Тема 6

Жизнь контейнера: логи, здоровье, рестарты, ресурсы



Что происходит с сервисом, когда что-то идёт не так, — и как это прочитать

План лекции

Сегодня

1. Что может пойти не так
2. События и exit-коды
3. Политики перезапуска
4. Liveness и readiness
5. Логи, ресурсы, диагностика



После лекции вы сможете

- Определить причину падения по exit-коду и событиям
- Настроить автоматический рестарт
- Объяснить разницу live и ready
- Ограничить ресурсы контейнера

Что может пойти не так

Сбой	Для пользователя	С контейнером
Ошибка в запросе	500 на одном запросе	Жив
Падение процесса	Недоступен несколько секунд	Завершился, перезапущен
Утечка памяти	Тормоза, затем недоступность	Убит ядром (OOM)
Упала БД	500 на запросах к данным	Жив, но не готов
Сервис завис	Запросы висят	Жив, healthcheck не проходит

Задача — понять по следам, что случилось, и часть проблем решать без человека

Exit-коды: как умер процесс

0

Штатное завершение, в т.ч. после `docker stop`

1

Приложение само завершилось с ошибкой (у нас `/api/chaos/crash`)

137

SIGKILL: нехватка памяти (OOM) или не уложился в grace period

143

SIGTERM, процесс не обработал сигнал

Что делает docker stop



SIGTERM

«Аккуратно заверши работу». ASP.NET перестаёт принимать запросы





Grace period

10 секунд на завершение текущих запросов





SIGKILL

Не успел — убит немедленно, exit 137

Сигнал — уведомление процессу от Linux. SIGKILL перехватить нельзя

Политики перезапуска

`restart`	Поведение
`no`	Никогда (по умолчанию)
`on-failure[:N]`	Только при exit ≠ 0, до N раз
`always`	Всегда, даже остановленный вручную после рестарта Docker
`unless-stopped`	Всегда, кроме остановленных вручную — наш выбор

Рестарт лечит случайные сбои. Падение на старте каждый раз — читайте логи

Liveness и readiness



`/health/live` — жив?

- Проверяет только процесс
- Смотрит Docker HEALTHCHECK
- Упал — сервис действительно сломан



`/health/ready` — готов?

- Проверяет зависимости: БД
- Смотрят деплой и мониторинг
- Упала БД — сервис жив, но не готов

Проверять БД в liveness нельзя: при сбое БД «больными» станут все экземпляры сразу



Docker сам НЕ перезапускает unhealthy-контейнеры

Статус используют `depends_on`, `compose up --wait` и мониторинг. Перезапуск по liveness делают оркестраторы — Kubernetes

Логи

```
logging:
  driver: json-file
  options:
    max-size: "10m"      # размер файла
    max-file: "3"        # сколько хранить

# в контейнере сервис пишет JSON:
{ "LogLevel": "Error",
  "Message": "An unhandled exception...",
  "Exception": "..." }
```

1. **stdout, не файлы**

   Docker собирает логи сам, `docker logs`

2. **Ротация**

   Иначе логи съедят диск

3. **JSON**

   Удобно фильтровать и отправлять в системы сбора

4. **Уровни**

   Information, Warning, Error (все 500), Critical

Ресурсы: память и CPU

Лимиты — это cgroups

`memory: 256M`, `cpus: "1.0"`. Без лимитов одна утечка кладёт весь сервер

.NET знает лимит

Куча GC ≤ 75% лимита. Кончилась — `OutOfMemoryException`, 500, процесс жив

Нативная утечка

Не ограничена GC → ядро убивает процесс: `oom` + exit 137 → рестарт

CPU не убивает

Лимит CPU троттлит: сервис становится медленным

Инструменты диагностики

Команда	Что даёт
`docker compose ps -a`	Статус, health, аптайм
`docker logs --tail 100`	Последние слова перед падением
`docker events --since 10m`	Хронология: `die exitCode`, `oom`
`docker inspect -f '{{json .State}}'`	`RestartCount`, `Health.Log`
`docker stats --no-stream`	Память и CPU сейчас
Docker Desktop → Containers	Всё то же в интерфейсе

`OOMKilled` сбрасывается после рестарта — правду хранят события, логи и мониторинг

Алгоритм «контейнер упал»

1

ps -a

Рестартится?
unhealthy?

2

logs

Исключение?
Сообщение о завершении?

3

events

Какой exitCode?
Был oom?

4

stats

Память у лимита?
CPU в потолке?

5

Изменения

Что деплоили последним?

Не делайте `down/up` вместо разбора — уничтожите улики

Типичные ошибки

- Ждать, что Docker перезапустит unhealthy
- Проверять БД в liveness
- Логи в файлы внутри контейнера или без ротации
- Нет лимитов памяти
- `down/up` вместо разбора причины

Проверьте себя

1. Рестарт с exit 137: две причины и как их различить?
2. Почему HEALTHCHECK смотрит на live, а не ready?
3. Чем `always` отличается от `unless-stopped`?
4. Почему после рестарта `OOMKilled=false`?
5. Какие сбои не видны по `docker compose ps`?



Практика

Лаба 6. Ломаем сервис и разбираем сбои

Устраиваем сбои через `/api/chaos` и читаем следы

- Ошибка в запросе, падение процесса, утечка памяти до OOM
- Политика `no` против `unless-stopped`
- Останавливаем БД: live 200, ready 503
- Ломаем healthcheck и проверяем, перезапустит ли Docker