

Департамент образования города Москвы

Государственное автономное образовательное учреждение  
высшего образования города Москвы  
«Московский городской педагогический университет»

Институт цифрового образования  
Департамент информатики, управления и технологий

**Лабораторная работа 3.1**  
**по дисциплине «Интеграция и развертывание программного**  
**обеспечения с помощью контейнеров»**

**Тема: «Compose для мультиконтейнерных приложений»**

Направление подготовки 38.03.05 – бизнес-информатика  
Профиль подготовки «Аналитика данных и эффективное управление»  
(очная форма обучения)

Выполнила:  
Студентка группы АДЭУ-211  
St\_88

Москва  
2025

## **ВВЕДЕНИЕ**

Цель работы: освоить использование Docker Compose для управления многоконтейнерными приложениями.

Задачи:

1. Создать файл `docker-compose.yml` для указанного многоконтейнерного приложения.
2. Запустить приложение с помощью Docker Compose.
3. Проверить работоспособность приложения и взаимодействие между контейнерами.
4. Выполнить индивидуальное задание.

**Вариант 1.** Создать файл `docker-compose.yml` для многоконтейнерного приложения анализа продаж (Node.js + MongoDB).

Запустить приложение и проверить взаимодействие между контейнерами через API.

Добавить аналитический отчет по общей выручке за месяц.

## ХОД РАБОТЫ

### 1. Создание рабочей директории:

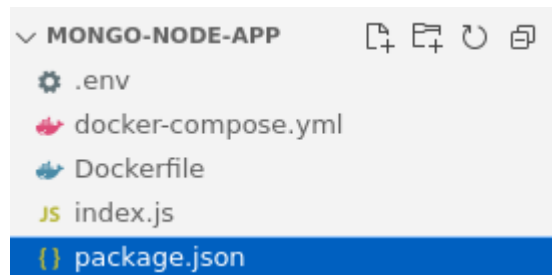
```
dev@dev-vm:~$ mkdir node-mongo-app
```

Для дальнейшего удобства работы переходим в VS Code в созданную папку.

### 2. Создаем и наполняем файлы проекта:

```
dev@dev-vm:~/mongo-node-app$ tree
```

```
.
├── docker-compose.yml
├── Dockerfile
├── index.js
└── package.json
```



- Index.js – основной код приложения на Node.js (прописаны функции приложения: подключение к MongoDB, генерация и запись данных о продажах в таблицу sales, агрегация данных и обновление таблицы с общей выручкой по месяцам monthly\_revenue, запросы POST и GET)
- Package.js – с основной информацией о приложении, скриптами и зависимостями;
- Dockerfile для приложения;
- Docker-compose.yaml – для управления многоконтейнерным приложением. В данном случае, оно будет состоять из 3-х контейнеров: контейнер приложения, контейнер базы данных MongoDB и контейнером серверного фреймворка веб-приложений для Node.js – Mongo Express.
- .env - для изоляции приложения от среды, в которой оно запускается.

### 3. Листинг файла Docker-compose.yaml:

```
version: '3.8' - версия приложения
services:
  app:
    build: . - сборка образа из Dockerfile
    ports:
      - "${APP_PORT}:${APP_PORT}" - определение порта из переменной
окружения
    env_file:
      - .env - загрузка переменных окружения
    depends_on:
      - mongo - зависимость от сервиса mongo
    environment:
      - MONGO_URI=mongodb://mongo:27017/salesdb - переопределяет
переменную

  mongo:
    image: mongo:4.4 - определение образа монго на 4.4 (т.к. он без
AVX, а более новые версии с AVX у меня не работает)
    restart: always - перезапуск при падении
    ports:
      - "${MONGO_PORT}:${MONGO_PORT}" - определение порта для монго
    environment:
      MONGO_INITDB_ROOT_USERNAME: "" - логин для монго (пустой)
      MONGO_INITDB_ROOT_PASSWORD: "" - пароль для монго (пустой)
    volumes:
      - mongo-data:/data/db - сохранение данных

  mongo-express:
    image: mongo-express:1.0.0 - определение версии образа для mongo
express
    restart: always - перезапуск при падении
    ports:
      - "${MONGO_EXPRESS_PORT}:${MONGO_EXPRESS_PORT}" - определение
портов из переменных окружения для mongo express
```

depends\_on:

