

# GitHub Actions Workshop

# Автоматизация сборки и тестирования

с помощью GitHub Actions

Практический воркшоп

# Пару слов о себе







**Максим Гусев**

Lead SRE Observability Team в Dodo Engineering

11 лет в IT, последние 5 лет - SRE инженер

Читаю лекции о SRE и DevOps, автор нескольких курсов и воркшопов по DevOps

## Что мы изучим сегодня

-  Основы GitHub Actions и синтаксис workflow
-  Создание CI pipeline с нуля
-  Интеграция с фреймворками тестирования
-  Параллельное выполнение и матрицы
-  Работа с артефактами и зависимостями
-  Production-ready практики

# Проблема

## Без CI/CD:

Developer → Commit → Manual Tests → Manual Build → Manual Deploy

↓                      ↓                      ↓                      ↓

Забыли?            Не запустили?    Ошибка?            Сломался prod?

## С CI/CD:






Developer → Push → Автоматические тесты  → Автоматическая сборка  → Деплой 

**Цель:** Автоматизировать все повторяющиеся процессы

# Что такое GitHub Actions?

GitHub Actions – встроенная CI/CD платформа

## Преимущества:

-  Интеграция прямо в GitHub
-  Бесплатно для публичных репозиториев
-  Тысячи готовых actions в Marketplace
-  Мультиплатформенность (Linux, Windows, macOS)
-  Не нужно поднимать отдельную инфраструктуру

## Основные концепции

|          |  |
|----------|--|
| Workflow | # YAML файл с инструкциями                 |
| ↓        |  |
| Jobs     | # Параллельные или последовательные задачи |
| ↓        |  |
| Steps    | # Отдельные команды или actions            |
| ↓        |  |
| Actions  | # Переиспользуемые модули                  |

**Runner** – виртуальная машина, где это все выполняется

# Структура Workflow

```
name: CI Pipeline                                # Имя

on:                                              # Триггеры
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:                                           # Задачи
  test:
    runs-on: ubuntu-latest                    # Runner

    steps:                                     # Шаги
      - uses: actions/checkout@v4             # Action
      - run: pytest tests/                   # Команда
```



# Когда запускается Workflow?

## События (Triggers):

```
on:
  push:                # При push
  pull_request:        # При PR
  schedule:            # По расписанию
    - cron: '0 2 * * *' # Каждый день в 2:00
  workflow_dispatch:    # Ручной запуск
  release:             # При создании релиза
```

Можно комбинировать несколько триггеров

# Базовый Workflow

```
name: Hello World

on: push

jobs:
  greet:
    runs-on: ubuntu-latest

    steps:
      - name: Say hello
        run: echo "Hello from GitHub Actions!"
```

Это уже работающий workflow!

# Реальный пример - Python тесты

```
name: Python Tests

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - run: |
          pip install -r requirements.txt
          pytest tests/ -v
```

# Кэширование

## Проблема:

Каждый раз устанавливаем зависимости → 30-60 секунд

## Решение:

```
- uses: actions/cache@v4
  with:
    path: ~/.cache/pip
    key: ${ runner.os }-pip-${ hashFiles('requirements.txt') }
```

Экономия времени: до 10х быстрее!

# Matrix Strategy

## Задача:

Тестировать на Python 3.9, 3.10, 3.11, 3.12 + Ubuntu, Windows, macOS

## Решение:

```
strategy:
  matrix:
    os: [ubuntu-latest, windows-latest, macos-latest]
    python-version: ['3.9', '3.10', '3.11', '3.12']

runs-on: ${ matrix.os }
```

**Результат:** 12 jobs параллельно из 10 строк кода!

# Зависимости между Jobs

```
jobs:
  test:          # Запускается первым
    ...

  lint:          # Параллельно с test
    ...

  build:         # После test И lint
    needs: [test, lint]
    ...

  deploy:        # После build
    needs: build
    ...
```

**Workflow = граф зависимостей**

# Артефакты

## Сохранение результатов работы:

```
- name: Generate report
  run: pytest --html=report.html

- uses: actions/upload-artifact@v4
  with:
    name: test-report
    path: report.html
    retention-days: 30
```

Доступны для скачивания в GitHub UI

## Условное выполнение

```
jobs:
  deploy:
    if: github.ref == 'refs/heads/main' # Только main
    ...

steps:
  - name: Deploy to prod
    if: github.event_name == 'push'      # Только при push
    ...

  - name: Notify on failure
    if: failure()                        # Только при ошибке
    ...
```



# Секреты

## НЕ ДЕЛАЙТЕ ТАК:

```
- run: docker login -u admin -p MyPassword123
```

## ДЕЛАЙТЕ ТАК:

```
- uses: docker/login-action@v3  
  with:  
    username: ${ secrets.DOCKERHUB_USERNAME }  
    password: ${ secrets.DOCKERHUB_TOKEN }
```

