Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

**Управление контейнерами с помощью Docker Compose   
и использование готовых образов**

**ОТЧЕТ**

**по практической работе № 2**

**по дисциплине «Основы методологии Development Operation»**

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа  
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Студент

группы НМТ-413901 И.В.Машуков

Преподаватель:

профессор, д.т.н. В.В.Лавров

Екатеринбург

2024

СОДЕРЖАНИЕ

[Практическая работа «Управление контейнерами с помощью Docker Compose и использование готовых образов» 3](#_Toc152161721)

[1.1 Цель работы 3](#_Toc152161722)

[1.2 Ход проведения работы 3](#_Toc152161723)

[1.2.1 Разработка тестового приложения Visual Studio .NET Core 3](#_Toc152161724)

[1.2.2 Создание конфигурационного файла docker-compose.yml 4](#_Toc152161725)

[1.2.3 Размещение приложения в системе контроля версий GitHub 5](#_Toc152161726)

[1.2.4 Сборка образов с использованием команды docker compose 5](#_Toc152161727)

[1.2.5 Установка программы HeidiSQL для сопровождения базы данных в СУБД MariaDB 6](#_Toc152161728)

[1.2.6 Настройка миграции базы данных в MariaDB 6](#_Toc152161729)

[1.2.7 Демонстрация работы приложения с базой данных в СУБД MariaDB 6](#_Toc152161730)

[1.3 Выводы 7](#_Toc152161731)

[Приложение А Листинг программного кода приложения «Калькулятор» (контроллер) 8](#_Toc152161732)

[Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json) 9](#_Toc152161733)

[Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json) 10](#_Toc152161734)

[Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs) 11](#_Toc152161735)

# **Практическая работа «Управление контейнерами с помощью Docker Compose и использование готовых образов»**

1.1 Цель работы

Изучить процессы управления контейнерами с использованием Docker Compose, создать и настроить многоконтейнерное приложение, включающее тестовое приложение Visual Studio .NET Core и базу данных MariaDB, а также продемонстрировать взаимодействие этих компонентов через контейнеризацию.

1.2 Ход проведения работы

### 1.2.1 Разработка тестового приложения Visual Studio .NET Core

В рамках выполнения практической работы №1 было создано тестовое приложение на платформе Visual Studio .NET Core под названием «11\_Calculate», реализующее функции стандартного калькулятора: сложение, вычитание, умножение и деление. В процессе работы проект был доработан, дополнительно подключены следующие пакеты NuGet (рис. 1):

* Microsoft.EntityFrameworkCore (6.0.21);
* Microsoft.EntityFrameworkCore.Design (6.0.21);
* Microsoft.EntityFrameworkCore.Tools (6.0.21);
* Pomelo.EntityFrameworkCore.MySql (6.0.2);

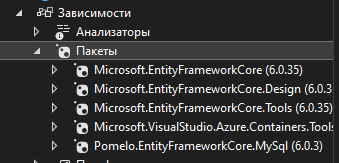


Рисунок 1 - Подключенные пакеты NuGet

Выполнена сборка и протестирована работа приложения, чтобы убедиться - оно выполняет требуемые функции.

Фрагменты Visual Studio c отображением листинга программного кода файла appsettings.json и appsettings.Development.json представлены на рисунках 2,3 соответственно. Листинг кода некоторых файлов представлен в приложениях А-Г.

