2.5.6.1 Basic Steps for MySQL Server Deployment with Docker

Warning

The MySQL Docker images maintained by the MySQL team are built specifically for Linux platforms. Other platforms are not supported, and users using these MySQL Docker images on them are doing so at their own risk. See the discussion here for some known limitations for running these containers on non-Linux operating systems.

- Downloading a MySQL Server Docker Image
- Starting a MySQL Server Instance
- Connecting to MySQL Server from within the Container
- Container Shell Access
- Stopping and Deleting a MySQL Container
- Upgrading a MySQL Server Container
- More Topics on Deploying MySQL Server with Docker

Downloading a MySQL Server Docker Image

Downloading the server image in a separate step is not strictly necessary; however, performing this step before you create your Docker container ensures your local image is up to date. To download the MySQL Community Edition image, run this command:

```
docker pull mysql/mysql-server:tag
```

The ***tag*** is the label for the image version you want to pull (for example, `5.5`, `5.6`, `5.7`, `8.0`, or `latest`). If **:*tag*** is omitted, the `latest` label is used, and the image for the latest GA version of MySQL Community Server is downloaded. Refer to the list of tags for available versions on the mysql/mysql-server page in the Docker Hub.

You can list downloaded Docker images with this command:

```
shell> docker images
REPOSITORY           TAG                 IMAGE ID            CREATED             SIZE
mysql/mysql-server   latest              3157d7f55f8d        4 weeks ago         241MB
```

To download the MySQL Enterprise Edition image from the My Oracle Support website, sign in to your Oracle account, download from **Patches and Updates** the `tar.zip` file for the Docker image (`mysql-commercial-`

version`_linux_x86_64_docker_tar.zip`), unzip it to obtain the tarball inside (`mysql-enterprise-server-`***version***`.tar`), and then load the image by running this command:

```
docker load -i mysql-enterprise-server-version.tar
```

Starting a MySQL Server Instance

To start a new Docker container for a MySQL Server, use the following command:

```
docker run --name=container_name -d image_name:tag
```

The image name can be obtained using the **docker images** command, as explained in Downloading a MySQL Server Docker Image. The `--name` option, for supplying a custom name for your server container, is optional; if no container name is supplied, a random one is generated.

For example, to start a new Docker container for the MySQL Community Server, use this command:

```
docker run --name=mysql1 -d mysql/mysql-server:8.0
```

To start a new Docker container for the MySQL Enterprise Server with a Docker image downloaded from My Oracle Support, use this command:

```
docker run --name=mysql1 -d mysql/enterprise-server:8.0
```

If the Docker image of the specified name and tag has not been downloaded by an earlier **docker pull** or **docker run** command, the image is now downloaded. Initialization for the container begins, and the container appears in the list of running containers when you run the **docker ps** command. For example:

```
shell> docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED             STATUS
a24888f0d6f4   mysql/mysql-server   "/entrypoint.sh my..."   14 seconds ago      Up 13
```

The container initialization might take some time. When the server is ready for use, the `STATUS` of the container in the output of the **docker ps** command changes from `(health: starting)` to `(healthy)`.

The `-d` option used in the **docker run** command above makes the container run in the background. Use this command to monitor the output from the container:

```
docker logs mysql1
```

Once initialization is finished, the command's output is going to contain the random password generated for the root user; check the password with, for example, this command:

```
shell> docker logs mysql1 2>&1 | grep GENERATED
GENERATED ROOT PASSWORD: Axegh3kAJyDLaRuBemecis&EShOs
```

Connecting to MySQL Server from within the Container

Once the server is ready, you can run the **mysql** client within the MySQL Server container you just started, and connect it to the MySQL Server. Use the **docker exec -it** command to start a **mysql** client inside the Docker container you have started, like the following:

```
docker exec -it mysql1 mysql -uroot -p
```

