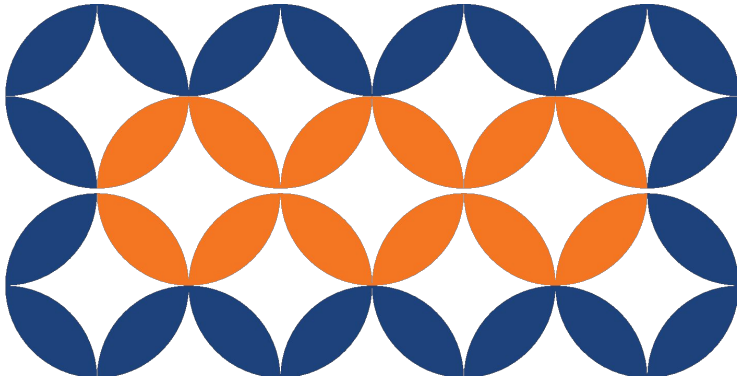
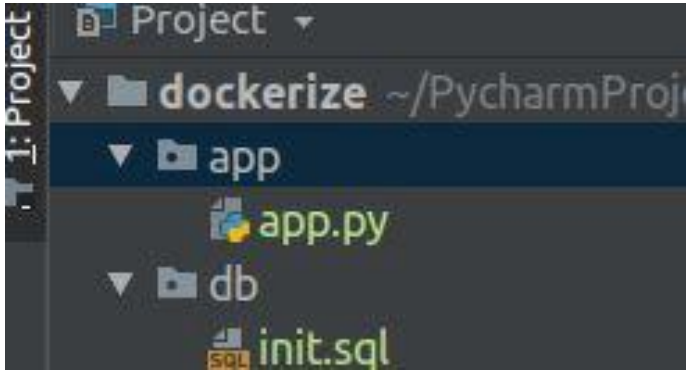
alterra



Dockerizing a Flask-Mysql app with docker-compose

alterra

Let's begin with the following project layout



app.py—contains the Flask app which connects to the database (db) and exposes one REST API endpoint

init.sql—an SQL script to initialize the database before the first time the app runs

alterra

Creating a Docker image for our app

We want to create a Docker image for our app, so we need to create a **Dockerfile** in the app directory.

```
# Use an official Python runtime as an image
FROM python:3.6

# The EXPOSE instruction indicates the ports on which a container # # will listen for conn
# Since Flask apps listen to port 5000  by default, we expose it
EXPOSE 5000

# Sets the working directory for following COPY and CMD instructions
# Notice we haven't created a directory by this name - this
# instruction creates a directory with this name if it doesn't exist
WORKDIR /app

# Install any needed packages specified in requirements.txt
COPY requirements.txt /app
RUN pip install -r requirements.txt

# Run app.py when the container launches
COPY app.py /app
CMD python app.py
```

alterra

Creating a docker-compose.yml

We are using two services, one is a container which exposes the REST API (app), and one contains the database (db).

```
1  version: "2"
2  services:
3    app:
4      build: ./app
5      links:
6        - db
7      ports:
8        - "5000:5000"
```

build: specifies the directory which contains the Dockerfile containing the instructions for building this service
links: links this service to another container. This will also allow us to use the name of the service
ports: mapping of <Host>:<Container> ports

alterra

Database (db) service container :

```
db:
  image: mysql:5.7
  ports:
    - "32000:3306"
  environment:
    MYSQL_ROOT_PASSWORD: root
  volumes:
    - ./db:/docker-entrypoint-initdb.d/:ro
```

image: using an existing image from a repository
ports: mapping port to container
environment: add environment variables. The specified variable is required for this image
volumes: the directory containing our init.sql script to the entry point for this container, which by the image's specification runs all .sql scripts in the given directory

alterra

Running the app

In order to run the our dockerized app, we will execute the following command from the terminal:
$docker-compose up

If everything went right, you will see the following line:
app_1 | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

We can find out that everything is running as expected by typing this url in a browser or using curl (http://0.0.0.0:5000/), and receiving the following response:
{"favorite_colors": [{"Lancelot": "blue"}, {"Galahad": "yellow"}]}

alterra

Conclusion

We have learned how to dockerize a simple Flask-MySQL using docker-compose. Now this app can be used without tiresome preconfiguration on every host with Docker and docker-compose.

Github source :
https://github.com/erysepulsa/dockerizing-flask-mysql