

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»

Сборка и запуск контейнера в Docker

ОТЧЕТ

**по практической работе № 1
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)

Вариант №09

Студент

группы НМТ-413901

Я.В.Крашенинников

Преподаватель:

профессор, д.т.н.

В.В.Лавров

Екатеринбург
2024

Оглавление

1.1 Цель работы	3
1.2 Ход проведения работы.....	3
1.2.1 Разработка тестового приложения Visual Studio .NET Core	3
1.2.2 Создание контейнера с использованием Docker.....	4
1.2.3 Размещение приложения в системе контроля версий GitHub.....	5
1.2.4 Сборка образа с использованием команды docker build	5
1.2.5 Создание контейнера с использованием Docker-образа и запуск.....	5
1.2.6 Демонстрация возможности публикации и остановки контейнера..	6
1.3 Выводы.....	7
Приложение А Листинг программного кода приложения «Калькулятор»... (контроллер).....	8
Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json).....	9
Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json).....	10
Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs)	11

Практическая работа «Сборка и запуск контейнера в Docker»

1.1 Цель работы

Цель работы состоит в настройке и выполнении сборки контейнера в Docker с использованием PuTTY для удаленного доступа к серверу или машине в локальной сети. Основная задача заключается в демонстрации процесса сборки Docker-контейнера на удаленной машине через SSH-подключение с помощью PuTTY. После завершения действий целью является успешная сборка и запуск контейнера в Docker на удаленной машине в локальной сети, обеспечивая тем самым возможность управления и тестирования приложения в изолированной среде на удаленной платформе.

