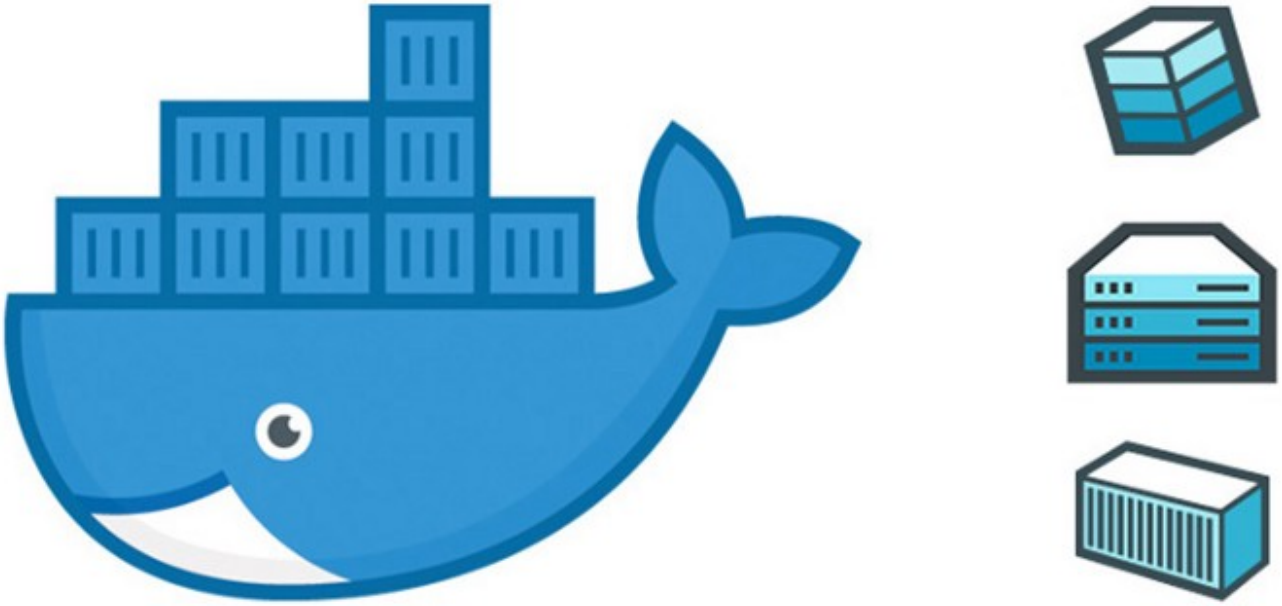
Setting Up a MySQL Database in Docker

Learn to set up and run Docker containers for databases

Ashutosh Karna Follow
Aug 8, 2019 · 4 min read ★



Docker offers many advantages for deploying and testing applications and databases that are an integral part of your application, so it's worth learning to set up and run Docker containers for databases.

In this article, we will focus on the following topics:

1. Create a Docker Compose YAML file for a MySQL Docker container.

2. Connect to the MySQL database, running on a container, using various methods.
3. Create and run multiple versions of MySQL in Docker containers.

Let's go through these one-by-one.

1. Create a Docker Compose YAML File for a MySQL Docker Container

Let's create a directory, `db-docker`, and then create a `docker-compose.yml` file in that directory:

```
mkdir db-docker
cd db-docker
touch docker-compose.yml
```

Basically, here, we will specify the services we are going to use and set up the environment variables related to those.

We will change this file multiple times throughout this article.

Add the following in the `docker-compose.yml` file we just created:

```yaml
version: '3'

services:

  mysql-development:
    image: mysql:8.0.17
    environment:
      MYSQL_ROOT_PASSWORD: helloworld
      MYSQL_DATABASE: testapp
    ports:
      - "3308:3306"
```

We specified the name of our MySQL container as `mysql-development` and the Docker image to be used is `mysql:8.0.17`. Where if don't specify the tag as `8.0.17`, it will take the latest one.

The next thing we need to specify is the environment variables, i.e. the user, password, and database. If you don't specify the user, by default it will be `root`.

We will use `helloworld` as password and `testapp` as database.

Another important thing is port mapping. `3308:3306` means that the MySQL running in the container at port `3306` is mapped to the localhost of the host machine at port `3308`. You can use a different port as well.

