il permet de séparer les applications de l'infrastructure. Il s'appuie sur le système de fichier copy-on-write et les containers linux pour apporter les 'data containers'. Alors pourquoi ne pas l'utiliser pour nos bases de données?

Attention : Ce n'est pas quelque chose que vous pouvez mettre en place dès maintenant en production. Je ne décris pas une solution mature mais seulement un prototype, l'idée est de vous montrer ce qui est faisable avec Docker, et de vous aider à faire vos propres tests dans votre infrastructure.
Merci de me tenir informé de vos essais (@FranckPachot ou Franck.Pachot@dbi-services.com).

Je ne vais pas décrire ce qu'est Docker, il y a le site web pour cela: www.docker.com. Je vais vous présenter uniquement un exemple: j'ai installé Oracle dans un container Docker et j'y ai créé une base de données standby qui sera rafraichie continuellement à partir d'une base de données source. Et je peux créer de nouvelles bases de données virtuelles très légères: un nouveau container Docker partagera le stockage pour les données qui ne sont pas modifiées.

Installation Docker

Vous pouvez installer Docker directement sur Linux, ou utiliser la VM ‘boot2docker’ avec VirtualBox sous Windows.

J’ai téléchargé boot2docker et j’ai installé Virtual Box sur mon ordinateur portable Windows:

```
boot2docker init
```

Ceci crée une VM Virtualbox et je partage un répertoire /Users afin de partager le répertoire Windows depuis Docker. Je vais utiliser ce répertoire afin de mettre les fichiers d'installation d’Oracle:

```
VBoxManage sharedfolder add boot2docker-vm -hostpath
"F:\SOFTWARE" -name /Users -automount
```

Ensuite, je peux démarrer la VM et me connecter en ssh:

```
boot2docker start
Waiting for VM and Docker daemon to start...
.....................................
Started.
Docker client does not run on Windows for now. Please use
"C:\Program Files\Boot2Docker for Windows\boot2docker.exe" ssh
to SSH into the VM instead.
```

Connectons-nous en ssh dessus pour continuer:

```
boot2docker ssh
                        ##         .
                  ## ## ##        ==
               ## ## ## ##       ===
           /""""""""""""""""\___/ ===
      ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
           \______ o          __/
             \    \        __/
              \____\______/
 _                 _   ____     _            _
| |__   ___   ___ | |_|___ \ __| | ___   ___| | _____ _ __
| '_ \ / _ \ / _ \| __| __) / _` |/ _ \ / __| |/ / _ \ '__|
| |_) | (_) | (_) | |_ / __/ (_| | (_) | (__|   <  __/ |
|_.__/ \___/ \___/ \__|_____\__,_|\___/ \___|_|\_\___|_|
Boot2Docker version 1.4.1, build master : 86f7ec8 - Tue Dec 16 23:11:29 UTC 2014
Docker version 1.4.1, build 5bc2ff8
docker@boot2docker:~$
```

J’ai besoin de connaitre mon adresse IP:

```
docker@boot2docker:~$ ip addr
…
    inet 192.168.59.104/24 brd 192.168.59.255 scope global
…
```

L’adresse IP est: 192.168.59.104

Installation d’Oracle

Avec Docker, vous ne commencez jamais à partir de zéro. Vous commencez à partir d’un container que vous téléchargez (pull) à partir du dépôt. Dans le dépôt public, vous ne pouvez pas trouver une image avec Oracle déjà installé. La raison est liée aux licences : la gestion des licences interdit la distribution d'une VM avec le logiciel déjà installé. Il y a une seule exception à cela, c’est Oracle XE. Et vous pouvez trouver des images Docker avec Oracle XE dans le référentiel. Mais je veux utiliser les fonctionnalités complètes d’Oracle 12c et je vais configurer la standby avec DataGuard, donc je dois l’installer moi-même.