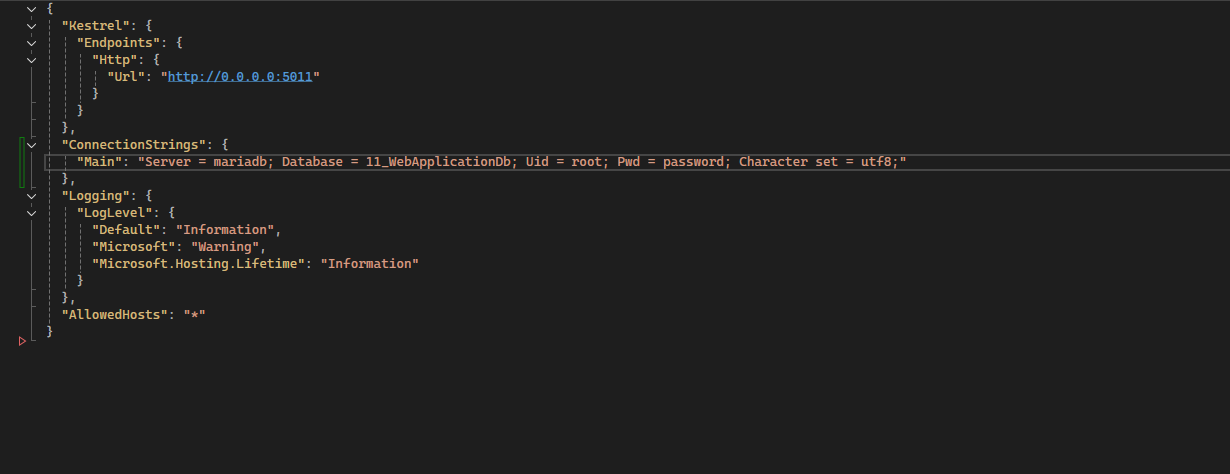


Рисунок 2 - Фрагмент Visual Studio c отображением листинга программного кода файла appsettings.json

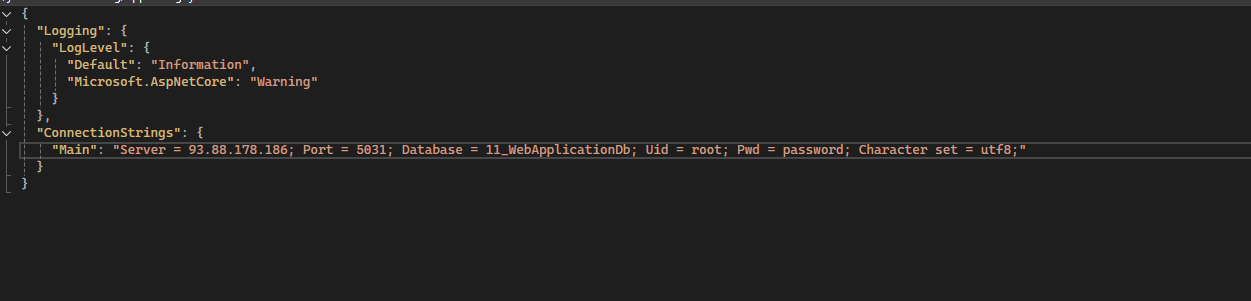


Рисунок 3 - Фрагмент Visual Studio c отображением листинга программного кода файла appsettings.Development.json

### 1.2.2 Создание конфигурационного файла docker-compose.yml

В корневой папке тестового проекта создан файл с именем docker-compose.yml. В этом файле описаны настройки двух контейнеров web-mariadb (база данных MariaDB) и web-app-calculator (веб-приложение «Калькулятор») (рис. 4). Контейнер web-mariadb использует следующий формат портов <IP-сервера>:<Порт-приложения>: внешний порт 5031, а порт приложения 3306. Контейнер web-app-calculator использует для работы следующий формат портов <IP-сервера>:<Порт-приложения>: внешний порт 5011 и внутренний 5011.