Now, after creating the .yml file, we need to run the following command in the same directory where the .yml file is located:

```
docker-compose up
```

This will pull the Docker image (if the image is not available locally, it will pull from Docker Hub) and then run the container.

We can check the status with:

```
docker-compose ps
```

```
→ db-docker docker-compose ps
           Name                         Command            State                 Ports
-------------------------------------------------------------------------------------------------
db-docker_mysql-development_1   docker-entrypoint.sh mysqld   Up      0.0.0.0:3308->3306/tcp,
                                                                      33060/tcp
```

This will show the name of the container, command, and state of the container, which shows, for example, that the container is running. It also shows port mapping.

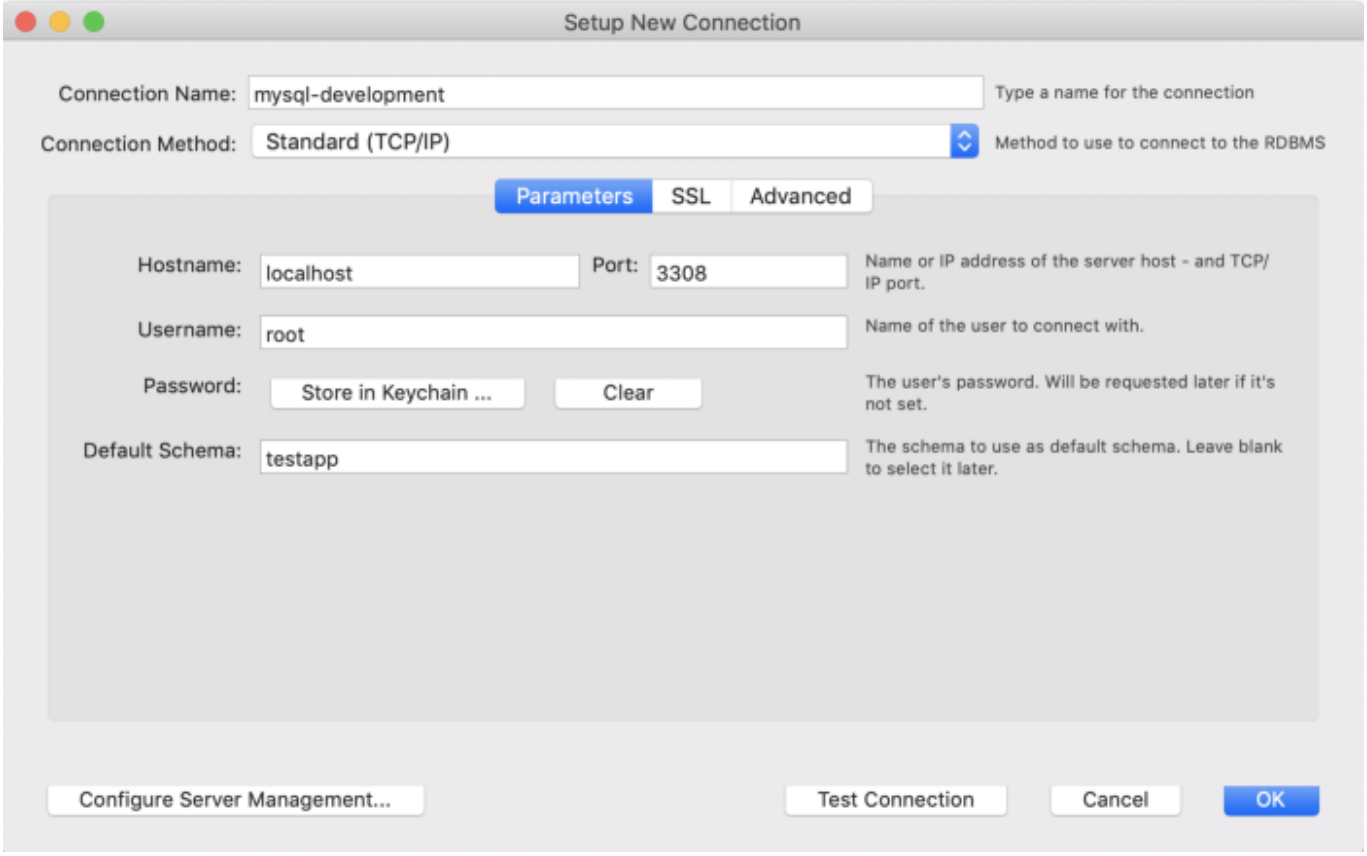
In the next step, we will connect to this MySQL container and run some commands.