1.2 Ход проведения работы

1.2.1 Разработка тестового приложения Visual Studio .NET Core

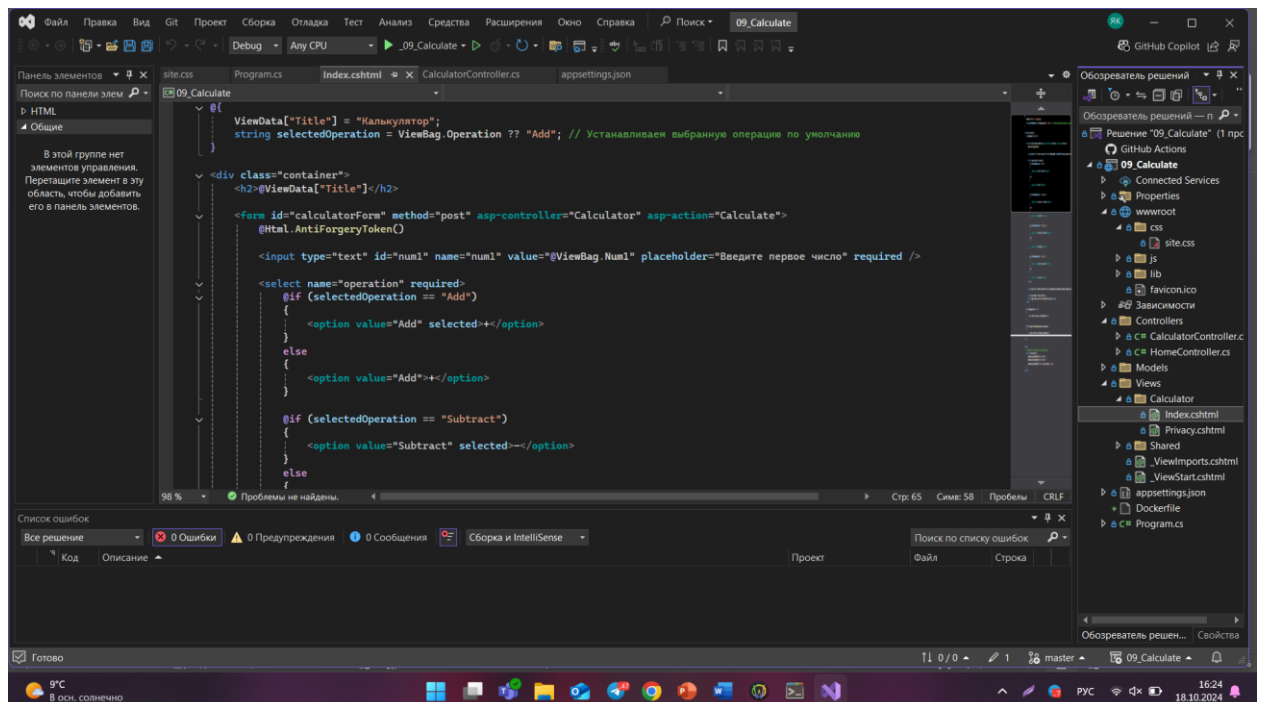


Рисунок 1 – Создание MVC приложения в MVS2022

1.2.2 Создание контейнера с использованием Docker

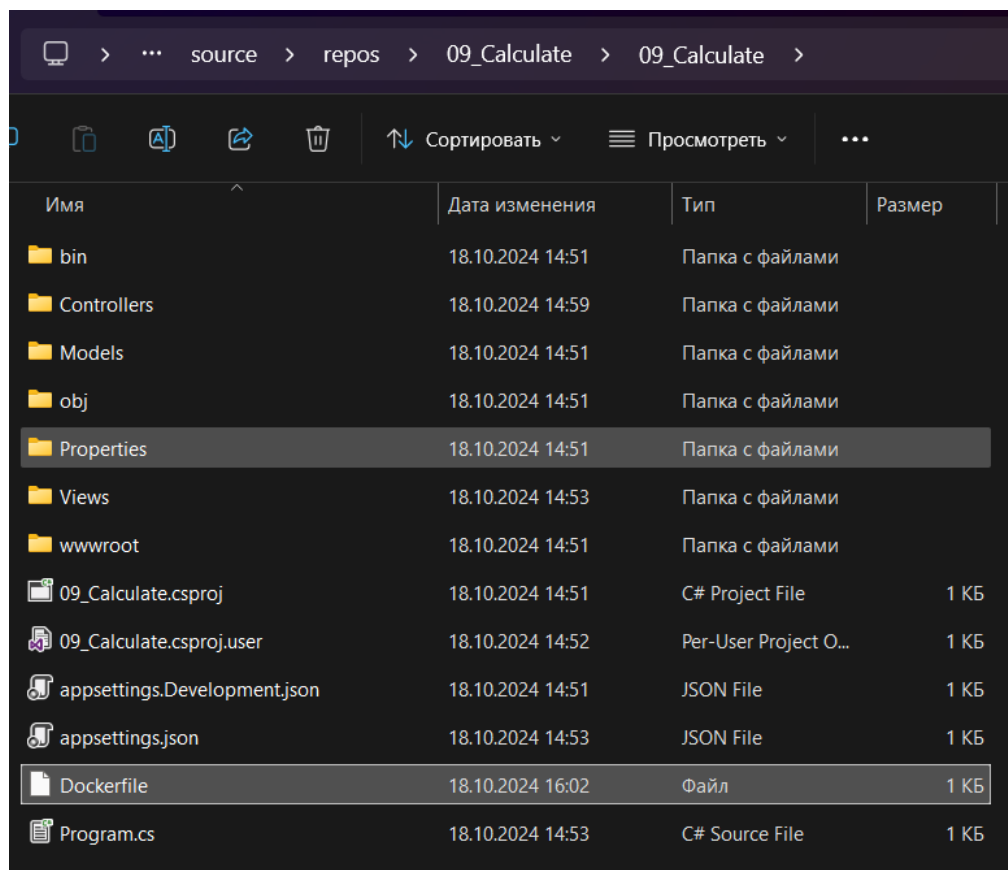


Рисунок 2 – Создание докерфайла в корневой папке проекта

Листинг кода:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
COPY . /src
WORKDIR /src
RUN ls
RUN dotnet restore
RUN dotnet build ".\09_Calculate.csproj" -c Release -o /app/build
FROM build AS publish
RUN dotnet publish ".\09_Calculate.csproj" -c Release -o /app/publish
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish ./
ENTRYPOINT ["dotnet", "09_Calculate.dll"]
```

1.2.3 Размещение приложения в системе контроля версий GitHub

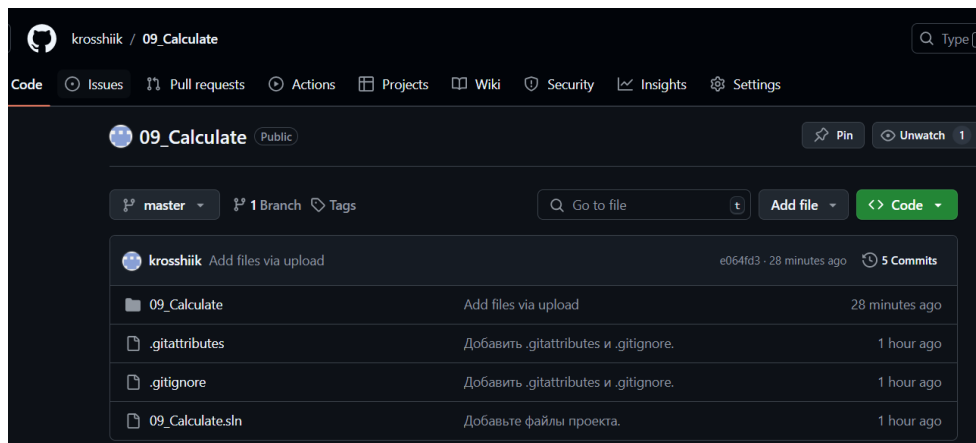


Рисунок 3 -Опубликованный публичный проект на моем Гите

Ссылка - https://github.com/krosshiik/09_Calculate.git

1.2.4 Сборка образа с использованием команды docker build

```
student@sk5-cod-tim-student:~/09_calculate/09_Calculate$ sudo docker build -t 09_calculate:latest .
[+] Building 11.5s (17/17) FINISHED
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 1.03kB 0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:6.0 0.7s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:6.0 0.6s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:6.0@sha256:9772f0741a 0.0s
