Сценарий демонстрации CI на лекции (Gitea Actions)

К дисциплине: «Современные технологии разработки программного обеспечения» (СТРПО), неделя 9
Основа: лекция `../09l_л_Метрики_сложности.md`, блок CI (слайды 18–24)
Стенд: эта папка `Демо_CI_Gitea`; GitHub-версия — `../Демо_CI` (мини-проект «калькулятор скидок»)
Длительность: 10–12 минут; бонусные акты — ещё 4 минуты
Формат: живой показ на проекторе; реплики лектора — в блоках «Что говорить»

Карта актов ↔ слайды лекции

Акт	Что показываем	Слайд
0	Подготовка стенда (до лекции)	—
1	Зелёный локальный прогон, устройство пайплайна	«GitHub Actions: минимальный workflow»
2	Красный PR, лог падения, merge заблокирован	«Падай рано: правило 10 минут»
3	Лечение теста, зелёный прогон, merge, бейдж	«GitHub Actions: минимальный workflow»
4	Метрический гейт: `radon + xenon`	«Метрики в CI»
Бонус	Кеш, матрица, артефакты, «плохой» CI	«Стадии пайплайна»

Отличия Gitea-варианта

- workflow лежит в `.gitea/workflows/ci.yml`;
- прогоны выполняет act_runner;
- защита ветки — Status Check Contexts в Settings → Branches.

Акт 0. Подготовка (до лекции, 15–20 минут)

1. Создайте пустой репозиторий `ci-demo` на сервере Gitea.
2. Запушьте содержимое этой папки как корень репозитория:

```
git init
git add .
git commit -m "Стартовый каркас демо"
git branch -M main
git remote add origin https://gitea.example.com/OWNER/ci-demo.git
git push -u origin main
```

3. Замените `gitea.example.com` и `OWNER` в бейдже `README.md`.
4. Прогрейте кеш: дождитесь зелёного первого прогона act_runner.
5. Защита ветки: обязательные Status Check Contexts `lint`, `test`, `metrics`.

Чек-лист готовности: первый прогон зелёный; проверки `lint`, `test`, `metrics` видны в branch protection; кеш прогрет.

Акт 1. Зелёный старт (1–2 мин)

Покажите структуру: `app/`, `tests/`, `.gitea/workflows/ci.yml`. Запустите локально:

```
pytest -q
```

Ожидаемо: все тесты зелёные (сейчас 19). Откройте Actions — последний прогон на `main` зелёный.

Что говорить. «Локально зелёный — и в CI зелёный. CI не делает ничего волшебного: он повторяет те же шаги — checkout, Python, зависимости, pytest. Workflow — ваш локальный ритуал, записанный в YAML».

Покажите `ci.yml`: `on:` — триггер, `jobs.lint` — стиль, `jobs.test` — тесты с порогом покрытия, `jobs.metrics` — `radon` и `xenon`, `needs` — порядок и гейт.

Акт 2. Красный PR (3 мин)

```
git switch -c feature/vip-095
```

В `app/discounts.py` измените скидку VIP: `return total * 0.9` → `return total * 0.95`. Тесты не трогайте.

```
git add -A
git commit -m "VIP: скидка 0.95 (требование изменено)"
git push -u origin feature/vip-095
```

Откройте PR: `lint` — зелёный, `test` — красный, `metrics` — не запускается. В логе видно `assert ... pytest.approx(900.0)` и фактическое `950.0`. Кнопка merge заблокирована.

Что говорить. «Тест красный — merge нельзя. Это подарок: ошибка поймана за минуту. Красная main — stop-the-line. Упал только `test`; `lint` прошёл и не ждал тестов: быстрые проверки — первыми».

Акт 3. Лечим тест, мёржим (2 мин)

Спросите: что правильнее — вернуть код или обновить тест? Ответ: требование изменилось, значит меняется тест.

В `tests/test_discounts.py`: `pytest.approx(900.0)` → `pytest.approx(950.0)`.

```
git commit -am "Тест VIP приведён к новому требованию 0.95"
git push
```

Дождитесь зелёного прогона и нажмите Merge. Покажите бейдж в README.

Что говорить. «Зелёная галочка на PR — Definition of Done в действии: код, ревью, зелёный CI, merge. Без гейта DoD — просто список пожеланий».

Акт 4. Метрический гейт (2–3 мин)

Действия.

```
radon cc app -s -a
xenon --max-absolute C app/
```

Ожидаемо: `legacy_report.py:12 classify_order – C (17)` , гейт C зелёный.
Теперь скопируйте строгую заготовку:

```
Copy-Item workflows_заготовки/ci-metrics-strict.yml .gitea/workflows/ci.yml
git switch -c demo/strict-metrics
git add .
git commit -m "Демо: строгий метрический гейт"
git push -u origin demo/strict-metrics
```

В PR `metrics` красный: `xenon --max-absolute B` не пропускает ранг C.
PR закройте, верните основной workflow.

Что говорить. «Метрика работает, когда её нельзя игнорировать. Порог C пропускает функцию сложности 17; порог B — нет. Рост сложности становится видимым и блокирующим, а не темой для спора на ревью».

Бонусные акты (по времени)

- **Кеш**: сравните Duration первого и последнего прогонов; найдите `Cache hit` .
- **Матрица**: `Copy-Item workflows_заготовки/ci-matrix.yml ...` → 3.11 красный, 3.12/3.13 зелёные; PR не мёржить.
- **Артефакты**: `ci-artifacts.yml` → скачайте `coverage-report` .
- **Плохой CI**: `bad.yml` с `|| true` — тест падает, а проверка зелёная.

Если что-то пошло не так

- Первый прогон медленный — кеша ещё нет; прогрейте заранее.
- Branch protection не видит проверки — обновите страницу после первого прогона.
- Раннеры тормозят — покажите заранее сделанные прогоны по закладкам.

Уборка

```
git switch main
git checkout main -- .gitea/workflows/ci.yml
git push origin --delete feature/vip-095 demo/strict-metrics
```

Шпаргалка ответов

- **Почему отдельные job'ы?** Параллельность + `needs` задаёт порядок и гейт.
- **Что такое runner?** Чистая машина под прогон; состояние не сохраняется.
- **Можно ли доверять зелёному CI?** Только если проверки настоящие: без `|| true`.
- **Чем CI отличается от CD?** CI — интеграция и проверки; CD — доставка, отдельная тема.