**1. Создай Node.js-приложение**

📄 index.js

js

КопироватьРедактировать

const http = require('http');

const PORT = 3000;

http.createServer((req, res) => {

res.writeHead(200);

res.end('Docker работает!');

}).listen(PORT, () => {

console.log(`Сервер на порту ${PORT}`);

});

**2. Создай package.json**

bash

КопироватьРедактировать

npm init -y

**3. Напиши Dockerfile**

📄 Dockerfile

Dockerfile

КопироватьРедактировать

FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "index.js"]

**4. Собери образ и запусти контейнер**

bash

КопироватьРедактировать

docker build -t my-node-app .

docker run -p 3000:3000 my-node-app

Проверь: <http://localhost:3000>

**5. Напиши docker-compose.yml**

yaml

КопироватьРедактировать

version: '3.8'

services:

app:

build: .

ports:

- "3000:3000"

**6. Запусти через Compose**

bash

КопироватьРедактировать

docker-compose up --build

**7. Добавь .env**

📄 .env

ini

КопироватьРедактировать

PORT=3000

И подключи в код:

js

КопироватьРедактировать

require('dotenv').config();

const PORT = process.env.PORT || 3000;

**8. Добавь Volume**

yaml

КопироватьРедактировать

volumes:

- ./logs:/app/logs

**9. Пропиши healthcheck**

yaml

КопироватьРедактировать

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:3000"]

interval: 10s

retries: 3

**10. Добавь переменные окружения в Compose**

yaml

КопироватьРедактировать

environment:

- PORT=3000

**✅ Результат:**

* Работающее API
* Собранный Dockerfile
* Compose с переменными, volume, healthcheck
* Логика запуска полностью контейнеризирована