## **Docker Compose Tutorial: Flask Web Application with MySQL**

### **Introduction**

### Docker Compose is a powerful tool for defining and managing multi-container Docker applications. It allows you to use a YAML file to configure your application's services, networks, and volumes, simplifying the process of setting up complex environments for development, testing, and production.

### In this tutorial, we will create a simple Flask web application that connects to a MySQL database. The Flask app will demonstrate a successful connection to the MySQL database and display a confirmation message. This example will showcase the interaction between a web application and a database using Docker Compose.

### **Example Application Overview**

### In this example, we will set up two Docker containers:

### MySQL Container: This container will run the MySQL database server.

### Flask Web Application Container: This container will run a Flask web application that connects to the MySQL database and displays a confirmation message if the connection is successful.

### By the end of this tutorial, you will have a basic understanding of how to use Docker Compose to manage multi-container applications and how to configure a Flask web application to interact with a MySQL database.

### 

#### **Step 1: Set Up Your Project Directory**

Create a project directory and navigate into it:

mkdir docker-compose-webapp-check

cd docker-compose-webapp-check

#### **Step 2: Create the Flask Application**

1. Create a directory for the Flask app:  
   mkdir webapp  
   cd webapp
2. Create the Flask application file (app.py):

# webapp/app.py  
from flask import Flask  
import mysql.connector  
from mysql.connector import Error  
app = Flask(\_\_name\_\_)  
@app.route('/')  
def check\_db\_connection():

try:

connection = mysql.connector.connect(

host='mysql',

database='testdb',

user='user',

password='password'

)

if connection.is\_connected():

db\_info = connection.get\_server\_info()

cursor = connection.cursor()

cursor.execute("SELECT DATABASE();")

record = cursor.fetchone()

return f"Connected to MySQL Server version {db\_info}. Connected to database: {record}"

except Error as e:

return f"Error while connecting to MySQL: {e}"

finally:

if connection.is\_connected():

cursor.close()

connection.close()

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0')

1. Create the requirements file (requirements.txt):  
   Flask  
   mysql-connector-python
2. Create the Dockerfile for the Flask app:

# webapp/Dockerfile

FROM python:3.8-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY app.py .

CMD ["python", "app.py"]

#### **Step 3: Create the Docker Compose File**

Navigate back to the root of your project directory and create a docker-compose.yml file:

# docker-compose.yml

version: '3.8'

services:

mysql:

image: mysql:5.7

environment:

MYSQL\_ROOT\_PASSWORD: rootpassword

MYSQL\_DATABASE: testdb

MYSQL\_USER: user

MYSQL\_PASSWORD: password

ports:

- "3306:3306"

volumes:

- mysql-data:/var/lib/mysql

healthcheck:

test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]

interval: 10s

timeout: 5s

retries: 3

webapp:

build: ./webapp

ports:

- "5000:5000"

depends\_on:

mysql:

condition: service\_healthy

environment:

MYSQL\_HOST: mysql

MYSQL\_DATABASE: testdb

MYSQL\_USER: user

MYSQL\_PASSWORD: password

volumes:

mysql-data:

#### **Step 4: Initialize the Database**

We don't need a specific initialization script for this simplified example, but we will make sure the database is set up correctly by Docker Compose.

#### **Step 5: Run the Application with Docker Compose**

1. Build and Start the Containers:  
   docker-compose up --build
2. Verify the Application:
   * Open your web browser and navigate to http://localhost:5000.

#### **Step 6: Check the Output**

You should see a message indicating a successful connection to the MySQL database, like:

Connected to MySQL Server version 5.7.x. Connected to database: ('testdb',)

If there is an error, you'll see an error message indicating what went wrong with the connection.

### **Conclusion**

This simplified Flask web application demonstrates how to verify a connection to a MySQL database using Docker Compose. It highlights the basics of setting up and running a multi-container application, showing the interaction between a web service and a database service. This example is a great starting point for further development and experimentation with Docker Compose.