CI/CD и автоматизация (GitHub Actions + Docker + тесты)

* Запускать тесты и сборку при каждом коммите
* Автоматически собирать Docker-образы
* Готовить проект к деплою в облако
* Делать всё это через **GitHub Actions**

**🧠 8.1 Что такое CI/CD?**

| **Термин** | **Расшифровка** | **Назначение** |
| --- | --- | --- |
| CI | Continuous Integration | Автоматический запуск тестов, линтеров, сборки при пуше |
| CD | Continuous Delivery/Deployment | Авто-доставка кода на прод/сервер |

**🗃️ 8.2 Структура проекта с CI/CD**

css

КопироватьРедактировать

rest-api-project/

├── .github/

│ └── workflows/

│ └── ci.yml

├── Dockerfile

├── docker-compose.yml

├── app.js

├── index.js

├── \_\_tests\_\_/

**🛠️ 8.3 GitHub Actions: минимальная конфигурация**

📄 .github/workflows/ci.yml

yaml

КопироватьРедактировать

name: CI

on:

push:

branches: [ main ]

pull\_request:

branches: [ main ]

jobs:

build-and-test:

runs-on: ubuntu-latest

services:

mongo:

image: mongo

ports:

- 27017:27017

options: >-

--health-cmd "mongosh --eval 'db.runCommand({ ping: 1 })'"

--health-interval 10s

--health-timeout 5s

--health-retries 3

steps:

- name: 📥 Checkout code

uses: actions/checkout@v3

- name: 🟢 Setup Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

- name: 📦 Install dependencies

run: npm install

- name: 🧪 Run tests

env:

MONGO\_URI: mongodb://localhost:27017/testdb

run: npm test

**📍 8.4 Что делает этот workflow?**

| **Этап** | **Что происходит** |
| --- | --- |
| on: push | Запускается при пуше или PR |
| services.mongo | Поднимает MongoDB в CI |
| setup-node | Устанавливает нужную версию Node |
| npm install | Ставит зависимости |
| npm test | Запускает Jest + Supertest |

**🐳 8.5 Сборка Docker-образа в GitHub Actions**

Дополняем workflow:

yaml

КопироватьРедактировать

docker-build:

needs: build-and-test

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: 🔧 Build Docker image

run: docker build -t my-node-app .

- name: 🧪 Docker image test (optional)

run: docker run --rm my-node-app node -v

**📦 8.6 Автоматическая публикация на Docker Hub (пример)**

Добавим секцию:

yaml

КопироватьРедактировать

- name: 🔐 Login to Docker Hub

uses: docker/login-action@v2

with:

username: ${{ secrets.DOCKER\_USERNAME }}

password: ${{ secrets.DOCKER\_PASSWORD }}

- name: 🚀 Push to Docker Hub

run: docker tag my-node-app username/my-node-app:latest &&

docker push username/my-node-app:latest

🎯 Обязательно:

* Добавь переменные DOCKER\_USERNAME и DOCKER\_PASSWORD в Secrets репозитория

**🚀 8.7 Автодеплой на сервер (VPS или Heroku)**

**Для VPS (через ssh/scp):**

1. Установи SSH-ключ в GitHub Secrets
2. Добавь шаг в Actions:

yaml

КопироватьРедактировать

- name: 🔁 Deploy to VPS

uses: appleboy/ssh-action@v0.1.10

with:

host: ${{ secrets.HOST }}

username: ${{ secrets.USER }}

key: ${{ secrets.SSH\_KEY }}

script: |

cd /home/project

git pull

docker-compose down

docker-compose up -d --build

**🛡️ 8.8 Что ещё можно подключить?**

| **Инструмент** | **Назначение** |
| --- | --- |
| ESLint | Проверка кода |
| Prettier | Форматирование |
| Husky | Git-хуки перед коммитом |
| Codecov | Покрытие кода |
| Slack Bot | Уведомления о падении тестов |

**📋 8.9 Пример полной CI/CD-конвейера**

1. Разработчик пушит в main
2. GitHub Actions:
   * Устанавливает зависимости
   * Запускает тесты
   * Собирает Docker-образ
   * Пушит его в DockerHub
   * Отправляет на сервер через SSH
3. Сервер перезапускает docker-compose

✅ Всё — **автоматически**

**✅ 8.10 Результаты**

Теперь ты умеешь:

* Настраивать CI/CD pipeline
* Проверять код при каждом коммите
* Автоматически собирать и выкладывать Docker-образы
* Деплоить проект в реальную инфраструктуру