- mongo - *зависимость от монго*

environment: - *определение окружения для mongo express*

```
ME_CONFIG_MONGODB_ADMINUSERNAME: ""
ME_CONFIG_MONGODB_ADMINPASSWORD: ""
ME_CONFIG_MONGODB_SERVER: mongo
ME_CONFIG_MONGODB_PORT: ${MONGO_PORT}
ME_CONFIG_MONGODB_AUTH_DATABASE: admin
ME_CONFIG_BASICAUTH_USERNAME: "admin"
ME_CONFIG_BASICAUTH_PASSWORD: "admin"
```

volumes:

mongo-data: - *сохранение данных, тем самым они не будут теряться при перезапуске контейнеров*

#### 4. Когда все файлы проекта готовы, билдим и запускаем контейнеры:

```
dev@dev-vm:~/mongo-node-app$ docker compose up --build
WARN[0000] /home/dev/mongo-node-app/docker-compose.yml: the attribute `version`
s obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 16/16
✓ mongo-express Pulled 66.4s
✓ 1207c741d8c9 Pull complete 2.8s
✓ c44c11c253b8 Pull complete 29.2s
✓ b70131bf260b Pull complete 29.9s
✓ 8483f35fc686 Pull complete 30.1s
✓ baf095b3bcb9 Pull complete 60.7s
✓ a70ccd4eb223 Pull complete 60.9s
✓ mongo Pulled 79.6s
✓ 9cb31e2e37ea Already exists 0.0s
✓ 8c7403f42a60 Pull complete 3.1s
✓ 93c62e1a9e41 Pull complete 5.7s
✓ daf549cfb1b2 Pull complete 6.0s
✓ a726b42felaa Pull complete 6.3s
✓ 50bff903985a Pull complete 6.5s
✓ 754e4a0eb02b Pull complete 73.1s
✓ 735c8b80313f Pull complete 73.2s
Compose now can delegate build to bake for better performances
Just set COMPOSE_BAKE=true
[+] Building 26.7s (11/11) FINISHED docker:default
=> [app internal] load build definition from Dockerfile 0.2s
=> => transferring dockerfile: 171B 0.1s
=> [app internal] load metadata for docker.io/library/node:14-alpine 4.5s
```

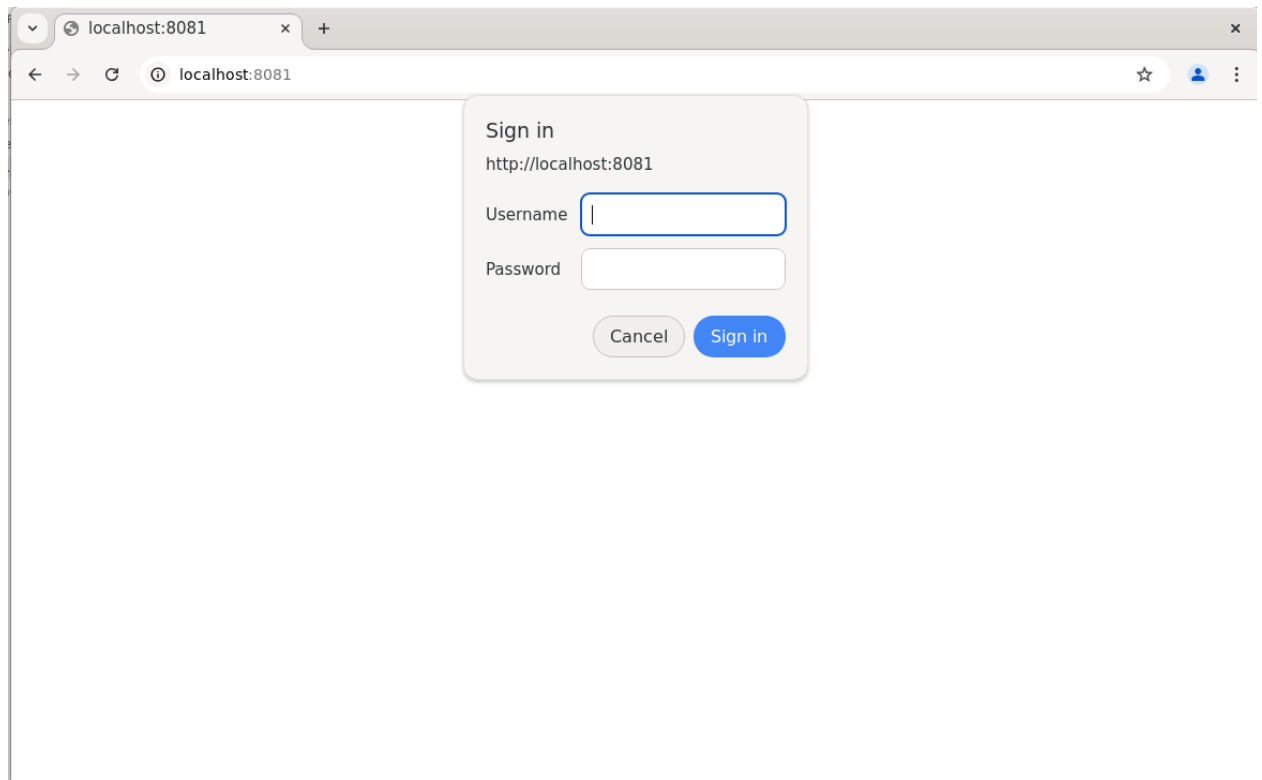
Проверка, что все три контейнера поднялись:

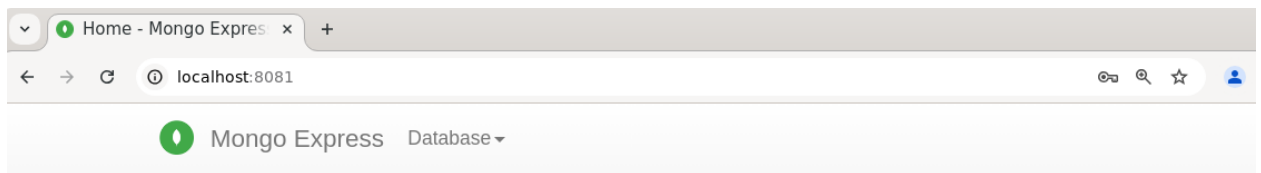
```

dev@dev-vm:~/mongo-node-app$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
c14d3fcb253a   mongo-express:1.0.0                "/sbin/tini -- /dock... 2 minutes ago   Up 23 seconds   0.0
.0.0:8081->8081/tcp, [::]:8081->8081/tcp   mongo-node-app-mongo-express-1
d5811973fef9   mongo-node-app-app                 "docker-entrypoint.s... 2 minutes ago   Up 2 minutes    0.0
.0.0:3000->3000/tcp, [::]:3000->3000/tcp   mongo-node-app-app-1
31762a447d79   mongo:7.0                          "docker-entrypoint.s... 2 minutes ago   Restarting (132) 15 seconds ago
mongo-node-app-mongo-1
dev@dev-vm:~/mongo-node-app$

```

## Проверка доступности Mongo Express на порту 8081:





## Mongo Express

Databases			
Database Name		+ Create Database	
View	admin	Del	
View	config	Del	
View	local	Del	
View	salesdb	Del	

Server Status			
Hostname	775c50a46f43	MongoDB Version	4.4.29
Uptime	260 seconds	Node Version	18.19.0

С помощью curl передаем запрос POST для генерации данных о продажах, записи в таблицу sales и обновления таблицы с отчетом по общей выручке по месяцам:

```
dev@dev-vm:~/mongo-node-app$ curl -X POST http://localhost:3000/sale
{"message":"Продажа сохранена и выручка обновлена","sale":{"item":"Product 25","quantity":3,"price":60.26,"date":"Thu Apr 03 2025 15:26:53","_id":"67eea8bdac591223d26be17a"},"id":"67eea8bdac591223d26be17a","monthlyRevenue":194.63}
dev@dev-vm:~/mongo-node-app$ curl -X POST http://localhost:3000/sale
{"message":"Продажа сохранена и выручка обновлена","sale":{"item":"Product 5","quantity":1,"price":13.85,"date":"Thu Apr 03 2025 15:27:33","_id":"67eea8e5ac591223d26be17b"},"id":"67eea8e5ac591223d26be17b","monthlyRevenue":194.63}
dev@dev-vm:~/mongo-node-app$
```

Проверка, что данные записались в общие таблицы с помощью curl:

```
dev@dev-vm:~/mongo-node-app$ curl -X GET http://localhost:3000/sales
[{"_id":"67eea8bdac591223d26be17a","item":"Product 25","quantity":3,"price":60.26,"date":"Thu Apr 03 2025 15:26:53"},{"_id":"67eea8e5ac591223d26be17b","item":"Product 5","quantity":1,"price":13.85,"date":"Thu Apr 03 2025 15:27:33"}]
dev@dev-vm:~/mongo-node-app$

dev@dev-vm:~/mongo-node-app$ curl -X GET http://localhost:3000/revenue/monthly
[{"_id":"67eea8be75c21b4cd1f57eb9","month":4,"year":2025,"createdAt":"2025-04-03T15:26:54.015Z","totalRevenue":194.63,"updatedAt":"2025-04-03T15:27:33.726Z"}]
dev@dev-vm:~/mongo-node-app$
```

