**Лабораторная работа №10. Работа с Docker Compose через Windows**

**Цель:** Научиться работать с Docker Compose для создания и управления многоконтейнерными приложениями.

**Задания:**

* Установка Docker Compose.
* Создание многоконтейнерного приложения с использованием Docker Compose.
* Написание файла docker-compose.yml.
* Запуск и управление многоконтейнерным приложением с помощью Docker Compose.

**Установка Docker Compose**

* Для Windows/MacOS после установки Docker Desktop у вас уже встроен функционал докер компоуза
* Для установки на Linux, например Ubuntu, можно воспользоваться следующей статьей или найти похожую под ваш дистрибутив:  
  https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-20-04

**Создание многоконтейнерного приложения**

Для примера, создадим приложение на Flask с использованием базы данных MySQL

Создаём файл app.py и помещаем туда код:

import mysql.connector  
from flask import Flask, jsonify  
  
app = Flask(\_\_name\_\_)  
  
  
@app.route('/')  
def home():  
 try:  
 conn = mysql.connector.connect(  
 host='db',  
 user='root',  
 password='example',  
 database='testdb'  
 )  
 except Exception as e:  
 return "Got error:", str(e)  
 else:  
 cursor = conn.cursor()  
 cursor.execute("SELECT 1")  
 result = cursor.fetchone()  
 cursor.close()  
 conn.close()  
  
 return f"Hello, MySQL successfully started, got result - {result}"  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(host='0.0.0.0')

Кстати, обратите внимание, что мы указываем как хост – db, то есть мы будем подключаться по имени контейнера во внутренней связке.

Создайте файл requirements.txt для зависимостей:

flask==3.0.3  
mysql-connector-python==8.4.0

Создайте файл **Dockerfile** для веб-приложения:

FROM python:3.12-slim  
WORKDIR /app  
COPY . /app  
RUN pip install -r requirements.txt  
CMD ["python", "app.py"]

Далее создайте файл **docker-compose.yml**, который будет собирать в себе контейнер базы данных MySQL и наше Flask приложение на Python.

version: '3'  
*# Список из сервисов*services:  
 *# Первый сервис - наше приложение python* web:  
 *# Указываем где находится Dockerfile* build: .  
 *# Пробрасываем порт 5001 наружу* ports:  
 - "5001:5000"  
 *# Зависимость от сервиса db, сначала запустится db* depends\_on:  
 - db  
 *# Второй сервис - база данных* db:  
 *# Используем готовый образ с Docker Hub* image: mysql:latest  
 *# Переменные окружения для базы данных* environment:  
 MYSQL\_ROOT\_PASSWORD: example  
 MYSQL\_DATABASE: testdb

**Запуск/остановка многоконтейнерного приложения:**

* Запустите приложение с помощью Docker Compose:

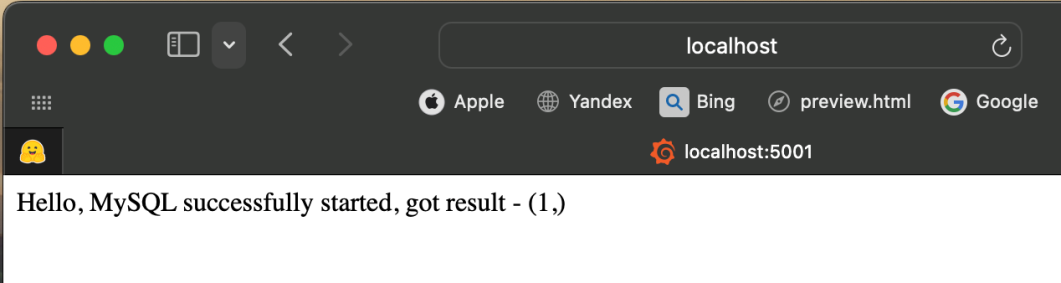
docker-compose up

* Если запуск был с использованием флага -d для запуска в фоновом режиме можно использовать команду:

docker-compose down, либо нажать Ctrl + C, если вы этот флаг не использовали.

**Проверка, что контейнеры действительно запустились:**

Переходим по ссылке <http://localhost:5001> на наше приложение и должны увидеть следующее сообщение:

****

**Дополнительные задания при сдаче работы:**

* Добавьте новый контейнер redis, который вы можете найти на Docker Hub.
* Настройте тома для хранения данных MySQL и Redis, чтобы они не хранились только пока живет контейнер. Для этого используйте **volumes**:

<https://docs.docker.com/compose/compose-file/07-volumes/>

* Ограничьте использование ресурсов для каждого сервиса по CPU и памяти:

<https://docs.docker.com/compose/compose-file/deploy/#resources>

Помните, что документация и Google эта имба, и если вы не умеете ими пользоваться, то время учиться ;(