**Что такое Docker, и зачем он нужен в Node.js-разработке**

**📌 1.1 Введение**

Программисты часто сталкиваются с проблемой, когда приложение:

* Работает на одном компьютере, но не запускается на другом
* Зависит от разных версий Node.js или баз данных
* Ломается при переносе на сервер

❓ Почему так происходит?

➡️ Потому что окружение (ОС, версии библиотек, переменные, порты, зависимости) на всех машинах **разное**.

**Docker** решает эту проблему.

**🧱 1.2 Что такое Docker?**

**Docker** — это инструмент, позволяющий упаковать ваше приложение и всё, что ему нужно (Node.js, MongoDB, зависимости, конфигурации), в **контейнер** — изолированную и воспроизводимую среду.

Контейнер ≠ Виртуальная машина  
Контейнер:

* легче
* быстрее
* использует ядро хост-системы
* запускается за миллисекунды

**🎯 1.3 Основные понятия Docker**

| **Термин** | **Определение** |
| --- | --- |
| 🔹 **Образ (image)** | Шаблон, в котором описано, что нужно для запуска |
| 🔹 **Контейнер (container)** | Запущенный экземпляр образа |
| 🔹 **Dockerfile** | Скрипт, описывающий, как создать образ |
| 🔹 **docker-compose** | YAML-файл для запуска нескольких контейнеров вместе |
| 🔹 **Volume** | Постоянное хранилище данных контейнера |
| 🔹 **Registry** | Репозиторий образов, например, Docker Hub |

**⚙️ 1.4 Как Docker помогает Node.js разработчику?**

**Пример:**

У тебя есть Node.js приложение, использующее:

* Node.js v18
* MongoDB
* .env файл с переменными

Ты можешь:

* 📦 упаковать всё в Docker-контейнер
* 📤 развернуть его на сервере, в CI/CD, в облаке
* 🧪 протестировать в изолированной среде

bash

КопироватьРедактировать

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

Работает **всегда одинаково**, без проблем с ОС, версиями и т.д.

**📈 1.5 Преимущества Docker**

**✅ 1. Кроссплатформенность**

Один контейнер запускается одинаково на:

* Windows
* macOS
* Linux
* Сервере
* Облаке

**✅ 2. Изоляция**

Контейнер не зависит от твоей операционной системы.

**✅ 3. Безопасность**

Контейнеры работают в "песочнице" — ограниченный доступ к хосту.

**✅ 4. Масштабируемость**

Можно легко масштабировать через Docker Swarm, Kubernetes.

**✅ 5. Совместимость с CI/CD**

Идеально интегрируется с GitHub Actions, GitLab CI, Jenkins, Heroku.

**📊 1.6 Разница: Docker vs Виртуальные машины**

| **Параметр** | **ВМ** | **Docker** |
| --- | --- | --- |
| Размер образа | ГБ | МБ |
| Время запуска | Минуты | Секунды |
| Использование RAM | Высокое | Низкое |
| Производительность | Низкая | Почти нативная |
| Изоляция | Полная | На уровне ядра |

**🧠 1.7 Типичный процесс с Docker**

1. ✍️ Пишешь Dockerfile — описание среды
2. 🔨 Собираешь образ:

bash

КопироватьРедактировать

docker build -t my-node-app .

1. 🚀 Запускаешь контейнер:

bash

КопироватьРедактировать

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

1. 💾 Используешь тома (volumes) для хранения данных
2. 📡 Открываешь порты (3000:3000)
3. 🧪 Тестируешь локально → выкатываешь в продакшн

**🔐 1.8 Безопасность контейнеров**

* Контейнер **не имеет доступа** к хосту, кроме того, что ты явно разрешишь
* Используй:
  + USER node вместо root
  + HEALTHCHECK — проверка живости
  + --read-only — запрещённые изменения файлов

**🌐 1.9 Где работает Docker?**

| **Платформа** | **Поддержка Docker** |
| --- | --- |
| ✅ VPS (Ubuntu, Debian) | Отлично |
| ✅ AWS, GCP, Azure | Полная |
| ✅ Heroku | Частично (через Docker CLI) |
| ✅ DigitalOcean | Поддержка контейнеров |
| ✅ GitHub Actions | Есть Docker контейнеры |

**📌 1.10 Итог главы**

Docker — фундаментальный инструмент для:

* разработки
* тестирования
* деплоя
* масштабирования Node.js приложений

Если ты хочешь, чтобы твой код "работал везде" — ты **обязан** уметь работать с Docker.