**Лабораторная работа: Запуск контейнеризированного приложения на виртуальной машине с помощью Docker и Docker Compose (время выполнения 30-35 минут)**

В ходе выполнения лабораторной работы слушатели развернут контейнеризированное приложение, где взаимодействие клиента с сервером осуществляется через HTTP-запросы к приложению Flask, а сервер сохраняет данные в локальной базе данных SQLite. Все компоненты приложения работают внутри Docker-контейнера, при этом база данных монтируется в отдельную папку хоста через **volume** — это механизм Docker, позволяющий сохранять данные за пределами контейнера, чтобы они не терялись при его перезапуске или удалении.

**Цель**

Научиться развёртывать простое клиент-серверное приложение с использованием Docker и Docker Compose на виртуальной машине.

**Требования**

* Виртуальная машина с установленной ОС Linux (например, Ubuntu 24.04)
* Установленные Docker и Docker Compose
* Доступ к терминалу с правами суперпользователя

**План лабораторной работы**

1. **Подготовка окружения**
2. **Создание клиент-серверного приложения**
3. **Написание Dockerfile и docker-compose.yml**
4. **Сборка и запуск контейнеров**
5. **Тестирование приложения**
6. **Завершение работы и очистка ресурсов**

**1. Подготовка окружения**

Установите Docker и Docker Compose на виртуальной машине:

С инструкцией установки вы можете ознакомиться на сайте - <https://docs.docker.com/engine/install/>, где представлены инструкции для установки Docker на различные операционные системы, в моём случае я буду использовать Ubuntu (<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository> ).

Для начала мы установим требуемое ПО и добавим официальный репозиторий Docker в пакетный менеджер:

*sudo apt-get update*

*sudo apt-get install ca-certificates curl -y*

*sudo install -m 0755 -d /etc/apt/keyrings*

*sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc*

*sudo chmod a+r /etc/apt/keyrings/docker.asc*

После этого мы добавим ключ репозитория (вся конструкция — это одна команда):

*echo \*

*"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \*

*$(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \*

*sudo tee /etc/apt/sources.list.d/docker.list > /dev/null*

*sudo apt-get update*

Теперь мы готовы приступить к установке Docker, Compose и сопутствующего ПО:

*sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y*

Далее мы добавим нашего пользователя в группу docker, чтобы он мог использовать docker без повышения привилегий:

*sudo usermod -aG $USER*

после этого перезагрузите систему и проверьте работоспособность docker командой:

*docker run hello-world*

полученное сообщение подтвердит, что вы всё правильно установили и настроили:



**2. Создание веб-приложения**

## Описание приложения

**Тип:**  
Веб-приложение "Заметки" на Flask, работающее в контейнере Docker.

## Основные функции

* **Инициализация базы данных**
  + При запуске создаёт базу данных SQLite (data.db), если она ещё не существует.
  + Таблица notes с полями:
    - id (целое число, автоинкремент)
    - content (текст заметки)
* **Добавление заметок**
  + Пользователь через веб-форму вводит текст новой заметки.
  + Приложение сохраняет запись в базу данных.
* **Отображение списка заметок**
  + На главной странице отображаются все сохранённые заметки в виде списка.
* **Минималистичный дизайн с Bootstrap**
  + Современный аккуратный внешний вид благодаря подключению Bootstrap 5 через CDN.
* **Работа через браузер**
  + Взаимодействие осуществляется полностью через web-интерфейс.

| **Компонент** | **Реализация** |
| --- | --- |
| Язык программирования | Python 3.9 |
| Web-фреймворк | Flask |
| База данных | SQLite |
| Frontend | HTML + Bootstrap 5 |
| Контейнеризация | Docker, Docker Compose |

**Создайте структуру проекта:**

*mkdir -p ~/docker\_lab/{app.data}*

*cd ~/docker\_lab*

**Создайте файл app.py в каталоге app со следующим содержимым:**

*from flask import Flask, request, redirect, url\_for, render\_template\_string*

*import sqlite3*

*import os*

*app = Flask(\_\_name\_\_)*

***# Путь к базе данных внутри контейнера***

*DB\_PATH = '/app/data/data.db'*

*def init\_db():*

*os.makedirs(os.path.dirname(DB\_PATH), exist\_ok=True) # Создать папку data, если нет*

*conn = sqlite3.connect(DB\_PATH)*

*c = conn.cursor()*

*c.execute('CREATE TABLE IF NOT EXISTS notes (id INTEGER PRIMARY KEY AUTOINCREMENT, content TEXT)')*