=> CACHED [build 1/6] FROM mcr.microsoft.com/dotnet/sdk:6.0@sha256:3e749 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 8.15MB 0.1s
=> [build 2/6] COPY . /src 0.1s
=> [build 3/6] WORKDIR /src 0.0s
=> [build 4/6] RUN ls 0.2s
=> [build 5/6] RUN dotnet restore 1.5s
=> [build 6/6] RUN dotnet build ".\09_Calculate.csproj" -c Release -o /a 6.1s
=> [publish 1/1] RUN dotnet publish ".\09_Calculate.csproj" -c Release - 2.5s
=> CACHED [base 2/2] WORKDIR /app 0.0s
=> CACHED [final 1/2] WORKDIR /app 0.0s
=> [final 2/2] COPY --from=publish /app/publish ./ 0.1s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:4913426b688ea2adcda39c589100df82ee1f2fc6988c3 0.0s
=> => naming to docker.io/library/09_calculate:latest 0.0s
```

Рисунок 4 – Создание контейнера с помощью команды `sudo docker build`

1.2.5 Создание контейнера с использованием Docker-образа и запуск

```
=> => naming to docker.io/library/09_calculate:latest 0.0s
student@sk5-cod-tim-student:~/09_calculate/09_Calculate$ sudo docker images
REPOSITORY          TAG             IMAGE ID        CREATED
09_calculate        latest          4913426b688e   About a minute ago
3_calculator        latest          75bc9e31ab96   10 minutes ago
7_calculator        latest          aba5f19889fc   48 minutes ago
8_calculator        latest          efe74c5906e5   2 hours ago
```

Рисунок 5 – Docker images, мой созданный контейнер

1.2.6 Демонстрация возможности публикации и остановки контейнера

