



## Lab Exercise: Data Persistence With Docker Volumes

### Preamble

This lab exercise makes use of Docker volumes in order to demonstrate that data persists beyond the life of a container that uses the volume. Additionally, the lab exercise demonstrates how a directory from the Docker host can be mounted into the container, in order for the container to write files to.

We will use a named volume to create a database with a container running MariaDB. Some sample data will be used to populate the database. We will use a container specifically to create and populate the database, test that it has succeeded via a temporary container, and then stop and remove the database container. With the database safely stored in the named volume, we'll start a new container to serve the database using the volume, before we perform a backup of the database in another container, with a host directory mounted inside it.

### Step 1 – download the sample database

The example database we will use for our MariaDB container is the classicmodels [sample database](#) used in the examples provided on the excellent [MySQLTUTORIALS](#) website. Download the sample database, and then extract it into its own directory somewhere in your HOME directory.

```
$ unzip ~/Downloads/mysqlsampledatabase1.zip -d ~/tmp/classicmodels
```

This will leave you with the file `mysqlsampledatabase.sql` in the specified directory, which contains a series of SQL statements for creating and populating the classicmodels database.

### Step 2 – create a network and a named volume

Create a network on which to run the database container.

```
$ docker network create bridge0
```

In order to ensure that the data in the database persists between invocations, it will need persistent storage, and we'll need to create a named volume for the purpose.

```
$ docker volume create --name data
```

List the volumes available using the `docker volume ls` sub-command:

---

```
$ docker volume ls
DRIVER          VOLUME NAME
local           data
```

### Step 3 – launch the MariaDB database container

The MariaDB container needs some very specific configuration options to be supplied for its invocation; let's explain each one:

```
-v data:/var/lib/mysql: the library/mariadb image uses the directory
/var/lib/mysql for its data files, and this configuration mounts the named volume we
just created onto the container's data file directory
-v $HOME/tmp/classicmodels:/docker-entrypoint-initdb.d: the image also
provides for database creation on container start-up, executing any .sh or .sql files
located in /docker-entrypoint-initdb.d, and this configuration option mounts the
directory where we saved mysqlsampledatabase.sql onto /docker-entrypoint-
initdb.d
-e MYSQL_ROOT_PASSWORD=pw: the database needs to be provided with a root password
when first created, which takes the form of an environmental variable (this could be
provided using a file along with the --env-file configuration option)
```

The other configuration options are not specific for the mariadb image, and have been covered previously. Start the container:

```
$ docker run -d -e MYSQL_ROOT_PASSWORD=glasnost --name db --net bridge0 \
> -v data:/var/lib/mysql \
> -v $HOME/tmp/classicmodels:/docker-entrypoint-initdb.d mariadb
```

The container will initialise and create the classicmodels database in detached mode. Use the docker logs command (with the -f, --follow configuration option), to check what's being written to stdout/stderr, and look out for a message indicating the server is listening for connections.

### Step 4 – launch a temporary client container to query the database

Now launch a client container using the same image in order to verify the database is running, and to query it's contents. In order to connect, you need to use the mysql client command, supplying the appropriate configuration options in order to allow the client to connect to the database (the MariaDB image exposes port 3306 for connections):

```
$ docker run -it --rm --net bridge0 mariadb mysql -hdb -P3306 -ppw
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.12-MariaDB-1~jessie mariadb.org binary distribution
```

---

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>

Use the `show databases;` command to list the databases, and then the command `use classicmodels;` in order to set the current database:

```
MariaDB [(none)]> show databases;
```

```
+-----+
| Database          |
+-----+
| classicmodels     |
| information_schema |
| mysql             |
| performance_schema |
+-----+
4 rows in set (0.00 sec)
```

```
MariaDB [(none)]> use classicmodels;
```

Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A

Database changed

```
MariaDB [classicmodels]>
```

Perform a simple query on the database:

```
MariaDB [(none)]> select lastname, firstname, jobtitle from
employees where lastname='Patterson';
```

```
+-----+-----+-----+
| lastname | firstname | jobtitle          |
+-----+-----+-----+
| Patterson | Mary      | VP Sales          |
| Patterson | William   | Sales Manager (APAC) |
| Patterson | Steve     | Sales Rep         |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

Now that you've established that the database is running, exit from the SQL client session by typing `exit`.

### Step 5 – stop and remove the MariaDB server container

Because we're using a named volume to store the data files for the `classicmodels` database, we can stop and then remove the MariaDB container, and the data will remain intact in the volume:

---

```
$ docker rm $(docker stop db)
db
```

#### Step 6 – start a new MariaDB server using the data files located in the named volume

As the database has already been initialised, the Docker CLI command to start a new MariaDB instance is slightly different. We can omit the environment variable for the root user, as this has already been set, and we can also omit the initialisation script.

```
$ docker run -d --name db --net bridge0 -v data:/var/lib/mysql mariadb
```

The database will be started in the new version of the MariaDB container. Repeat step 4 to verify that the `classicmodels` database is being served correctly. Exit from SQL client when finished.

#### Step 7 – perform a logical backup of the database

Finally, we can use a mounted directory from the host in order to perform a logical backup of the database, again from a client container started for the purpose. This time the host directory `$HOME/db/archive` is mounted on to `/archive` in the container, and the `mysqldump` command is executed with its output being redirected into a file located in `/archive`.

```
$ docker run -it --rm -v $HOME/db/archive:/archive --net bridge0 mariadb \
> sh -c "mysqldump -hdb -P3306 -uroot -ppw --single-transaction \
> classicmodels > /archive/db_$(date -u +%y-%m-%d_%H-%M-%S').sql"
```

Check for the backup in the `$HOME/db/archive` directory.

Remove the MariaDB server container, the named volume called `data`, and the network `bridge0`.