*conn.commit()*

*conn.close()*

*@app.route('/', methods=['GET', 'POST'])*

*def index():*

*conn = sqlite3.connect(DB\_PATH)*

*c = conn.cursor()*

*if request.method == 'POST':*

*note = request.form['note']*

*c.execute('INSERT INTO notes (content) VALUES (?)', (note,))*

*conn.commit()*

*return redirect(url\_for('index'))*

*c.execute('SELECT id, content FROM notes')*

*notes = c.fetchall()*

*conn.close()*

*return render\_template\_string('''*

*<!DOCTYPE html>*

*<html lang="ru">*

*<head>*

*<meta charset="UTF-8">*

*<meta name="viewport" content="width=device-width, initial-scale=1">*

*<title>Заметки в Docker</title>*

*<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">*

*</head>*

*<body class="bg-light">*

*<div class="container py-5">*

*<h1 class="mb-4 text-center">📝 Мои заметки</h1>*

*<form method="post" class="mb-4">*

*<div class="input-group">*

*<input type="text" name="note" class="form-control" placeholder="Введите новую заметку" required>*

*<button class="btn btn-primary" type="submit">Добавить</button>*

*</div>*

*</form>*

*<div class="card shadow">*

*<div class="card-body">*

*{% if notes %}*

*<ul class="list-group">*

*{% for id, content in notes %}*

*<li class="list-group-item d-flex justify-content-between align-items-center">*

*<span>{{ content }}</span>*

*<span class="badge bg-secondary rounded-pill">#{{ id }}</span>*

*</li>*

*{% endfor %}*

*</ul>*

*{% else %}*

*<p class="text-muted">Пока нет ни одной заметки...</p>*

*{% endif %}*

*</div>*

*</div>*

*</div>*

*</body>*

*</html>*

*''', notes=notes)*

*if \_\_name\_\_ == "\_\_main\_\_":*

*init\_db()*

*app.run(host='0.0.0.0', port=5000)*

**3. Написание Dockerfile**

**Создайте файл Dockerfile (Название обязательно с большой буквы):**

*# Используем официальный минимальный образ Python версии 3.9*

*FROM python:3.9-slim*

*# Копируем файл приложения внутрь контейнера*

*COPY app/app.py app.py*

*# Устанавливаем необходимые зависимости (только Flask)*

*RUN pip install flask*

*# Открываем порт 5000 внутри контейнера*

*EXPOSE 5000*

*# Команда для запуска приложения при старте контейнера*

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

**Наш Dockerfile состоит из ряда инструкций:**

* FROM: базовый образ.
* WORKDIR: где будет работать приложение.
* COPY: перенос файла в контейнер.
* RUN: установка зависимостей.
* EXPOSE: необязательно открывает порт, но служит подсказкой.
* CMD: что запустить по умолчанию.

**4. Создание docker-compose.yml**

**Создайте файл docker-compose.yml:**

*services: # Определение сервисов (контейнеров)*

*web: # Имя сервиса*

*build: . # Построить образ из текущей директории (где находится Dockerfile)*

*ports:*

*- "8080:5000" # Пробрасываем порт: внешний 8080 -> внутренний 5000*

*volumes:*

*- ./data:/app/data # Подключаем папку data на машине в папку /app внутри контейнера*