Для большей наглядности проверим в Mongo Express:

Таблица с записями по продажам:

localhost:8081/db/salesdb/sales

Mongo Express Database: salesdb Collection: sales

## Viewing Collection: sales

New Document

New Index

Simple

Advanced

Key

Value

String

Find

Delete all 2 documents retrieved

_id	item	quantity	price	date
<div></div> 67eea8bdac591223d26be17a	Product 25	3	60.26	Thu Apr 03 2025 15:26:53
<div></div> 67eea8e5ac591223d26be17b	Product 5	1	13.85	Thu Apr 03 2025 15:27:33

Таблица с общей суммой продаж по месяцам:



localhost:8081/db/salesdb/monthly\_revenue

Mongo Express Database: salesdb Collection: monthly\_revenue

## Viewing Collection: monthly\_revenue

New Document

New Index

Simple

Advanced

Key

Value

String











Find

Delete all 1 documents retrieved

_id	month	year	createdAt	totalRevenue	updatedAt
<div><div></div><div>67eea8be75c21b4cd1f57eb9</div></div>	4	2025	Thu Apr 03 2025 15:26:54 GMT+0000 (Coordinated Universal Time)	194.63	Thu Apr 03 2025 15:27:33 GMT+0000 (Coordinated Universal Time)

Для проверки корректности сгенерируем записей о продажах, и проверим, что сумма общих продаж за апрель 2025 года увеличилась:

Добавили 18 записей:

localhost:8081/db/salesdb/sales					
Mongo Express Database: salesdb Collection: sales					
Delete all 18 documents retrieved					
<div> <div>← First</div> <div>← Prev</div> <div>Next →</div> <div>Last →</div> </div>					
_id	item	quantity	price	date	
 67eea8bdac591223d26be17a	Product 25	3	60.26	Thu Apr 03 2025 15:26:53	
 67eea8e5ac591223d26be17b	Product 5	1	13.85	Thu Apr 03 2025 15:27:33	
 67eead12ac591223d26be17c	Product 24	6	47.19	Thu Apr 03 2025 15:45:22	
 67eead16ac591223d26be17d	Product 48	2	41.61	Thu Apr 03 2025 15:45:26	
 67eead17ac591223d26be17e	Product 80	7	10.94	Thu Apr 03 2025 15:45:27	
 67eead19ac591223d26be17f	Product 70	3	68.33	Thu Apr 03 2025 15:45:29	
 67eead1aac591223d26be180	Product 91	1	23.57	Thu Apr 03 2025 15:45:30	
 67eead1bac591223d26be181	Product 52	6	72.02	Thu Apr 03 2025 15:45:31	
 67eead1eac591223d26be182	Product 57	1	12.51	Thu Apr 03 2025 15:45:34	
 67eead23ac591223d26be183	Product 96	5	18.63	Thu Apr 03 2025 15:45:39	

И видим, что общая выручка за месяц актуализировалась:

# Viewing Collection: monthly\_revenue

New Document

New Index

Simple

Advanced

Key

Value

String

Find

Delete all 1 documents retrieved

_id	month	year	createdAt	totalRevenue	updatedAt
<div><div></div><div>67eea8be75c21b4cd1f57eb9</div></div>	4	2025	Thu Apr 03 2025 15:26:54 GMT+0000 (Coordinated Universal Time)	2754.06	Thu Apr 03 2025 15:47:24 GMT+0000 (Coordinated Universal Time)

## **ВЫВОДЫ**

Таким образом, в ходе выполнения лабораторной работы были выполнены все поставленные задачи:

1. Создан файл `docker-compose.yml` для `yml` для многоконтейнерного приложения анализа продаж (Node.js + MongoDB);
2. С помощью `docker compose` были успешно запущены все 3 контейнера приложения;
3. Была успешно протестирована работа приложения – данные генерируются записываются в базу данных.

Таким образом, была достигнута главная цель лабораторной работы - освоить использование `Docker Compose` для управления многоконтейнерными приложениями.

Полученное приложение вполне может использоваться в компании для фиксации общего уровня объема продаж по месяцам.