Ok, je vais commencer à partir d’une image qui est prête pour Oracle. C’est la ‘breed85 / oracle-12c’, vous pouvez l’obtenir avec cette commande:

```
docker@boot2docker:~$ docker pull breed85/oracle-12c
```

Les voici:

```
docker@boot2docker:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED       VIRTUAL SIZE
breed85/oracle-12c latest      9d3b9aab053b 11 weeks ago 1.728 GB
breed85/oracle-12c preinstall 9d3b9aab053b 11 weeks ago 1.728 GB
```

Tout est là sauf oracle. Vous pouvez télécharger le logiciel sur le site d’oracle (http://www.oracle.com/technetwork/database/enterprise-edition/downloads/) et obtenir les deux fichiers zip : linuxamd64_12102_database_1of2.zip et linuxamd64_12102_database_2of2.zip. Vous devez les placer dans un répertoire que vous pouvez monter dans le container Docker.
Comme j’utilise boot2docker, je partage le répertoire ‘/Users’ sous Windows dans la VM boot2docker.

Je lance maintenant l’image ‘breed85/oracle-12c’ avec le tag ‘12101’ et je partage le boot2docker /Users en ‘/User’ dans le container:

```
docker@boot2docker:~$ docker run --name 12102 -ti -v
 /Users:/Users breed85/oracle-12c:preinstall /bin/bash
```

Si vous n'avez pas récupéré l'image 'breed85/oracle-12c', téléchargez là.
Ensuite, je dois installer les binaires 'unzip' ainsi que 'preinstall' pour être sûr de les avoirs.

```
[root@2cd790c5a069 /]# yum -y install oracle-rdbms-server-
 12cR1-preinstall unzip
```

Je fais un peu d'espace dans le yum cache et je crée le répertoire /oracle où je vais mettre tout ce qui doit être accessible par oracle:

```
[root@2cd790c5a069 /]# mkdir -p /oracle/install
[root@2cd790c5a069 /]# chown -R oracle:dba /oracle
```

Décompression des fichiers:

```
[root@2cd790c5a069 /]# unzip
 /Users/linuxamd64_12102_database_1of2.zip -d
 /oracle/install/linuxamd64_12102_database
[root@2cd790c5a069 /]# unzip
 /Users/linuxamd64_12102_database_2of2.zip -d
 /oracle/install/linuxamd64_12102_database
```

Je suis prêt à installer. Je génère un 'response file' en mode silencieux:

```
[root@2cd790c5a069 /]# cat > /oracle/db_install.rsp <<-'CAT'
oracle.install.responseFileVersion=/oracle/install/rspfmt_dbin
stall_response_schema_v12.1.0
oracle.install.option=INSTALL_DB_SWONLY
ORACLE_HOSTNAME=
UNIX_GROUP_NAME=oinstall
INVENTORY_LOCATION=/oracle/oraInventory
SELECTED_LANGUAGES=en
ORACLE_HOME=/oracle/product/12102
ORACLE_BASE=/oracle
oracle.install.db.InstallEdition=EE
oracle.install.db.DBA_GROUP=dba
oracle.install.db.OPER_GROUP=dba
oracle.install.db.BACKUPDBA_GROUP=dba
oracle.install.db.DGDBA_GROUP=dba
oracle.install.db.KMDBA_GROUP=dba
CAT
```

Et je l'installe en tant qu'utilisateur oracle:

```
[root@2cd790c5a069 /]# su - oracle -c
 "/oracle/install/linuxamd64_12102_database/database/runInstal
 ler -silent -ignoreSysPrereqs -ignorePrereq -silent -noconfig
 -responseFile /oracle/db_install.rsp"
```

J'ignore beaucoup de prérequis car je ne suis de toute façon pas sur une configuration supportée.

À ce moment-là, je dois patienter 5 minutes pour l'installation, puis exécuter le script root.sh

```
[root@2cd790c5a069 /]# /oracle/oraInventory/orainstRoot.sh
[root@2cd790c5a069 /]# /oracle/product/12102/root.sh
```

Je n'ai pas beaucoup d'espace dans ma VM Docker, donc je supprime les fichiers décompressés dont je n'ai plus besoin ainsi que d'autres dont je n'aurais pas besoin (Ai-je déjà dit de ne pas le faire en production?)

```
[root@2cd790c5a069 /]# rm -rf /var/cache/yum/x86_64
 /oracle/install/linuxamd64_12102_database
 /oracle/product/12102/inventory/backup/*
 /oracle/product/12102/assistants/dbca/templates
```