2. Connect to the MySQL Database Running in a Container

We will discuss two methods to connect and run SQL commands on the MySQL running in a Docker container.

The first method is to use tools like MySQL Workbench (DataGrip also can be used).

As we now have a MySQL container running at our local machine's port `3308`, we can connect using the following configuration parameters:



The connection through the local machine's port `3308` was only possible due to port mapping.

If we want to connect to containerized MySQL, without mapping ports, i.e. from another application running on the same Docker network, we have to use tools like Adminer, which is our other method.

Adminer is a PHP-based web application for accessing databases.

Now, we will add another service for Adminer in our `docker-compose.yml` file. But, before we make changes here, we need to stop running the container and remove it with the following command:

```
docker-compose down
```

Let's add the following in our `docker-compose.yml` file:

```
version: '3'

services:

  mysql-development:
    image: mysql:8.0.17
    environment:
      MYSQL_ROOT_PASSWORD: helloworld
      MYSQL_DATABASE: testapp
    ports:
      - "3308:3306"

  admin:
    image: adminer
    ports:
      - "8080:8080"
```

Now, let's start the Docker containers again:

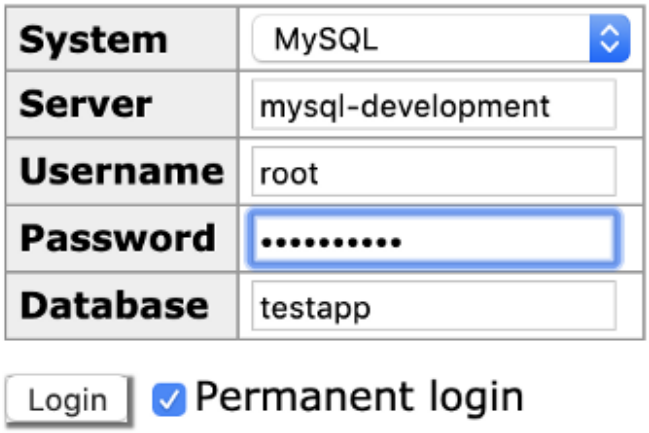
```
docker-compose up
```

After running this, the image for Adminer will be pulled and the container for both MySQL and Adminer will be started.

We can check this using `docker-compose ps`.

Now, we can go to our browser and go to `localhost:8080` for Adminer. As Adminer runs on the same Docker network as MySQL, it can access the MySQL container via port `3306` (or simply, by the container's name).

Note: We can't access the MySQL container through port `3308` in Adminer, as this will try to access port `3308` of the Docker Compose network, not our local machine's `3308` port.



We can also use the MySQL command-line interface with the following command:

```
docker-compose exec  mysql-development mysql -uroot -phelloworld
testapp
```

3. Create and Run Multiple Versions of MySQL in Docker Containers

If we have an application that uses some other version of MySQL, we can create a service for that as well and run it in the same Docker network.

For example, if we need MySQL version 5.7.27, we need to make the following changes to the `docker-compose.yml` file, stop running containers, and start again.

```
version: '3'

services:

  mysql-development:
    image: mysql:8.0.17
    environment:
      MYSQL_ROOT_PASSWORD: helloworld
      MYSQL_DATABASE: testapp
    ports:
      - "3308:3306"

  admin:
    image: adminer
    ports:
      - "8080:8080"

   mysql-old:
    image: mysql:5.7.27
    environment:
      MYSQL_ROOT_PASSWORD: helloworld
      MYSQL_DATABASE: coolapp
    ports:
      - "3309:3306"
```

If you want to know further about how to use data and configuration volumes and also how to check logs of mysql container, checkout my article on it: https://medium.com/@ashutosh_34428/volumes-and-logs-in-mysql-docker-61122f8c1d84

We have reached the end of the article. I hope you've learned something. Thanks for reading!

Get this newsletter newsletter.

MySQL Docker Docker Compose Database Programming

About Help Legal

Get the Medium app