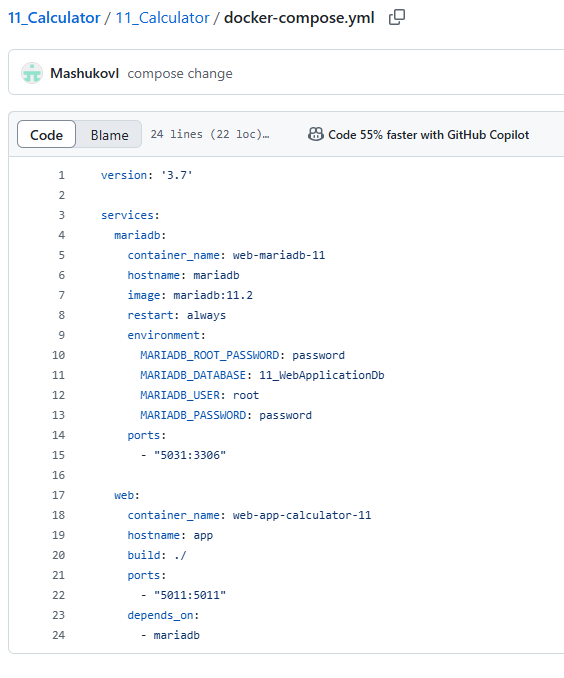


Рисунок 4 - Окно личного репозитория GitHub с размещением тестового проекта с открытым файлом docker-compose.yml

### 1.2.3 Размещение приложения в системе контроля версий GitHub

После того, как тестовое приложение работает без ошибок, загружаем его в личный репозиторий на GitHub.

### 1.2.4 Сборка образов с использованием команды docker compose

В терминальном режиме на удаленном сервере клонируем проект из личного репозитория GitHub. Для этого выполняем команду: git clone <HTTPS>

Здесь <HTTPS> – это HTTPS-адрес проекта в GitHub.

Результаты клонирования представлены на рисунке 5.

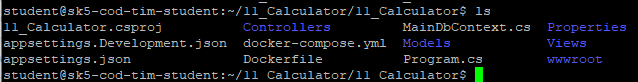


Рисунок 5 - Окно терминальной консоли с результатами клонирования тестового проекта

Далее запускаем многоконтейнерное приложение. Открываем терминал в папке с файлом docker-compose.yml, затем выполняем команду: docker compose up -d

Проверяем существование созданных образов в списке активных контейнеров и их идентификаторов с помощью команды: docker ps -a (рис. 6)

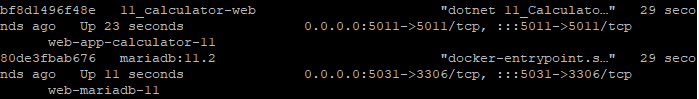


Рисунок 6 - Окно терминальной консоли с демонстрацией созданных образов

### 1.2.5 Установка программы HeidiSQL для сопровождения базы данных в СУБД MariaDB

Скачиваем и устанавливаем на рабочий компьютер программу HeidiSQL с сайта <https://www.heidisql.com/>. Запускаем файл heidisql.exe.

В окне программы создаем новый сеанс. Настраиваем подключение к хосту IP . Вводим следующие реквизиты подключения:

* пользователь root;
* пароль password;
* порт 5031.

Сохраняем настройки сеанса и открываем сеанс. В появившемся окне создаем базу данных с именем в формате 11-WebCalculatorDb.

### 1.2.6 Настройка миграции базы данных в MariaDB

В среде Visual Studio, где открыт тестовый проект «Калькулятор» вызываем Консоль диспетчера пакетов и выполняем команду: add-migration m\_init, в проекте создана папка Migrations с двумя файлами, что тговорит об успешности выполнения команды.

Далее выполняем обновление базы данных с помощью команды: update-database

В окне программы HeidiSQL убедимся, что появилась таблица DataInputVariants (ID\_DataInputVariant, Operand\_1, Operand\_2, Type\_operation, Result).

### 1.2.7 Демонстрация работы приложения с базой данных в СУБД MariaDB

В заключение проверяем работу калькулятора на нашем порту (рис. 7) и результат вычисления проверяем в MariaDB (рис. 8).

