Running the Database Container

The awaited database integration to our app.

We'll cover the following

In this lesson, we will add a database container to our app. We will use the MySQL server version 5.7 because it is the most used version of MySQL server.

Steps to add the database

- Pull MySQL server's official Docker Hub image
- Run the image with MySQL environment variables
- Install the DB connector for the flask
- Change the application code
- Run the Flask app with a link to the database container

We will study all the steps above in detail below.

MySQL image

Type `docker pull mysql/mysql-server:5.7` to pull the MySQL image. Once the image is downloaded, let's create a container from the image like this:

```
Venkateshs-MacBook-Air:Flask_app venkateshachintalwar$ docker run --name datab
ase mysql/mysql-server:5.7
[Entrypoint] MySQL Docker Image 5.7.29-1.1.15
[Entrypoint] No password option specified for new database.
[Entrypoint]   A random onetime password will be generated.
[Entrypoint] Initializing database
[Entrypoint] Database initialized
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skippi
```

```
ng it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skippi

ng it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping
 it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping
 it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipp
ing it.
[Entrypoint] GENERATED ROOT PASSWORD: eBycacYl4hyB%3f2EmiJUf@zzAno

[Entrypoint] ignoring /docker-entrypoint-initdb.d/*

[Entrypoint] Server shut down
[Entrypoint] Setting root user as expired. Password will need to be changed be
fore the database can be used.

[Entrypoint] MySQL init process done. Ready for start up.

[Entrypoint] Starting MySQL 5.7.29-1.1.15
```

This is the simplest command to run the MySQL container. Try to read the output line by line.

According to MySQL's official documentation, we can configure **MYSQL_ROOT_PASSWORD**, **MYSQL_USER**, **MYSQL_USER_PASSWORD**, **MYSQL_DATABASE** from the command line while starting the container using environment variables passed to a container.

So, let's try that. Open a new terminal window and type `docker stop database` and then `docker rm database` to remove the container.

Running with ENV variables

Type `docker run --name mysql -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=backend -e MYSQL_USER=testuser -e MYSQL_PASSWORD=admin123 mysql/mysql-server:5.7`.

This is a little long command, but let's break this down into chunks below.

The `--name` argument is used to define the container name. `-e` is for environment variables. Each variable is used by MySQL in the startup script.

- `MYSQL_ROOT_PASSWORD=root`: sets the root user's password to 'root'

- `MYSQL_DATABASE=backend`: creates the database named 'backend' while starting up
- `MYSQL_USER=testuser`: creates a non-root user to access the database
- `MYSQL_PASSWORD=admin123`: set the non-root user's password

After this command, you will get the output which is like:

```
$ docker run --name mysql -e MYSQL_ROOT_PASSWORD=root     -e MYSQL_DATABASE=ba
ckend -e MYSQL_USER=testuser     -e MYSQL_PASSWORD=admin123 mysql/mysql-server
:5.7[Entrypoint] MySQL Docker Image 5.7.29-1.1.15
[Entrypoint] Initializing database
[Entrypoint] Database initialized
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skippi
ng it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skippi
ng it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping
 it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping
 it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipp
ing it.

[Entrypoint] ignoring /docker-entrypoint-initdb.d/*

[Entrypoint] Server shut down

[Entrypoint] MySQL init process done. Ready for start up.

[Entrypoint] Starting MySQL 5.7.29-1.1.15
```

Until now, what we have been working with may not look like a regular MySQL DB. This is because, in non-Docker MySQL DB, we directly access the MySQL shell and not the installation part.

As soon as our DB container is ready, we can access the MySQL shell in the new command prompt/terminal by using the MySQL user and password we provided in the environment variables.

Type `docker exec -it mysql bash` followed by `mysql -u testuser -padmin123`.

```
s venkateshachintalwar$ docker exec -it mysql bash
bash-4.2# mysql -u testuser -padmin123
mysql: [Warning] Using a password on the command line interface can be insecur
e.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.7.29 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

You can perform different SQL queries here. Try to check if the database is created or not using `show databases` query.

However, we haven't created a table yet and there are no users we can validate our login mechanism with.

We have completed the first two steps to add the database to our project. Play around the container and try to connect to it from our app container and from the host machine. If you are not able to access the container from the host machine, refer to the architecture diagram provided in the last lesson.

In the next lesson, we modify our app code and connect to the DB container.