```
student@sk5-cod-tim-student:~/09_calculate/09_Calculate$ sudo docker stop ccf539e79655
ccf539e79655
student@sk5-cod-tim-student:~/09_calculate/09_Calculate$ sudo docker start ccf539e79655
ccf539e79655
student@sk5-cod-tim-student:~/09_calculate/09_Calculate$
```

Рисунок 6 – Введенные команды старт стоп

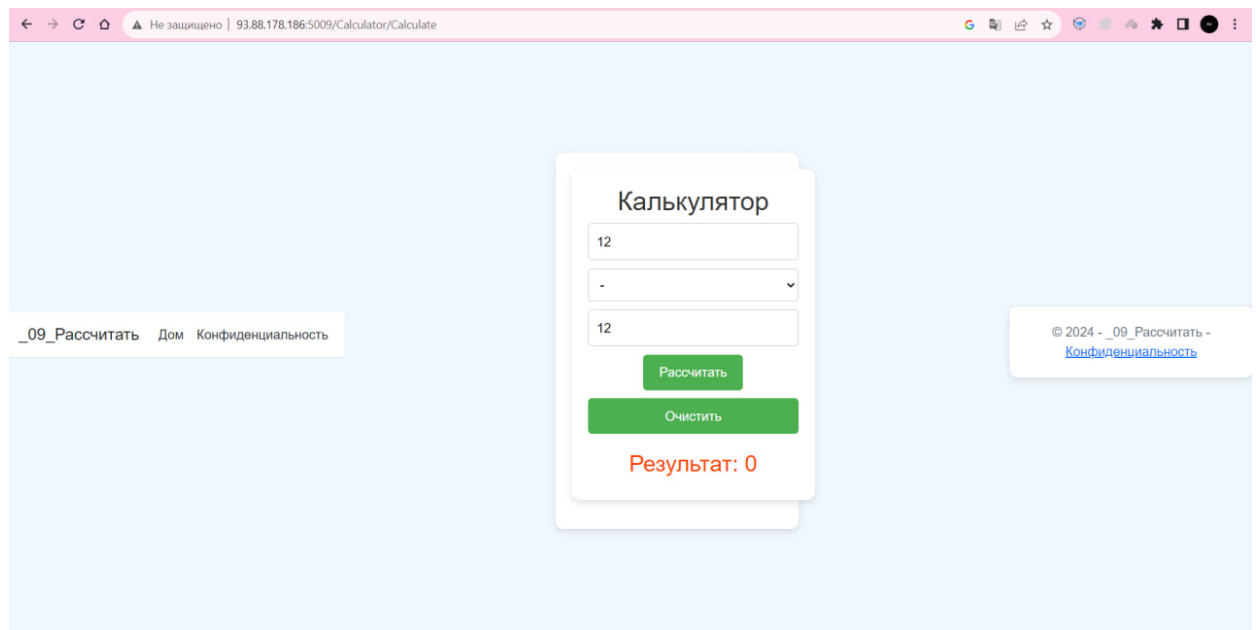


Рисунок 7 – Работающий проект на нужном IP

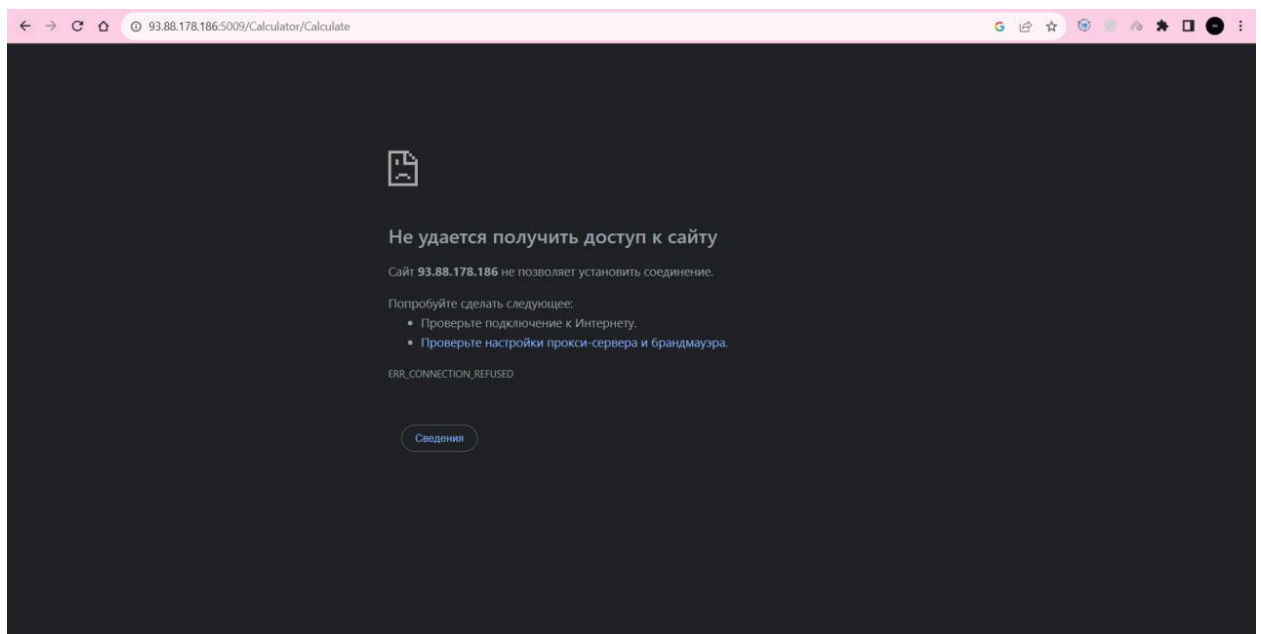


Рисунок 8 – Неработающее приложение после остановки контейнера через Putty

1.3 Выводы

В ходе работы с PuTTY в локальной сети УРФУ и выполнения действий по сборке и запуску контейнеров в Docker мне удалось изучить процесс удаленного доступа через SSH с помощью PuTTY, а также освоить основы контейнеризации приложений с использованием Docker. Это полезные навыки для разработчиков и администраторов, позволяющие управлять удаленными серверами, контейнеризировать приложения для удобного развертывания и масштабирования. В ходе работы я настраивал и устанавливал PuTTY для удаленного доступа и создавал, собирал и запускал контейнеры в Docker для изоляции приложений и обеспечения их портбельности и удобства управления. Эти навыки помогут мне эффективно управлять и развивать приложения на серверах в локальной сети у заказчика.

Приложение А Листинг программного кода приложения «Калькулятор»

(контроллер)

