Неделя 13. Лаба 7. Метрики и алерты: мониторинг сервиса задач

Часть 1 из 2: разделы «1. Стек мониторинга — через PR» — «4. PromQL».

Цель: поднять Prometheus, Grafana, cAdvisor и Dozzle, научиться читать метрики сервиса и контейнера, увидеть срабатывание алертов на реальных сбоях. **Время:** 120 мин. **Теория:** лекция 7. **Инструменты:** Prometheus, Grafana, cAdvisor, Dozzle (контейнеры), Docker Desktop, браузер, PowerShell 7.

Подготовка:

```
cd C:\dev\demo-api
$K = "C:\dev\kit"              # путь к kit из лабы 3
Test-Path $K\monitoring        # True
$api = 'http://localhost:8080'
docker compose ps              # api и db — healthy
```

Стек мониторинга займёт порты 9090, 3000, 8081 и 8888. Проверьте, что они свободны (пустой вывод — свободны):

```
Get-NetTCPConnection -State Listen -LocalPort 9090,3000,8081,8888 -ErrorAction SilentlyContinue |
  Select-Object LocalPort, @{ n = 'Process'; e = { (Get-Process -Id $_.OwningProcess).ProcessName } }
```

1. Стек мониторинга — через PR

```
git switch main; git pull
git switch -c feat/monitoring
Copy-Item -Recurse $K\monitoring, $K\scripts .
Get-ChildItem -Recurse monitoring, scripts | Unblock-File   # снять пометку «скачано из интернета»
docker compose -f monitoring/compose.yaml up -d
docker compose -f monitoring/compose.yaml ps
```

UI	URL	Что проверить
Prometheus	http://localhost:9090	*Status* → *Target health*: `demo-api`, `cadvisor`, `prometheus` — все **UP**
Grafana	http://localhost:3000	admin / admin (смену пароля можно пропустить), дашборд *DevOps course* → *DemoApi*
cAdvisor	http://localhost:8081	*Docker Containers* → `demo-api-1`
Dozzle	http://localhost:8888	Логи всех контейнеров в браузере

Прочитайте `monitoring/prometheus.yml` и `monitoring/alerts.yml`: откуда Prometheus берёт метрики и какие правила проверяет.

2. Сырые метрики

```
curl.exe -s $api/metrics | Select-String "^http_requests_received_total"
curl.exe -s http://localhost:8081/metrics | Select-String 'container_memory_working_set_bytes\{.*demo-api-1'
```

Найдите в метриках сервиса лейблы `controller` и `action`. Сопоставьте их с классами и методами в `src/DemoApi/Controllers`.

Зачем: за каждой строкой на дашборде стоит конкретный метод вашего кода.

3. Нагрузка

В **отдельном** терминале — фоновая нагрузка на всю лабу:

```
cd C:\dev\demo-api
pwsh ./scripts/load.ps1        # каждые 20 итераций печатает счётчик
```

Прочитайте скрипт: он создаёт проект, затем в цикле создаёт задачу, переводит её в `InProgress` и запрашивает список.

4. PromQL

Prometheus → *Query* → вкладка *Graph*. Выполните и запишите, что показывает каждый запрос:

```
up
sum by (controller, action) (rate(http_requests_received_total{job="demo-api"}[1m]))
histogram_quantile(0.95, sum by (le) (rate(http_request_duration_seconds_bucket{job="demo-api"}[1m])))
container_memory_working_set_bytes{name="demo-api-1"} / 1024 / 1024
sum by (name) (rate(container_cpu_usage_seconds_total{name=~"demo-.*"}[1m]))
```

Напишите сами:

1. Количество созданных задач за последнюю минуту (подсказка: `increase`, лейблы `controller="Tasks"`, `action="Create"`, `code="201"`).
2. Память `demo-api-1` в процентах от лимита.
3. p95 времени ответа **по каждому действию** (подсказка: `sum by (le, controller, action)`).
4. Частота ответов 4xx по действиям. Создайте ошибки и найдите их на графике:

```
        irm -Method Post "$api/api/projects" -ContentType application/json -Body '{"name":""}'          # 400
        $p = irm -Method Post "$api/api/projects" -ContentType application/json -Body "{`"name`":`"errors-$(Get-Random)`"}"
        $t = irm -Method Post "$api/api/tasks" -ContentType application/json -Body "{`"projectId`":$($p.id),`"title`":`"x`"}"
        irm -Method Patch "$api/api/tasks/$($t.id)/status" -ContentType application/json -Body '{"status":"Done"}'   # 409
```

Ошибки 400 и 409 PowerShell покажет красным — это ожидаемо.