Bien sûr, j'aurais pu créer la VM Docker un peu plus grande, mais c'est juste un test:

```
[root@2cd790c5a069 /]# exit
```

Ok, c'était la partie complète de la préparation, téléchargement, décompression et installation. Je l'enregistre dans une nouvelle image.

```
docker@boot2docker:~$ docker commit 12102 breed85/oracle-
 12c:12102
```

Je pourrais vous aider et juste mettre cette image dans le référentiel docker, mais je suis désolé, je ne veux pas payer les licences Entreprise Edition pour les CPUs qui sont derrière ;)

Voici mes images:

```
docker@boot2docker:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        VIRTUAL
 SIZE
breed85/oracle-12c 12102       9cc040ef1d6e 5 minutes ago 6.264 GB
breed85/oracle-12c latest      9d3b9aab053b 11 weeks ago  1.728 GB
breed85/oracle-12c preinstall 9d3b9aab053b 11 weeks ago  1.728 GB
```

Et je peux supprimer le container que je viens de valider:

```
docker@boot2docker:~$ docker ps -as
CONTAINER ID IMAGE                     COMMAND     CREATED        STATUS
 PORTS NAMES SIZE
2cd790c5a069 breed85/oracle-12c:latest "/bin/bash" 18 minutes ago Exited
 (0) 5 minutes ago       12102 4.536 GB

docker@boot2docker:~$ docker rm 12102
```

Dupliquer la base de données

Comme mon but est de simuler la virtualisation d'une base de données existante, je la dupliquerai avec RMAN et afin d'en faire une physical standby.
Ma base de données source est créée simplement avec dbca et toutes les options par défaut (Sauf que je la créée comme une non-CDB pour garder notre exemple simple)

Voici la base de données source:

```
RMAN> report schema;

using target database control file instead of recovery catalog
Report of database schema for database with db_unique_name DEMO111

List of Permanent Datafiles
===========================
File Size(MB) Tablespace           RB segs Datafile Name
---- -------- -------------------- ------- ------------------------
1    780      SYSTEM               YES     /u02/…/o1_mf_system_bf67nzw0_.dbf
3    600      SYSAUX               NO      /u02/…/o1_mf_sysaux_bf67mwls_.dbf
4    255      UNDOTBS1             YES     /u02/…/o1_mf_undotbs1_bf67pgdj_.dbf
6    5        USERS                NO      /u02/…/o1_mf_users_bf67pf5q_.dbf

List of Temporary Files
=======================
File Size(MB) Tablespace           Maxsize(MB) Tempfile Name
---- -------- -------------------- ----------- --------------------
1    60       TEMP                 32767       /u02/…/o1_mf_temp_bf67qnwz_.tmp
```

Ma base de données source est appelée DEMO111 et est sur l'hôte 192.168.78.111 et le mot de passe SYS est "oracle" et j'ai paramétré dg_broker_start=true comme je créerai la standby avec le broker de DataGuard. Ma VM docker a l'adresse IP 192.168.59.104.

Donc si vous faites un copier/coller, vous devriez changer ces valeurs.

J'exécute un shell dans un container depuis l'image que j'ai créé auparavant. Je l'appelle 'DEMO111_SBY' comme hostname. Je n'ai plus besoin du répertoire /Users. Mais maintenant, je vais rediriger le port 1521 vers le port 9000. Cela signifie que quand je démarre le listener sur le port 1521 dans le container (écoute de //DEMO111_SBY:1521 dans le container) je serai en mesure d'accéder depuis l'extérieur avec l'adresse IP //192.168.59.104:9000

Voici ma commande à exécuter (-h est pour le hostname, -ti correspond au terminal interactif):

```
docker@boot2docker:~$ docker run --name "DEMO111_SBY" -ti -h
 "DEMO111_SBY" -p 9000:1521 breed85/oracle-12c:12102 su -
 oracle
```

J’ai configuré mon environnement. Le nom de l’instance sera DEMO111:

```
[oracle@DEMO111_SBY ~]$ export
 ORACLE_HOME=/oracle/product/12102
[oracle@DEMO111_SBY ~]$ export ORACLE_SID=DEMO111
```

et j’ai créé un password file identique à la base de donnée primaire:

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/orapwd
 file=$ORACLE_HOME/dbs/orapwDEMO111 password=oracle
```

J’ai besoin que le listener soit configuré en tant que service statique. Gardons cela au plus simple:

```
[oracle@DEMO111_SBY ~]$ cat >
 $ORACLE_HOME/network/admin/listener.ora <<-CAT
LISTENER=(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=DEMO111_SBY
 )(PORT=1521)))
SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(GLOBAL_DBNAME=DEMO111_S
 BY_DGMGRL)(ORACLE_HOME=$ORACLE_HOME)(SID_NAME=DEMO111)))
CAT
```

Et je le démarre:

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/lsnrctl start
```

Voici le message de sortie où nous pouvons voir le point d’écoute et le service statique:

```
…
Listener Parameter File   /oracle/product/12102/network/admin/listener.ora
Listener Log File
 /oracle/diag/tnslsnr/DEMO111_SBY/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=DEMO111_SBY)(PORT=1521)))
Services Summary...
Service "DEMO111_SBY_DGMGRL" has 1 instance(s).
  Instance "DEMO111", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

J’utiliserai OMF et vais donc créer les répertoires:

```
[oracle@DEMO111_SBY ~]$ mkdir /oracle/oradata
 /oracle/recovery_area
```

J’ai besoin d’un init.ora pour démarrer mon instance:

```
[oracle@DEMO111_SBY ~]$ cat > /tmp/init.ora <<-CAT
db_name='DEMO111'
compatible=12.1.0.2.
db_unique_name='DEMO111_SBY'
db_create_file_dest='/oracle/oradata'
db_recovery_file_dest_size='1G'
db_recovery_file_dest='/oracle/recovery_area'
dg_broker_start=true
CAT
```

Créez alors le spfile et démarrez l’instance:

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/sqlplus / as sysdba
 <<-'SQL'
create spfile from pfile='/tmp/init.ora';
startup nomount;
SQL
```

Testons la connexion au réseau:

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/tnsping
 "//192.168.78.111:1521/DEMO111"
…
Attempting to contact
 (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=DEMO111))(ADDRESS=(PROTOCOL=TCP)(
 HOST=192.168.78.111)(PORT=1521)))
OK (0 msec)

[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/tnsping
 "//192.168.59.104:9000/DEMO111_SBY_DGMGRL"
…
Attempting to contact
 (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=DEMO111_SBY_DGMGRL))(ADDRESS=(PRO
 TOCOL=TCP)(HOST=192.168.59.104)(PORT=9000)))
OK (0 msec)
```

Tout est parfait. Comme je peux me connecter à la source ainsi qu’à la cible, faisons un duplicate from active avec rman. Une bonne pratique est de déclarer la chaîne de connexion dans le fichier tnsnames.ora mais faisons le rapidement avec la chaîne ezconnect (qui fonctionne dans la mesure où elle est inférieur à 64 caractères).

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/rman

RMAN> connect target sys/oracle@//192.168.78.111:1521/DEMO111;
connected to target database: DEMO111 (DBID=2551031863)
```

```
RMAN> connect auxiliary
 sys/oracle@//192.168.59.104:9000/DEMO111_SBY_DGMGRL;
connected to auxiliary database: DEMO111 (not mounted)

RMAN> duplicate database for standby from active database;
Starting Duplicate…
…
Finished Duplicate Db…
```

Configurer la standby

J'utilise le broker DataGuard pour configurer rapidement la standby:

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/dgmgrl <<-DG
connect sys/oracle@"//192.168.78.111:1521/DEMO111"
create configuration 'DEMO111' as primary database is
 'DEMO111' connect identifier is
 '192.168.78.111:1521/DEMO111';
add database 'DEMO111_SBY' as connect identifier is
 '//192.168.59.104:9000/DEMO111_SBY_DGMGRL ';
enable configuration;
edit database 'DEMO111_SBY' set state=apply-on;
DG
```

N'oubliez pas que le but est juste d'avoir une standby qui est synchronisée. Nous ne basculerons jamais vers la base du container Docker ! Voici l'écran de sortie:

Ok, il suffit d'attendre un peu et il est synchronisé:

```
DGMGRL> connect /
Connected as SYSDBA.
DGMGRL> show database "DEMO111_SBY"
Database - DEMO111_SBY

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 20 seconds ago)
  Apply Lag:          0 seconds (computed 20 seconds ago)
  Average Apply Rate: 30.00 KByte/s
  Real Time Query:    OFF
  Instance(s):
    DEMO111
```

Et c'est tout. Afin de conserver cette configuration, nous pouvons quitter le shell interactif et enregistrer cette image:

```
docker@boot2docker:~$ docker commit DEMO111_SBY
 breed85/oracle-12c:DEMO111_SBY
```

Supprimer le container:

```
docker@boot2docker:~$ docker rm DEMO111_SBY
```

Et vérifier la nouvelle image qui a 2.13GB supplémentaire pour notre base de données:

```
docker@boot2docker:~$ docker images
REPOSITORY        TAG         IMAGE ID     CREATED          VIRTUAL SIZE
breed85/oracle-12c DEMO111_SBY 31068a2689fd About a minute ago 8.394 GB
breed85/oracle-12c 12102       9cc040ef1d6e About an hour ago  6.264 GB
breed85/oracle-12c latest      9d3b9aab053b 11 weeks ago       1.728 GB
breed85/oracle-12c preinstall  9d3b9aab053b 11 weeks ago       1.728 GB
```

Rafraîchir la standby

Notre but est d'avoir une standby rafraîchit en permanence, nous allons donc exécuter un container que nous allons laisser tourner. Je peux le mettre en arrière-plan, mais pour les tests, je vais le laisser tourner dans une fenêtre.

C'est la même ligne de commande sauf que j'appelle le container 'refresh'

```
docker@boot2docker:~$ docker run --name "refresh" -ti -h
 "DEMO111_SBY" -p 9000:1521 breed85/oracle-12c:DEMO111_SBY su
 - oracle
```

Et je lance le minimum requis, démarrer le listener:

```
[oracle@DEMO111_SBY ~]$ export
 ORACLE_HOME=/oracle/product/12102
[oracle@DEMO111_SBY ~]$ export ORACLE_SID=DEMO111
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/lsnrctl start
```

Et l'instance:

```
[oracle@DEMO111_SBY ~]$ $ORACLE_HOME/bin/sqlplus / as sysdba
 startup nomount;
 alter database mount standby database;
```

Je démarre docker dans une autre fenêtre:

```
cd "C:\Program Files\Boot2Docker for Windows"
boot2docker ssh
docker@boot2docker:~$
```

J'ai les images suivantes:

```
docker@boot2docker:~$ docker images
REPOSITORY          TAG          IMAGE ID     CREATED         VIRTUAL SIZE
breed85/oracle-12c DEMO111_SBY 31068a2689fd 41 minutes ago 8.394 GB
breed85/oracle-12c 12102        9cc040ef1d6e 2 hours ago    6.264 GB
breed85/oracle-12c latest       9d3b9aab053b 11 weeks ago   1.728 GB
breed85/oracle-12c preinstall   9d3b9aab053b 11 weeks ago   1.728 GB
```

Les images étant construites les unes à partir des autres, je n'aurai pas besoin physiquement de la totalité de l'espace virtuel affiché ci-dessus.

Le 'refresh' container est en cours de fonctionnement.

```
docker@boot2docker:~$ docker ps -as
…IMAGE           COMMAND       PORTS                     NAMES    SIZE
…12c:DEMO111_SBY "su - oracle" 0.0.0.0:9000->1521/tcp   refresh 1.793 GB

docker@boot2docker:~$ docker ps -as
CONTAINER ID IMAGE                          COMMAND       CREATED
 STATUS        PORTS                    NAMES    SIZE
9f2d6e5333bb breed85/oracle-12c:DEMO111_SBY "su - oracle" 20 minutes ago Up
 20 minutes 0.0.0.0:9000->1521/tcp refresh 1.808 GB
```

Nous voyons la taille (qui est la taille ajoutée à l'image de base) et le port de redirection. Le statut est en cours.

Maintenant, je sauvegarde cet état dans une autre image:

```
docker@boot2docker:~$ docker commit refresh breed85/oracle-
 12c:vdb1
```

J'ai donc une nouvelle image:

```
docker@boot2docker:~$ docker images
REPOSITORY          TAG          IMAGE ID     CREATED         VIRTUAL SIZE
breed85/oracle-12c vdb1         be0ba1450a98 42 seconds ago 10.2 GB
breed85/oracle-12c DEMO111_SBY 31068a2689fd 46 minutes ago 8.394 GB
breed85/oracle-12c 12102        9cc040ef1d6e 2 hours ago    6.264 GB
breed85/oracle-12c latest       9d3b9aab053b 11 weeks ago   1.728 GB
breed85/oracle-12c preinstall   9d3b9aab053b 11 weeks ago   1.728 GB
```

Vous voyez que j'ai maintenant un serveur de base de données virtuel supplémentaire qui fait 10.2GB (Système + logiciel Oracle + fichiers de données + fichiers d'archive) mais je n'ai besoin que de 1.79GB pour le stocker – ce qui correspond à la taille de la couche 'vbd1'.

Ce sont des tailles virtuelles. La taille physique réelle est visible sur le système de fichiers AUFS qui est utilisé par docker:

```
docker@boot2docker:~$ df -h /mnt/sda1/var/lib/docker/aufs
Filesystem                Size      Used Available Use% Mounted on
/dev/sda1                18.2G     11.3G      5.9G  66% /mnt/sda1
```

Quel est le coût d'approvisionnement d'une autre base de données virtuelle?

```
docker@boot2docker:~$ docker commit refresh breed85/oracle-12c:vdb2
Filesystem                Size      Used Available Use% Mounted on
/dev/sda1                18.2G     13.0G      4.2G  76% /mnt/sda1

docker@boot2docker:~$ docker images
REPOSITORY          TAG         IMAGE ID     CREATED             VIRTUAL SIZE
breed85/oracle-12c vdb2         61d372cc903d About a minute ago 10.2 GB
breed85/oracle-12c vdb1         be0ba1450a98 6 minutes ago      10.2 GB
breed85/oracle-12c DEMO111_SBY 31068a2689fd 51 minutes ago     8.394 GB
breed85/oracle-12c 12102        9cc040ef1d6e 2 hours ago        6.264 GB
breed85/oracle-12c latest       9d3b9aab053b 11 weeks ago       1.728 GB
breed85/oracle-12c preinstall   9d3b9aab053b 11 weeks ago       1.728 GB
```

La nouvelle base de données utilise physiquement 1.7GB mais nous pouvons voir une copie virtuelle de 10.2GB.

Ouverture de la base de données virtuelle

Alors, de quoi avons-nous besoin pour utiliser la nouvelle base de données virtuelle ? Il suffit juste de lancer l'image dans un nouveau container et de rediriger le port 1521 vers un autre:

```
docker@boot2docker:~$ docker run --name "vdb1" -ti --rm -h
 "DEMO111_vdb1" -h "vdb1" -p 9001:1521 breed85/oracle-12c:vdb1
 su - oracle
```

J'ai ajouté l'option '–rm' de sorte que le container est supprimé quand je quitte le shell. Je le fais quand je veux juste l'utiliser pour une courte période de temps.

Je vais être capable d'y accéder en utilisant le port redirigé 9001 de sorte que la chaîne de connexion sera: //192.169.59.104:9001/DEMO111_SBY mais pour le moment, je dois démarrer le listener. Notez que j'ai supprimé le listener.ora qui était configuré pour un autre nom d'hôte:

```
[oracle@vdb1 ~]$ export ORACLE_HOME=/oracle/product/12102
[oracle@vdb1 ~]$ export ORACLE_SID=DEMO111
[oracle@vdb1 ~]$ rm $ORACLE_HOME/network/admin/listener.ora
[oracle@vdb1 ~]$ $ORACLE_HOME/bin/lsnrctl start
```

Puis démarrer l'instance:

```
[oracle@vdb1 ~]$ $ORACLE_HOME/bin/sqlplus / as sysdba
 startup nomount;
 alter database mount standby database;
```

Et activer la standby pour qu'elle devienne la primaire afin de l'ouvrir en lecture/écriture, ce qui est appelé un failover.

