# Неделя 14. Лаба 8. Итоговая: инцидент в сервисе задач

**Цель:** пройти полный цикл эксплуатации: выпуск через пайплайн → обнаружение деградации мониторингом → откат → исправление через PR → постмортем. **Время:** 120 мин + защита 10 мин. **Инструменты:** всё из курса: Visual Studio 2026, Git и GitHub, GitHub Actions и runner, Docker Desktop, Prometheus и Grafana, Dozzle, PowerShell 7. **Нужно:** runner запущен, мониторинг поднят, нагрузка `pwsh ./scripts/load.ps1` идёт.

## Легенда

Продакт просит «добавить аудит» при чтении списка задач. Разработчик делает это неудачно: запрос стал медленным. Тесты зелёные — функционально всё верно.

## 1. Плохой релиз

1. Issue: `feat: аудит чтения списка задач`.
2. Ветка от свежего `main`: `git switch main; git pull; git switch -c feat/<N>-audit`. В `src/DemoApi/Services/TaskService.cs` замените метод `ListAsync` целиком на:

```
        public async Task<IReadOnlyList<TaskItem>> ListAsync(int? projectId, TaskItemStatus? status, CancellationToken ct)
        {
            await Task.Delay(1500, ct); // "аудит"
            return await tasks.ListAsync(projectId, status, ct);
        }
```

3. Коммит, PR с `Closes #N`, зелёный `test`, ревью, merge.
4. Засеките **T0** — время merge.

## 2. Обнаружение

Не проверяйте сервис вручную — ждите алерта.

- Prometheus → *Alerts*: дождитесь `ApiSlowRequests` в **firing**. Засеките **T1**.
- На дашборде видно, что p95 вырос. Какое действие виновато, покажет запрос:

```
histogram_quantile(0.95, sum by (le, controller, action) (rate(http_request_duration_seconds_bucket{job="demo-api"}
[1m])))
```

- Определите, какой деплой это вызвал: `irm http://localhost:8080/` → `version` → коммит в GitHub.
- Найдите в коде метод, соответствующий `controller` и `action` из запроса, и строку, которая тормозит.

## 3. Откат

Сначала восстановить сервис, потом разбираться. Откатитесь на предыдущий `sha-` тег, как в лабе 5 п. 8 (или кнопкой, если сделали ⭐).

Дождитесь, пока алерт уйдёт из firing. Засеките **T2**.

## 4. Инцидент

Issue `incident: p95 > 1s после деплоя sha-xxxxxxx` с хронологией:

```
        ## Хронология
        - T0 hh:mm — merge PR #…, деплой sha-…
        - T1 hh:mm — ApiSlowRequests firing
        - hh:mm — найдено действие Tasks/List, связано с деплоем sha-…
        - T2 hh:mm — откат на sha-…, алерт снят

        ## Метрики
        - Время обнаружения (T1 − T0):
        - Время восстановления (T2 − T1):
```

Приложите скриншоты алерта и панели Grafana.

## 5. Исправление

1. На странице смерженного PR нажмите **Revert** → GitHub создаст PR с отменой. В описании укажите `Refs #`

`<incident>`.

2. Зелёный `test` → ревью → merge → пайплайн задеплоит исправленную версию **поверх отката**.
3. Проверьте: `version` = новый sha, p95 в норме, алертов нет.

## 6. Постмортем

Добавьте в issue инцидента комментарий и закройте issue:

```
## Что случилось
## Влияние (что видели пользователи, сколько времени)
## Первопричина
## Почему не поймали до прода
## Что сработало хорошо
## Действия (каждое — отдельный issue)
- [ ] …
```

Минимум одно действие должно **предотвращать повторение**. Например: smoke-проверка времени ответа `GET /api/tasks` в job `deploy` с порогом (`curl.exe -w "%{time_total}"`) или правило ревью «задержки и внешние вызовы в горячих запросах — только с обоснованием».

## 7. Релиз

```
git switch main; git pull
git tag v1.1.0; git push origin v1.1.0
```

Release notes: *Generate release notes* + ссылка на постмортем.

## Сдача и защита

Покажите преподавателю за 10 минут:

| Критерий | Баллы |
|---|---|
| Пайплайн `test → image → deploy` зелёный, `main` защищён, merge только через PR с CI | 20 |
| Плохой релиз обнаружен **алертом**, а не вручную; найдены действие, строка кода и деплой-виновник | 20 |
| Откат выполнен, время восстановления зафиксировано | 20 |
| Исправление через Revert PR, задеплоено пайплайном | 15 |
| Инцидент: хронология, скриншоты, постмортем с действиями | 15 |
| Ответы на 2 вопроса преподавателя по любой теме курса | 10 |

**Зачёт — от 70 баллов.**

Примеры вопросов на защите:

- Пройдите путь запроса `PATCH /api/tasks/{id}/status` от контроллера до БД. Где рождается 409?
- Контейнер рестартится с exit 137. Ваши действия?
- Почему деплоим `sha-` тег, а не `latest`?
- Чем readiness отличается от liveness, и что из них проверяет Docker?
- Где хранится пароль БД в CI и на вашей машине? Что будет, если его поменять?
- Что увидит cAdvisor и не увидит `/metrics` приложения?