Формат Практика 3 — CI/DevSecOps: конвейер безопасной доставки ПО

Цель: автоматизировать сборку, тесты и проверки безопасности (Shift-Left) до слияния в основную ветку. Основной стек: Git, D cker, GitHub Acti ns/GitLab CI, Gitleaks (Secr ≣

Раздел 1. Базовый CI pipeline (Build + Test в контейнере)

Вариант A — GitHub Acti ns (.github/w rkfl ws/ci.yml)

```

```

Вариант B — GitLab CI (.gitlab-ci.yml)

```
''''
```

```
"                    "
```

Раздел 2. Интеграция quality gates (pre-c mmit в CI)

GitHub Acti ns (добавьте j b)

```

```

GitLab CI

```

```

Важно: пайплайн должен падать, если форматирование/линт не соответствует стандартам.

Раздел 3. DevSecOps проверки

3.1 Secret Scanning — Gitleaks

GitHub Actions

GitLab CI

3.2 SAST — Semgrep (OWASP T p 10)

GitHub Actions

GitLab CI

3.3 SCA/C ntainer Scan — Trivy

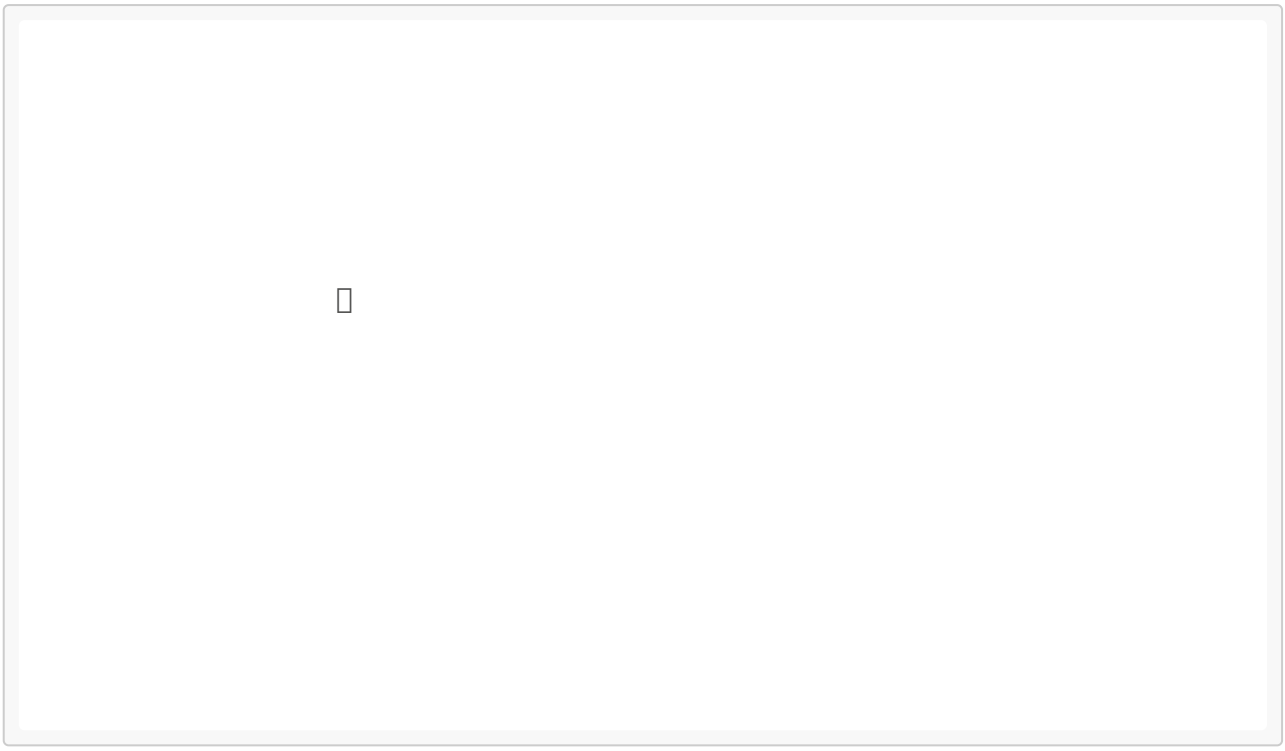
Политика: провал CI при HIGH/CRITICAL уязвимостях.

GitHub Actions (FS + Imag)

GitLab CI (FS + Imag)

Pa

GitHub Actions



2. Настроить политику fail- n для Trivy: HIGH,CRITICAL.
3. Сгенерировать SBOM (Cycl neDX) и приложить как артефакт.
4. Включить C de Scanning (GH) и загрузку SARIF (при использовании GitHub).
5. Настроить защиту ветки main и required status checks.
6. Подготовить REPORT.md с логами/скринами, ссылками на CI и анализом результатов.
7. Оформить ADR-0002 «Стратегия CI и инструменты DevSecOps».

Формат сдачи

- Ссылка на репозиторий (доступ на чтение преподавателю/жюри).
- Ветка: feature/practice3 → PR/MR → merge в main после «зелёного» CI.
- Обязательные файлы/директории:
  - .github/w rkfl ws/ci.yml или .gitlab-ci.yml
  - REPORT.md
  - d cs/adr/ADR-0002.md
- Артефакты/доказательства:
  - Ссылки на успешные/провальные прогонки CI с логами.
  - Скачанные отчёты (SARIF/JSON) и SBOM.
  - Скриншоты настроек Branch Pr tecti n/Required checks.

Критерии оценивания (Rubric)

+3 Репозиторий собирается, тесты стабильно проходят в контейнере (лог CI).

+3 Полный набор DevSecOps проверок: Gitleaks, Semgrep, Trivy (FS, Image), политики применены (HIGH/CRITICAL → fail).

+2 Отчётность: SARIF/JSON загружены/опубликованы, SBOM сформирован и приложен.

+2 Защита ветки включена, Required checks настроены (скрин/лог).

+2 ADR-0002 оформлен (контекст, решение, последствия +/–, критерии пересмотра, security c nsiderati ns).

+1 (Бонус) Интеграция GitHub C de Scanning или GitLab Security Dashb ard.

+1 (Бонус) Автогенерация SBOM по образу (Syft) и его проверка внешним валидатором.

–2..–4 За нерепродуцируемость/отсутствие логов/нечитаемые скриншоты.

–2 Секреты в репозитории, отсутствие сканирования секретов.

Tr ublesh ting

- D cker Buildx/кеш: обновите d cker/setup-buildx-acti n; при flake отключите кэширование временно.
- Таймауты/Rate limits Trivy: используйте локальный DB кэш (по умолчанию), повторный запуск ускорится.
- False p sitives Semgrep/Gitleaks: документируйте исключения (baseline/ign re), не отключайте правила без ADR.

- GitLab DIND права: """, совместимые версии d cker:service.
- SARIF не виден в GH: проверьте и путь .
- pre-c mmit не находит хуки: убедитесь в наличии .pre-c mmit-c nfig.yaml в корне.

Дополнительные материалы и чек-листы

Шаблон REPORT.md

Шаблон ADR-0002

Self-check (перед сдачей)

- CI успешно собирает образ и запускает тесты в контейнере.
- Pre-c mmit проверки проходят локально и в CI.
- [illegible]