## Описание файла приложения

1. Файл `index.js`, в котором прописываются сами действия, выполняемые будущим приложением, а именно:

1.1. Подключение к MongoDB, инициализация двух баз данных:

```
JS index.js > ...
1  const express = require('express');
2  const { MongoClient } = require('mongodb');
3
4  const app = express();
5  const port = 3000;
6  const mongoUri = process.env.MONGO_URI || 'mongodb://localhost:27017/salesdb';
7
8  let db;
9  function cleanDateString(dateString) {
10 |   return dateString.replace(/(\(.*\)|\s*\+.*)/, '');
11 | }
12
13 async function initDB() {
14 |   try {
15 |     const client = new MongoClient(mongoUri);
16 |     await client.connect();
17 |     db = client.db();
18
19 |     const collections = await db.listCollections().toArray();
20 |     const collectionNames = collections.map(c => c.name);
21
22 |     if (!collectionNames.includes('sales')) {
23 |       await db.createCollection('sales');
24 |       console.log('Коллекция sales создана');
25 |     }
26
27 |     if (!collectionNames.includes('monthly_revenue')) {
28 |       await db.createCollection('monthly_revenue');
29 |       console.log('Коллекция monthly_revenue создана');
30 |     }
31
32 |     console.log('Подключение к MongoDB установлено');
33 |   } catch (error) {
34 |     console.error('Ошибка подключения к MongoDB:', error);
35 |   }
36 | }
```

1.2. Генерация записи данных о продажах:

```
function generateSale() {
  return {
    item: 'Product ' + Math.floor(Math.random() * 100),
    quantity: Math.floor(Math.random() * 10) + 1,
    price: parseFloat((Math.random() * 100).toFixed(2)),
    date: cleanDateString(new Date().toString().replace(" ", "").replace("GMT", "+00:00"))
  };
}
```

1.3. Вычисление общей выручки, группировка ее по месяцам, запись и обновление данных в таблице общей выручки по месяцам:

```

50
51 async function updateMonthlyRevenue() {
52   try {
53     const now = new Date();
54     const month = now.getMonth() + 1;
55     const year = now.getFullYear();
56     const firstDay = new Date(year, month - 1, 1);
57     const lastDay = new Date(year, month, 0, 23, 59, 59, 999);
58
59     const aggregationResult = await db.collection('sales').aggregate([
60       {
61         $addFields: {
62           parsedDate: {
63             $dateFromString: {
64               dateString: "$date",
65               timezone: "UTC"
66             }
67           }
68         },
69       },
70       {
71         $match: {
72           parsedDate: {
73             $gte: firstDay,
74             $lte: lastDay
75           }
76         },
77       },
78       {
79         $group: {
80           _id: null,
81           totalRevenue: {
82             $sum: { $multiply: ["$price", "$quantity"] }
83           },
84           count: { $sum: 1 }
85         }
86       }
87     ]).toArray();

```

```

88
89     const totalRevenue = aggregationResult.length > 0
90     ? parseFloat(aggregationResult[0].totalRevenue.toFixed(2))
91     : 0;
92
93     await db.collection('monthly_revenue').updateOne(
94     { month, year },
95     {
96         $set: {
97             totalRevenue,
98             updatedAt: new Date()
99         },
100         $setOnInsert: {
101             createdAt: new Date(),
102             month,
103             year
104         }
105     },
106     { upsert: true }
107 );
108
109     return { month, year, totalRevenue };
110 } catch (error) {
111     console.error('Ошибка при расчете выручки:', error);
112     throw error;
113 }
114 }

```

- 1.4. запросы для генерации данных о продажах и записи в таблицы, а также получения данных из таблиц:

```

116 // API
117 app.post('/sale', async (req, res) => {
118   const sale = generateSale();
119   try {
120     console.log(142142)
121     const result = await db.collection('sales').insertOne(sale);
122     const revenueData = await updateMonthlyRevenue();
123
124     res.json({
125       message: 'Продажа сохранена и выручка обновлена',
126       sale,
127       id: result.insertedId,
128       monthlyRevenue: revenueData
129     });
130   } catch (error) {
131     console.error('Ошибка при сохранении продажи:', error);
132     res.status(500).json({ error: 'Ошибка при сохранении продажи' });
133   }
134 });
135
136 // API
137 app.get('/sales', async (req, res) => {
138   try {
139     const sales = await db.collection('sales').find().toArray();
140     res.json(sales);
141   } catch (error) {
142     res.status(500).json({ error: 'Ошибка при получении данных' });
143   }
144 });
145
146 // API
147 app.get('/revenue/monthly', async (req, res) => {
148   try {
149     const revenue = await db.collection('monthly_revenue')
150       .find()
151       .sort({ year: 1, month: 1 })
152       .toArray();
153     res.json(revenue);
154   } catch (error) {
155     res.status(500).json({ error: 'Ошибка при получении выручки' });
156   }
157 });

```