# Неделя 11. Лаба 6. Жизнь контейнера: ломаем сервис и разбираем сбои

*Часть 1 из 2: разделы «1. Осмотр» — «5. Штатная остановка».*

**Цель:** воспроизвести типовые сбои и научиться определять причину по логам, событиям, exit-коду и ресурсам. **Время:** 90 мин. **Теория:** лекция 6. **Инструменты:** Docker Desktop (CLI и вкладка Containers), PowerShell 7.

**Подготовка:** в лабе вы будете временно менять `compose.yaml`, поэтому работаем с локальной сборкой актуального `main`:

```
cd C:\dev\demo-api
git switch main; git pull
docker compose up -d --build --wait      # оба сервиса healthy
$api = 'http://localhost:8080'
```

Откройте **три терминала**:

| Терминал | Команда |
|---|---|
| T1 | для команд лабы |
| T2 | `docker events --filter container=demo-api-1 --filter event=start --filter event=kill --filter event=die --filter event=oom --filter event=stop --filter event=health_status` |
| T3 | `docker stats demo-api-1 demo-db-1` |

Фильтры в T2 отсекают служебные события `exec_*`: их каждые 10 секунд порождает HEALTHCHECK, и в них тонут нужные `die` и `oom`.

По ходу лабы заполняйте таблицу для сдачи:

| Сценарий | События (T2) | Exit code | Рестарт? | Что в логах |
|---|---|---|---|---|

## 1. Осмотр

```
docker compose ps
docker compose logs api --tail 20
docker inspect demo-api-1 --format '{{json .HostConfig.LogConfig}}'   # ротация логов
docker inspect demo-api-1 --format '{{.HostConfig.RestartPolicy.Name}}'
```

Откройте Docker Desktop → *Containers* → `demo-api-1`: вкладки Logs, Inspect, Exec, Stats.

## 2. Ошибка в запросе — процесс жив

```
curl.exe -s -w " [%{http_code}]`n" $api/api/chaos/error
docker compose logs api --tail 3 --no-log-prefix
```

В ответе 500 без деталей, в логе — JSON-запись с `"LogLevel":"Error"` и полем `Exception` со стеком исключения. Контейнер не перезапускался.

**Зачем:** большинство ошибок — это не падение контейнера, а 500 в отдельных запросах. Их видно только в логах и метриках.

## 3. Падение процесса

```
curl.exe -X POST $api/api/chaos/crash
```

В T2: `die` (`exitCode=1`) → `start` → `health_status: healthy`.

```
docker inspect demo-api-1 --format '{{.RestartCount}}'              # 1
docker compose logs api --since 2m --no-log-prefix | Select-String Critical
```

## 4. Политика `no`

В `compose.yaml` у `api` поставьте `restart: "no"`, затем:

```
        docker compose up -d --no-build
        curl.exe -X POST $api/api/chaos/crash
        Start-Sleep 5
        docker compose ps -a        # api: Exited (1)
```

Верните `restart: unless-stopped` и выполните `docker compose up -d --no-build`.

## 5. Штатная остановка

```
        docker compose stop api
        docker compose logs api --tail 5 --no-log-prefix     # Application is shutting down...
        docker compose start api
```

В T2 появятся `kill` (`signal=15`), `die` с `exitCode=0` и `stop`. **Почему после `stop` Docker не поднял контейнер сам?**