Неделя 10. Лаба 5. CI/CD: тесты, образ и автодеплой сервиса

Часть 2 из 2: разделы «6. Первый деплой» — «⭐ Дополнительно». Первая часть лабы — в материалах предыдущей недели; перед продолжением выполните подготовку ниже.

Цель: пайплайн «тесты → образ в GHCR → автодеплой на свой Docker Desktop → smoke test», блокировка merge при красных тестах, откат. **Время:** 120 мин. **Теория:** лекция 5. **Инструменты:** GitHub Actions, GitHub Container Registry (GHCR), self-hosted runner, Docker Desktop, Visual Studio 2026, PowerShell 7.

Подготовка:

```
cd C:\dev\demo-api
$K = "C:\dev\kit"              # путь к kit из лабы 3
Test-Path $K\.github           # True
```

6. Первый деплой

Смержите PR (squash). *Actions* → запуск на `main`: **test → image → deploy**.

Проверка:

```
git switch main; git pull                # squash-merge создал на GitHub новый коммит
git log -1 --format=%h                   # его короткий sha
docker compose ps                        # образ ghcr.io/<you>/demo-api:sha-<тот же sha>
irm http://localhost:8080/               # version = sha-<тот же sha>
```

На странице репозитория → *Packages* → `demo-api` с тегами `sha-…` и `latest`.

7. Изменение через пайплайн

```
git switch main; git pull
git switch -c feat/info-message
```

1. В `src/DemoApi/Controllers/InfoController.cs` внизу файла добавьте в запись `InfoResponse` поле `string Message`.
2. В методе `Get` того же файла передайте `"hello from CI/CD"`.
3. `dotnet build` — компилятор подскажет, если что-то забыто.

PR → зелёный `test` → ревью → merge. Через пару минут `irm http://localhost:8080/` показывает новое поле и новый `version`. **Руками вы ничего не деплоили.**

8. Откат

Предыдущий образ уже есть локально — его скачивал runner:

```
docker image ls ghcr.io/<you>/demo-api     # найдите предыдущий sha-тег
$env:API_IMAGE = "ghcr.io/<you>/demo-api"  # в нижнем регистре
$env:API_TAG   = "sha-<предыдущий>"
docker compose up -d --no-build --wait
irm http://localhost:8080/                 # поля message нет, version старый
Remove-Item Env:API_IMAGE, Env:API_TAG
```

Засеките время отката: от первой команды до ответа `irm` со старым `version`. Вернуться на актуальную версию: *Actions → последний запуск на main → Re-run all jobs*.

Зачем уметь откатываться: при плохом релизе сначала восстанавливают сервис, потом разбираются. Лаба 8 построена на этом.

9. Релиз

```
git switch main; git pull
git tag v1.0.0
git push origin v1.0.0
```

В *Packages* появился тег `1.0.0`. Деплой на тег не запускается — сверьтесь с таблицей из лекции.

Сдача

- [ ] Ответы на 5 вопросов в описании PR
- [ ] Скриншот заблокированного merge при красном `test` и строка ошибки теста из лога
- [ ] Скриншот успешного запуска `test → image → deploy`
- [ ] `irm http://localhost:8080/` с `version`, равным sha последнего коммита в `main`
- [ ] Скриншот *Packages* с тегами `sha-…`, `latest`, `1.0.0`
- [ ] Выполненный откат и его время

☆ Дополнительно

1. Добавьте в workflow событие `workflow_dispatch` с input `tag` и job `rollback`, который деплоит указанный тег. Откат становится кнопкой в *Actions*.
2. Выведите в *Job summary* задеплоенный образ: `"Deployed ${env:API_IMAGE}:${env:API_TAG}" >> $env:GITHUB_STEP_SUMMARY`.
3. Публичный репозиторий: создайте Environment `local` с *Required reviewers* и добавьте `environment: local` в job `deploy`. Теперь деплой ждёт ручного одобрения.