```
[oracle@vdb1 ~]$ $ORACLE_HOME/bin/dgmgrl /
  failover to "DEMO111_SBY" immediate;

…
Performing failover NOW, please wait...
Failover succeeded, new primary is "DEMO111_SBY"
```

Bien sûr, ce serait une bonne idée d'isoler le container lors du démarrage afin d'être sûr que la nouvelle primaire n'essaye pas de communiquer avec les bases primaire et standby actuelles. Nous pouvons changer le nom de la base de données mais le container peut fournir l'isolation dont on a besoin. Je ne veux qu'il y ait qu'une seule chose qui puisse passer à travers: la redirection de port 9001 -> 1521.

Le résultat est que j'ai maintenant deux containers qui tournent:

```
IMAGE            CREATED  STATUS  PORTS              NAMES    SIZE
12c:vdb1         9 minut  Up 9 m  …:9001->1521/tcp   vdb1     2.128 GB
12c:DEMO111_SBY About a   Up Abo  …:9000->1521/tcp   refresh  1.813 GB
```

J'ai fait quelques mises à jour sur 'vdb1', c'est pourquoi la taille a augmenté et un 'refresh' applique toujours les redo depuis la primaire.

Je peux exécuter autant de base de données virtuelle que je veux. Elles sont gérées comme des containers Docker. Et si vous voulez retrouver les informations concernant les ressources, les voici:

```
Mem: 1889632K used, 166604K free, 0K shrd, 376K buff, 376K cached
CPU:  0.1% usr  0.0% sys  0.0% nic 99.8% idle  0.0% io  0.0% irq  0.0% sirq
Load average: 0.20 0.32 0.41 2/411 8023
…
 5960  5908 54321    S     662m 32.8   5  0.0 {ora_pmon_demo11} ora_pmon_DEMO111
…
 7691  7631 54321    S     662m 32.8   1  0.0 {ora_pmon_demo11} ora_pmon_DEMO111
…
```

Toutes les instances sont en train de tourner (vous voyez de multiples pmon). Elles tournent toutes sur le même serveur mais isolées dans des containers.

Se connecter à une base de données virtuelle

Je peux me connecter à ma vdb1 depuis n'importe quel client qui a accès à la VM docker à travers le réseau:

```
SQL> connect system/oracle@//192.168.59.104:9001/DEMO111_SBY
```

Et vérifier depuis quel host il est exécuté:

```
SQL> select host_name from v$instance;
HOST_NAME
----------------------------------------
DEMO111_vdb1
```

Qui est le nom d'hôte virtuel passé à la commande 'run' à travers l'argument '-h'.

Conclusion

Quel est le but d'écrire un article sur quelque chose que l'on ne peut pas faire en production? Je voulais montrer les concepts de la virtualisation des données et j'aime apprendre par la pratique. Toutes les technologies sous-jacentes (système de fichier copy-on-write, virtualisation, isolation) sont là et vous pouvez les utiliser avec des outils simples tels que Docker pour pratiquer sur votre PC portable.

Essayez-le. Téléchargez Docker. Téléchargez les fichiers zip d'Oracle. Et faites la même chose, ce qui est probablement aussi simple que de copier-coller les commandes et de remplacer seulement les adresses IP.

Bien sûr, pour votre production, vous devrez aller vers des logiciels robustes et supportés et de nombreux fournisseurs ont des bonnes solutions pour la virtualisation des données, les snapshots et les clones. Mais je suis sûr qu'une fois que vous aurez pratiqué avec ce do-it-yourself qui encore une fois, ne concerne pas une solution officiellement supportée, alors vous aurez de meilleures bases pour choisir et comparer les solutions commerciales.

Contact:
Franck Pachot
franck.pachot@dbi-services.com

Franck Pachot est consultant senior chez dbi services en Suisse. Il a plus de 20 ans d'expérience dans les bases de données Oracle, dans la plupart des domaines, depuis le développement, la modélisation des données, la performance, l'administration et la formation. Oracle Certified Master et Oracle ACE, il essaye de partager ses connaissances dans les forums, via des publications et des présentations.