CI/CD

Замечания и контекст

CI/CD должно демонстрировать **инженерную культуру**, автоматизацию, снижение ручных операций и стандартные практики DevOps.

Важно:
Студент **не обязан** иметь сложную инфраструктуру (Kubernetes, облака), но должен показать **осмысленный конвейер**, реальную автоматизацию и корректный деплой.

Минимум:

- CI (сборка + тесты) — обязательно
- CD (деплой) — приветствуется, но может быть выполнен в упрощённом варианте: деплой на сервер, контейнеризацию, публикация артефактов.

МИНИМАЛЬНЫЕ ТРЕБОВАНИЯ (на 5 баллов)

1. Документация

Обязательные разделы:

- **Диаграмма CI/CD конвейера**
  Сборка → тесты → сборка артефактов → деплой.
- **Описание используемого инструмента**
  GitHub Actions, GitLab CI, Azure DevOps, Jenkins, Bitbucket Pipelines или другое.
- **Что автоматизируется и почему**
  Обоснование этапов.
- **Перечень артефактов**
  Docker images, dist-файлы, release-артефакты, build artifacts.
- **Описание окружений**
  Dev / Test / Production (или эквивалент), даже если окружения эмулируются.

2. Архитектура CI/CD конвейера

Минимум должен включать:

CI (не менее 3 этапов)

1. **Проверка кода**
   - Lint (ESLint, TSLint, SonarLint)

    - Format check (Prettier)

2. **Сборка**
    - dotnet build, npm build, mvn package и т.п.
3. **Тестирование**
    - Unit-тесты — обязательно
    - e2e / integration — при наличии

CD (хотя бы один тип)

- Деплой в:
    - сервер (SSH + копирование артефактов)
    - Docker (сборка и запуск контейнера)
    - облако (Netlify, Vercel, Render, Fly.io, Azure, AWS, GCP)
    - локальный сервер с автоматическим обновлением build-артефактов

CD может быть **ручным (manual trigger)** — это допустимо.

3. Реализация CI/CD

Каждый pipeline должен:

- Автоматически запускаться:
    - при push,
    - при pull request,
    - вручную — если требуется.
- Использовать **кэширование** зависимостей:
  npm cache, dotnet nuget cache, Gradle cache и т. д.
- Формировать артефакты:
    - build output,
    - Docker image,
    - zip с артефактами.
- Иметь *хотя бы одну* форму деплоя:

    - Docker compose
    - GitHub Pages (для FE)
    - Vercel/Netlify
    - Self-hosted server
    - Azure WebApp / AWS ECS / GCP CloudRun

4. Качество и обработка ошибок

- Pipeline должен **падать** при ошибке сборки или тестов.
- Логи этапов должны быть понятными.
- В отчетах CI должны отображаться:
    - количество тестов
    - результаты линтера
- Должна быть обработка ошибок деплоя:
    - отмена деплоя при неудачном билде
    - проверки успешности запуска контейнера / сервера

5. Безопасность

Минимальные требования:

- Секреты (пароли, токены) должны храниться в:
    - GitHub Secrets,
    - GitLab Variables,
    - Jenkins credentials
    - или аналогичном безопасном хранилище
- Никаких секретов в коде или YAML-файлах
- Минимум один статический анализ безопасности:
    - npm audit
    - dotnet list package --vulnerable
    - pip audit

    - или SonarCloud (опционально)

6. Ограничения

Что НЕЛЬЗЯ:

- Загружать артефакты вручную
- Хранить секреты в репозитории
- CI/CD конфигурация в виде «одна команда внутри одного job»
- Отсутствие тестов
- Отсутствие документации pipeline
- Полностью ручной деплой без автоматизации

МАКСИМАЛЬНЫЕ ТРЕБОВАНИЯ (на 10 баллов)

Выполнить все минимальные требования **плюс** минимум два направления ниже.

Направление 1: Продвинутый CD

- Blue/Green deployment
- Canary releases
- Zero downtime deploy
- Rollback механизмы
- Автоматическая миграция базы в CD

Направление 2: Контейнеризация и оркестрация

- Docker multi-stage build
- Docker Compose с несколькими сервисами
- Kubernetes:
    - Deployments
    - Services
    - Ingress
- Автоматический деплой в Kubernetes через CI

Направление 3: Observability

- Интеграция с мониторингом:
    - Grafana
    - Prometheus
    - Elastic Stack
- Логи CI/CD в отдельном хранилище
- Auto-tagging контейнеров версиями
- Slack/Teams/Discord нотификации о статусе pipeline

Направление 4: Quality Gates

- SonarQube или SonarCloud анализ
- Покрытие тестами > 70%
- Code smells, bugs, vulnerabilities
- PR Quality rules:
    - обязателен успешный pipeline
    - обязателен code review

Направление 5: Infrastructure as Code

- Terraform
- Pulumi
- Ansible
- Автоматизация создания окружений
- Versioned environments