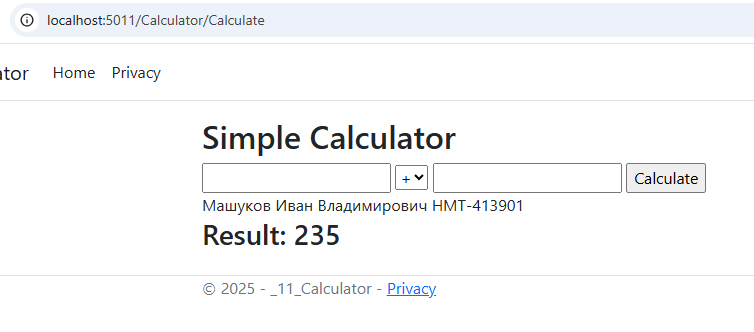


Рисунок 7 - Окно веб-браузера с демонстрацией работающего тестового приложения на выбранном порту

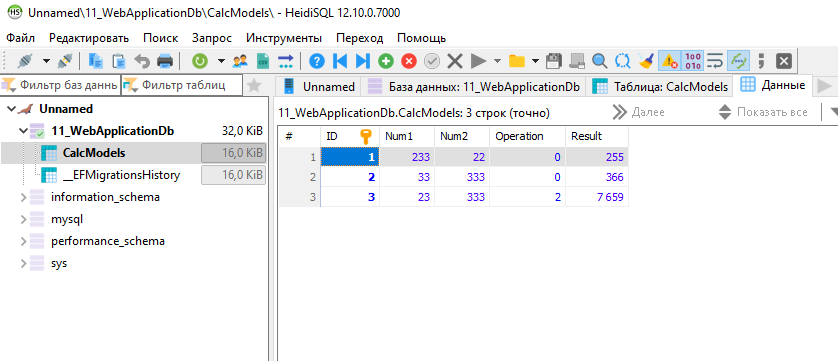


Рисунок 8 - Окно программы HeidiSQL с отображением наполненной базы в MariaDB

1.3 Выводы

В ходе работы был успешно создан и настроен многоконтейнерный проект с использованием Docker Compose. Тестовое приложение «Калькулятор», размещенное в контейнере, взаимодействует с базой данных MariaDB, находящейся в отдельном контейнере. Были проведены тестирования работы приложения, настройки базы данных и миграции данных. Результаты подтвердили корректность работы всех компонентов. Выполненные задачи позволили закрепить навыки разработки и настройки контейнерных приложений, а также взаимодействия с базами данных в контейнерной среде.

Приложение А Листинг программного кода приложения «Калькулятор» (контроллер)

using \_11\_Calculator;

using Microsoft.AspNetCore.Mvc;

namespace \_11\_Calculator.Controllers

{

public enum Operation { Add, Subtract, Multiply, Divide }

public class CalculatorController : Controller

{

private readonly MainDbContext \_dbContext;

public CalculatorController(MainDbContext dbContext)

{

\_dbContext = dbContext;

}

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Calculate(double num1, double num2, Operation operation)

{

double result = 0;

switch (operation)

{

case Operation.Add:

result = num1 + num2;

break;

case Operation.Subtract:

result = num1 - num2;

break;

case Operation.Multiply:

result = num1 \* num2;

break;

case Operation.Divide:

result = num1 / num2;

break;

}

var newRecord = new CalcModel

{

Num1 = num1,

Num2 = num2,

Operation = operation,

Result = result

};

await \_dbContext.CalcModels.AddAsync(newRecord);

await \_dbContext.SaveChangesAsync();

ViewBag.Result = result;

return View("Index");

}

}

} Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json)

{

"Kestrel": {

"Endpoints": {

"Http": {

"Url": "http://0.0.0.0:5011"

}

}

},

"ConnectionStrings": {

"Main": "Server = mariadb; Database = 11\_WebApplicationDb; Uid = root; Pwd = password; Character set = utf8;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json)

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"ConnectionStrings": {

"Main": "Server = 93.88.178.186; Port = 5031; Database = 11\_WebApplicationDb; Uid = root; Pwd = password; Character set = utf8;"

}

}

Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs)

using \_11\_Calculator;

using Microsoft.EntityFrameworkCore;

namespace \_11\_Calculator

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

var connStr = builder.Configuration.GetConnectionString("Main");

builder.Services.AddDbContext<MainDbContext>(opt => opt.UseMySql(connStr, new MySqlServerVersion(new Version(11, 2))));

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Calculator}/{action=Index}/{id?}");

app.Run();

}

}

}