**Docker Compose: запуск Node.js + MongoDB как один проект**

**📌 3.1 Зачем нужен Docker Compose?**

Когда приложение зависит от **нескольких сервисов** (например, Node.js + база данных MongoDB), управлять ими вручную сложно:

* Нужно запускать и настраивать каждый контейнер отдельно
* Настраивать сети, тома, переменные окружения

➡️ **Docker Compose** решает всё одним YAML-файлом.

**📁 3.2 Структура проекта**

pgsql

КопироватьРедактировать

docker-node-mongo/

├── docker-compose.yml

├── Dockerfile

├── .env

├── package.json

├── index.js

**📘 3.3 Что делает docker-compose.yml?**

Это конфигурация всех сервисов:

* app (Node.js сервер)
* mongo (база данных)
* volumes (для хранения данных)

**Пример:**

yaml

КопироватьРедактировать

version: '3.8'

services:

app:

build: .

ports:

- "3000:3000"

environment:

- MONGO\_URI=mongodb://mongo:27017/mydb

depends\_on:

- mongo

mongo:

image: mongo

ports:

- "27017:27017"

volumes:

- mongo-data:/data/db

volumes:

mongo-data:

**🔍 3.4 Разбор по частям**

**version: '3.8'**

Определяет версию спецификации Docker Compose.  
Версия 3.8 — стабильна и поддерживается в Docker Desktop.

**Секция services**

**app**

yaml

КопироватьРедактировать

app:

build: .

🔹 Собирает контейнер из текущей папки (Dockerfile)

yaml

КопироватьРедактировать

ports:

- "3000:3000"

🔹 Пробрасывает порт хоста 3000 в порт контейнера 3000

yaml

КопироватьРедактировать

environment:

- MONGO\_URI=mongodb://mongo:27017/mydb

🔹 Передаёт переменную окружения в контейнер (можно заменить .env)

yaml

КопироватьРедактировать

depends\_on:

- mongo

🔹 Гарантирует, что Mongo запустится перед Node.js

**mongo**

yaml

КопироватьРедактировать

mongo:

image: mongo

ports:

- "27017:27017"

🔹 Использует официальный образ MongoDB

yaml

КопироватьРедактировать

volumes:

- mongo-data:/data/db

🔹 Хранит данные Mongo в Docker volume, чтобы не терять их при перезапуске

**Секция volumes**

yaml

КопироватьРедактировать

volumes:

mongo-data:

🔹 Объявляет хранилище, доступное Mongo-контейнеру.

**🧠 3.5 Переменные окружения (.env)**

Создай файл .env:

ini

КопироватьРедактировать

MONGO\_URI=mongodb://mongo:27017/mydb

PORT=3000

Заменить в docker-compose.yml:

yaml

КопироватьРедактировать

environment:

- MONGO\_URI=${MONGO\_URI}

**🏗️ 3.6 Пример index.js (Express + MongoDB)**

js

КопироватьРедактировать

const express = require('express');

const mongoose = require('mongoose');

require('dotenv').config();

const app = express();

app.use(express.json());

mongoose.connect(process.env.MONGO\_URI)

.then(() => console.log('✅ Mongo подключена'))

.catch(err => console.error('❌ Ошибка подключения:', err));

app.get('/', (req, res) => res.send('Работает!'));

app.listen(process.env.PORT || 3000, () => {

console.log(`🚀 Сервер слушает ${process.env.PORT || 3000}`);

});

**🚀 3.7 Как запустить**

**Сборка и запуск:**

bash

КопироватьРедактировать

docker-compose up -d --build

**Просмотр логов:**

bash

КопироватьРедактировать

docker-compose logs -f

**Остановка:**

bash

КопироватьРедактировать

docker-compose down

**Удаление с томами:**

bash

КопироватьРедактировать

docker-compose down -v

**🔄 3.8 Общие команды Docker Compose**

| **Команда** | **Описание** |
| --- | --- |
| up | Запуск всех контейнеров |
| up -d | Запуск в фоне |
| build | Сборка образов |
| down | Остановка и удаление |
| logs | Просмотр логов |
| exec app bash | Подключение внутрь контейнера |
| ps | Список работающих контейнеров |

**⚙️ 3.9 Автоперезапуск (restart policy)**

В секции services:

yaml

КопироватьРедактировать

restart: always

Позволяет автоматически перезапускать контейнер, если он упал или сервер был перезагружен.

**📦 3.10 ИТОГ главы**

Теперь ты умеешь:

* Настраивать многоконтейнерное окружение (Node.js + MongoDB)
* Использовать .env и переменные в Compose
* Работать с томами для сохранения данных
* Запускать всё одной командой docker-compose up