When asked, enter the generated root password (see the last step in Starting a MySQL Server Instance above on how to find the password). Because the `MYSQL_ONETIME_PASSWORD` option is true by default, after you have connected a **mysql** client to the server, you must reset the server root password by issuing this statement:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'password';
```

Substitute ***`password`*** with the password of your choice. Once the password is reset, the server is ready for use.

Container Shell Access

To have shell access to your MySQL Server container, use the **docker exec -it** command to start a bash shell inside the container:

```
shell> docker exec -it mysql1 bash
bash-4.2#
```

You can then run Linux commands inside the container. For example, to view contents in the server's data directory inside the container, use this command:

```
bash-4.2# ls /var/lib/mysql
auto.cnf    ca.pem        client-key.pem  ib_logfile0  ibdata1  mysql       mysql.sock.l
ca-key.pem  client-cert.pem  ib_buffer_pool  ib_logfile1  ibtmp1   mysql.sock  performa
```

Stopping and Deleting a MySQL Container

To stop the MySQL Server container we have created, use this command:

```
docker stop mysql1
```

docker stop sends a SIGTERM signal to the **mysqld** process, so that the server is shut down gracefully.

Also notice that when the main process of a container (**mysqld** in the case of a MySQL Server container) is stopped, the Docker container stops automatically.

To start the MySQL Server container again:

```
docker start mysql1
```

To stop and start again the MySQL Server container with a single command:

```
docker restart mysql1
```

To delete the MySQL container, stop it first, and then use the **docker rm** command:

```
docker stop mysql1
```

```
docker rm mysql1
```

If you want the Docker volume for the server's data directory to be deleted at the same time, add the `-v` option to the **docker rm** command.

Upgrading a MySQL Server Container

> **Important**
>
> - Before performing any upgrade to MySQL, follow carefully the instructions in Section 2.11, “Upgrading MySQL”. Among other instructions discussed there, it is especially important to back up your database before the upgrade.
> - The instructions in this section require that the server's data and configuration have been persisted on the host. See Persisting Data and Configuration Changes for details.

Follow these steps to upgrade a Docker installation of MySQL 5.7 to 8.0:

- Stop the MySQL 5.7 server (container name is `mysql57` in this example):

  ```
  docker stop mysql57
  ```

- Download the MySQL 8.0 Server Docker image. See instructions in Downloading a MySQL Server Docker Image; make sure you use the right tag for MySQL 8.0.
- Start a new MySQL 8.0 Docker container (named `mysql80` in this example) with the old server data and configuration (with proper modifications if needed—see Section 2.11, “Upgrading MySQL”) that have been persisted on the host (by bind-mounting in this example). For the MySQL Community Server, run this command:

```
docker run --name=mysql80 \
   --mount type=bind,src=/path-on-host-machine/my.cnf,dst=/etc/my.cnf \
   --mount type=bind,src=/path-on-host-machine/datadir,dst=/var/lib/mysql \
   -d mysql/mysql-server:8.0
```

  If needed, adjust `mysql/mysql-server` to the correct repository name—for example, replace it with `mysql/enterprise-server` for MySQL Enterprise Edition images downloaded from My Oracle Support.
- Wait for the server to finish startup. You can check the status of the server using the **docker ps** command (see Starting a MySQL Server Instance for how to do that).
- *For MySQL 8.0.15 and earlier:* Run the mysql_upgrade utility in the MySQL 8.0 Server container (not required for MySQL 8.0.16 and later):

```
docker exec -it mysql80 mysql_upgrade -uroot -p
```

  When prompted, enter the root password for your old MySQL 5.7 Server.
- Finish the upgrade by restarting the MySQL 8.0 Server container:

```
docker restart mysql80
```

More Topics on Deploying MySQL Server with Docker

For more topics on deploying MySQL Server with Docker like server configuration, persisting data and configuration, server error log, and container environment variables, see Section 2.5.6.2, “More Topics on Deploying MySQL Server with Docker”.