```
using Microsoft.AspNetCore.Mvc;
namespace _09_Calculate.Controllers
{
    public enum Operation { Add, Subtract, Multiply, Divide }

    public class CalculatorController : Controller
    {
        [HttpGet]
        public IActionResult Index()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Calculate(double num1, double num2, Operation operation)
        {
            double result = 0;
            string errorMessage = null;
            try
            {
                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:
                        if (num2 != 0)
                        {
                            result = num1 / num2;
                        }
                        else
                        {
                            errorMessage = "Ошибка: деление на ноль невозможно.";
                        }
                        break;
                }
            }
            catch (Exception ex)
            {
                errorMessage = "Произошла ошибка: " + ex.Message;
            }
            ViewBag.Result = result;
            ViewBag.Num1 = num1; // Сохраняем первое число
            ViewBag.Num2 = num2; // Сохраняем второе число
            ViewBag.Operation = operation.ToString(); // Сохраняем выбранную операцию
            ViewBag.ErrorMessage = errorMessage; // Передаем сообщение об ошибке, если оно есть

            return View("Index");
        }
    }
}
```


Приложение Б Листинг программного кода приложения «Калькулятор»

(appsettings.json)

```
{
  "Kestrel": {
    "Endpoints": {
      "Http": {
        "Url": "http://0.0.0.0:5009" // 09 – номер моего варианта
      }
    }
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

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

```
"Logging": {  
  "LogLevel": {  
    "Default": "Information",  
    "Microsoft": "Warning",  
    "Microsoft.Hosting.Lifetime": "Information"
```

Приложение Г Листинг программного кода приложения «Калькулятор»

(Program.cs)

```
var builder = WebApplication.CreateBuilder(args);

// 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");
    // The default HSTS value is 30 days. You may want to change this for production scenarios,
    see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Calculator}/{action=Index}/{id?}");

app.Run();
```