*# (чтобы база данных SQLite сохранялась вне контейнера)*

**5. Сборка и запуск контейнера**

**Запустите:**

*docker compose up --build -d*

**Приложение будет доступно по адресу:**

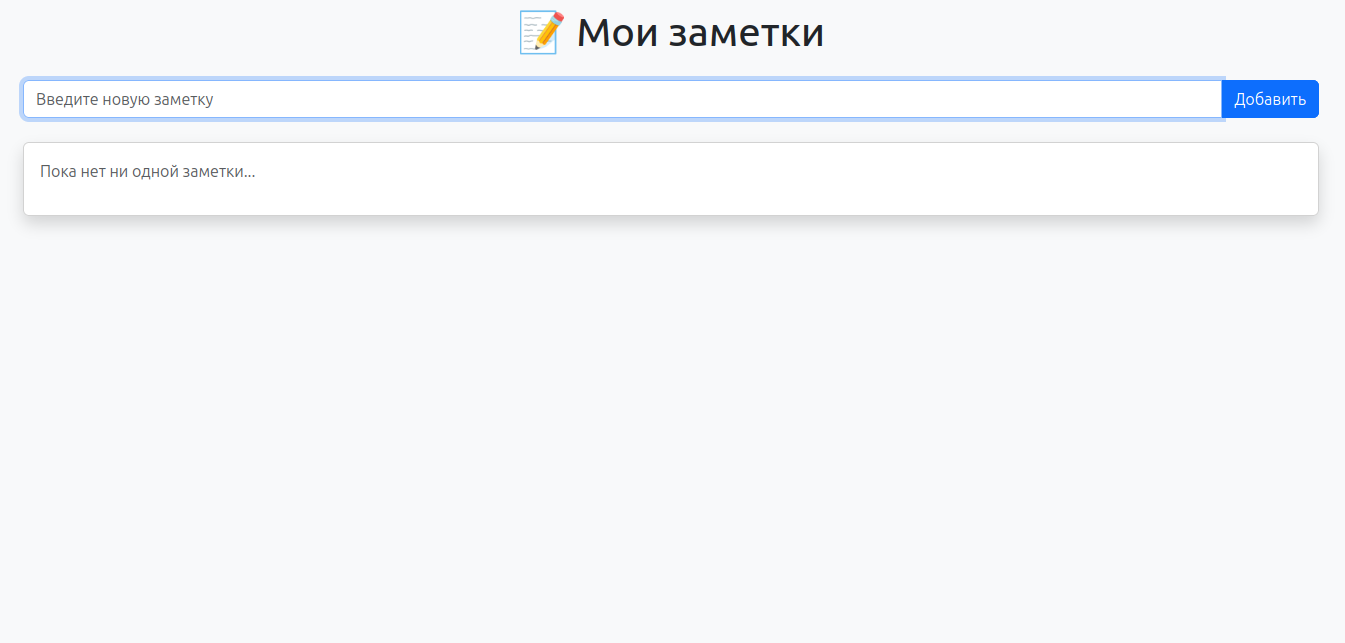
http://<IP-адрес вашей виртуальной машины>:8080

**Наш docker-compose состоит из ряда инструкций:**

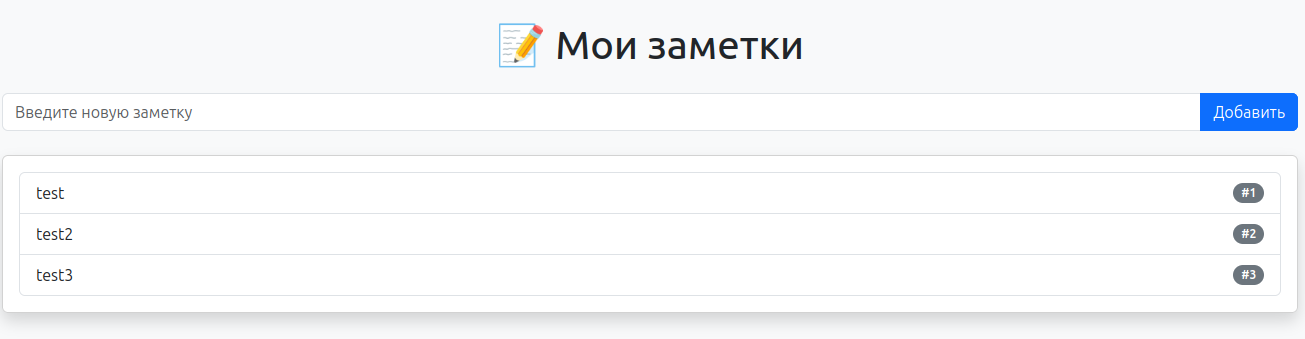
* services: сколько и какие сервисы запускаем.
* build: откуда строим образ (из текущей папки).
* ports: настройка доступа снаружи в контейнер.
* volumes: сохранять данные снаружи контейнера (иначе при пересборке всё пропадёт).

### 6. Тестирование

* Откройте веб-браузер.
* Перейдите на страницу http://<IP-адрес>:8080.



* Введите текст заметки в поле и нажмите **Добавить**.
* Убедитесь, что новая заметка появилась в списке.



**7. Завершение работы**

**Остановите контейнер:**

*docker compose down*

### Результаты

**Вы научились:**

* Создавать веб-приложение с базой данных.
* Контейнеризировать приложение с помощью Docker.
* Управлять несколькими сервисами через Docker Compose.
* Развёртывать и тестировать веб-приложение в браузере.