**Volumes, хранение и управление данными в Docker-контейнерах**

**📌 4.1 Что такое volumes в Docker?**

Когда ты запускаешь контейнер, он работает в **изолированном файловом пространстве**. Все данные хранятся **внутри**.  
Но! Если ты перезапустишь или удалишь контейнер — ❌ данные исчезнут.

➡️ Чтобы сохранить данные между перезапусками, мы используем **volumes**.

**🧱 4.2 Виды хранилищ в Docker**

| **Тип** | **Описание** |
| --- | --- |
| 🔹 **Volumes** | Управляются Docker. Перемещаются между контейнерами. |
| 🔹 **Bind Mounts** | Связывают папку хоста с контейнером напрямую. |
| 🔹 **Tmpfs** | Временное хранилище в оперативной памяти (для Linux). |

**🧪 4.3 Пример: volume для MongoDB**

**docker-compose.yml**

yaml

КопироватьРедактировать

services:

mongo:

image: mongo

volumes:

- mongo-data:/data/db

volumes:

mongo-data:

📦 Docker создаёт volume mongo-data, который сохраняет данные Mongo в /data/db.

➡️ Даже если контейнер удалить, данные останутся.

**🔍 4.4 Где хранятся volumes?**

**Посмотреть список volumes:**

bash

КопироватьРедактировать

docker volume ls

**Посмотреть детали:**

bash

КопироватьРедактировать

docker volume inspect mongo-data

Ты увидишь путь:

json

КопироватьРедактировать

"Mountpoint": "/var/lib/docker/volumes/mongo-data/\_data"

➡️ Там реально лежат файлы MongoDB.

**⚙️ 4.5 Использование bind mount (хоста)**

Иногда ты хочешь напрямую подключить папку твоей машины в контейнер.

yaml

КопироватьРедактировать

volumes:

- ./local-data:/data/db

📁 ./local-data — твоя папка  
📦 /data/db — путь в контейнере

Теперь Mongo хранит данные прямо у тебя в проекте.

**🧠 4.6 Когда использовать volume, когда bind mount?**

| **Сценарий** | **Лучше использовать** |
| --- | --- |
| Продакшн данные | ✅ volume (безопаснее, управляется Docker) |
| Разработка (hot reload) | ✅ bind mount |
| Хранилище логов | volume или bind |
| Настройка локальных конфигов | bind mount (подключить .env) |

**🧪 4.7 Управление volumes (CLI)**

| **Команда** | **Назначение** |
| --- | --- |
| docker volume ls | Список всех volumes |
| docker volume create myvol | Создать volume |
| docker volume inspect myvol | Посмотреть путь и метаданные |
| docker volume rm myvol | Удалить volume |
| docker volume prune | Удалить все неиспользуемые volumes |

**🧪 4.8 Работа с данными внутри контейнера**

Подключись к контейнеру:

bash

КопироватьРедактировать

docker exec -it my\_mongo\_container bash

Зайди в базу:

bash

КопироватьРедактировать

mongosh

Вводи команды:

js

КопироватьРедактировать

use test

db.users.insert({ name: "Alice" })

db.users.find()

Теперь эти данные сохраняются в volume!

**📤 4.9 Экспорт и импорт данных**

**🗃️ Экспорт данных MongoDB**

bash

КопироватьРедактировать

docker exec my\_mongo\_container mongodump --out /backup

docker cp my\_mongo\_container:/backup ./mongo-backup

**📥 Импорт данных**

bash

КопироватьРедактировать

docker cp ./mongo-backup my\_mongo\_container:/backup

docker exec my\_mongo\_container mongorestore /backup

**📓 4.10 Общие best practices по volume'ам**

* Не сохраняй важные данные **внутри** контейнера
* Не храни node\_modules в volume (может вызвать конфликты)
* Храни базы в volume, а не в bind-монтах — это безопаснее
* Используй named volumes, чтобы не терять данные между перезапусками

**✅ Резюме главы**

Теперь ты умеешь:

* Создавать и подключать volumes
* Отличать volumes от bind mounts
* Подключаться к базе и сохранять данные
* Делать дампы и восстанавливать базы