Settings → Secrets → Actions

# Production Pipeline

```
jobs:
  lint:                # Проверка кода
  test-unit:           # Unit тесты
  test-integration:    # Integration тесты

  coverage:            # Покрытие тестами
    needs: [test-unit, test-integration]

  build:               # Сборка
    needs: [lint, test-unit, test-integration]

  deploy:              # Деплой
    needs: [coverage, build]
    if: github.ref == 'refs/heads/main'
```

# Best Practices

## ✅ DO:

- Версионите actions ( `@v4` , не `@main` )
- Кэшируйте зависимости
- Используйте secrets для чувствительных данных
- Давайте понятные имена
- Настройте уведомления

## ❌ DON'T:

- Не храните секреты в коде
- Не запускайте тяжелые задачи без необходимости
- Не игнорируйте failed builds

# Полезные фишки

## Переменные окружения:

```
env:  
  NODE_VERSION: '20'  
  APP_NAME: myapp
```

## Ручной запуск:

```
on:  
  workflow_dispatch:  
    inputs:  
      environment:  
        required: true
```

## Полезные фишки, которые не влезли в прошлый слайд :)

### Отладка:

```
– name: Debug  
  run: echo "Context: ${toJson(github)}"
```

# Интеграции

GitHub Actions легко интегрируется с:

-  **Docker** - сборка и push образов
-  **Cloud Providers** - AWS, Azure, GCP
-  **Monitoring** - Datadog, New Relic
-  **Notifications** - Slack, Telegram, Email
-  **Package Registries** - npm, PyPI, Maven
-  **Security** - Snyk, Dependabot

# Стоимость

Публичные репозитории:

БЕСПЛАТНО без ограничений 🎉

Приватные репозитории:

- Free: 2,000 минут/месяц
- Team: 3,000 минут/месяц
- Enterprise: 50,000 минут/месяц

Linux runner: 1x

Windows runner: 2x

macOS runner: 10x

## Альтернативы

| Инструмент     | Плюсы                     | Минусы               |
|----------------|---------------------------|----------------------|
| GitHub Actions | Встроен в GitHub, простой | Молодой              |
| GitLab CI      | Мощный, гибкий            | Сложнее              |
| Jenkins        | Очень гибкий              | Нужна инфраструктура |
| CircleCI       | Быстрый                   | Платный              |
| Travis CI      | Простой                   | Менее популярен      |

Выбор зависит от ваших требований









# Практическое задание, для тех кто морально готов

## Ваша задача:

1. Форкните репозиторий
2. Добавьте новую функцию в калькулятор
3. Напишите тесты
4. Создайте Pull Request
5. Убедитесь, что CI прошел успешно

# Что дальше?

## Продвинутые темы:

-  Reusable workflows
-  Composite actions
-  Self-hosted runners
-  OIDC authentication
-  GitHub Packages
-  Deployment environments

# Что почитать?

## Ресурсы:




- [docs.github.com/actions](https://docs.github.com/actions)
- [github.com/marketplace](https://github.com/marketplace)
- [Awesome Actions](#)

# Итоги

## Мы изучили:

- ✓ Основы GitHub Actions
- ✓ Создание CI/CD pipeline
- ✓ Тестирование и проверка кода
- ✓ Параллелизация с matrix
- ✓ Работа с артефактами
- ✓ Production best practices

## Теперь вы можете:

-  Автоматизировать тестирование
-  Настроить CI/CD для своих проектов
-  Улучшить качество кода

Вопросы?

Спасибо за внимание! 🎉

Links:



Telegram:

@fadeinflames

Материалы воркшопа:

GitHub:

<https://github.com/fadeinflames/github-actions-workshop>

## Бонус: Быстрый старт

```
# 1. Создайте директорию
mkdir -p .github/workflows

# 2. Создайте файл ci.yml
cat > .github/workflows/ci.yml << 'EOF'
name: CI
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: echo "Hello CI/CD!"
EOF
```

## Бонус: Быстрый старт - продолжение

```
# 3. Коммит и push  
git add .github/  
git commit -m "Add CI"  
git push
```

```
# 4. Смотрите результат на github.com
```

## Бонус: Чек-лист CI/CD

- ☐ Автоматические тесты при каждом коммите
- ☐ Проверка качества кода (linting)
- ☐ Измерение покрытия тестами
- ☐ Сборка артефактов
- ☐ Автоматический деплой в staging
- ☐ Ручной деплой в production
- ☐ Уведомления о падениях
- ☐ Мониторинг времени выполнения

**Начните с малого, улучшайте постепенно!**