

# Профессия DevOps-инженер

**Константин Брюханов**

Lead DevSecOps

# Введение в DevOps

# Константин Брюханов



Lead DevSecOps

Ведущий вебинаров

Ментор и эксперт в стартап-акселераторе allthewayup

Помогаю открыть направление DevOps в магистратуре ИТМО

Пишу диссертацию о проблемах внедрения DevOps

# Что такое DevOps

# IT-Компании



## **Development - разработка**

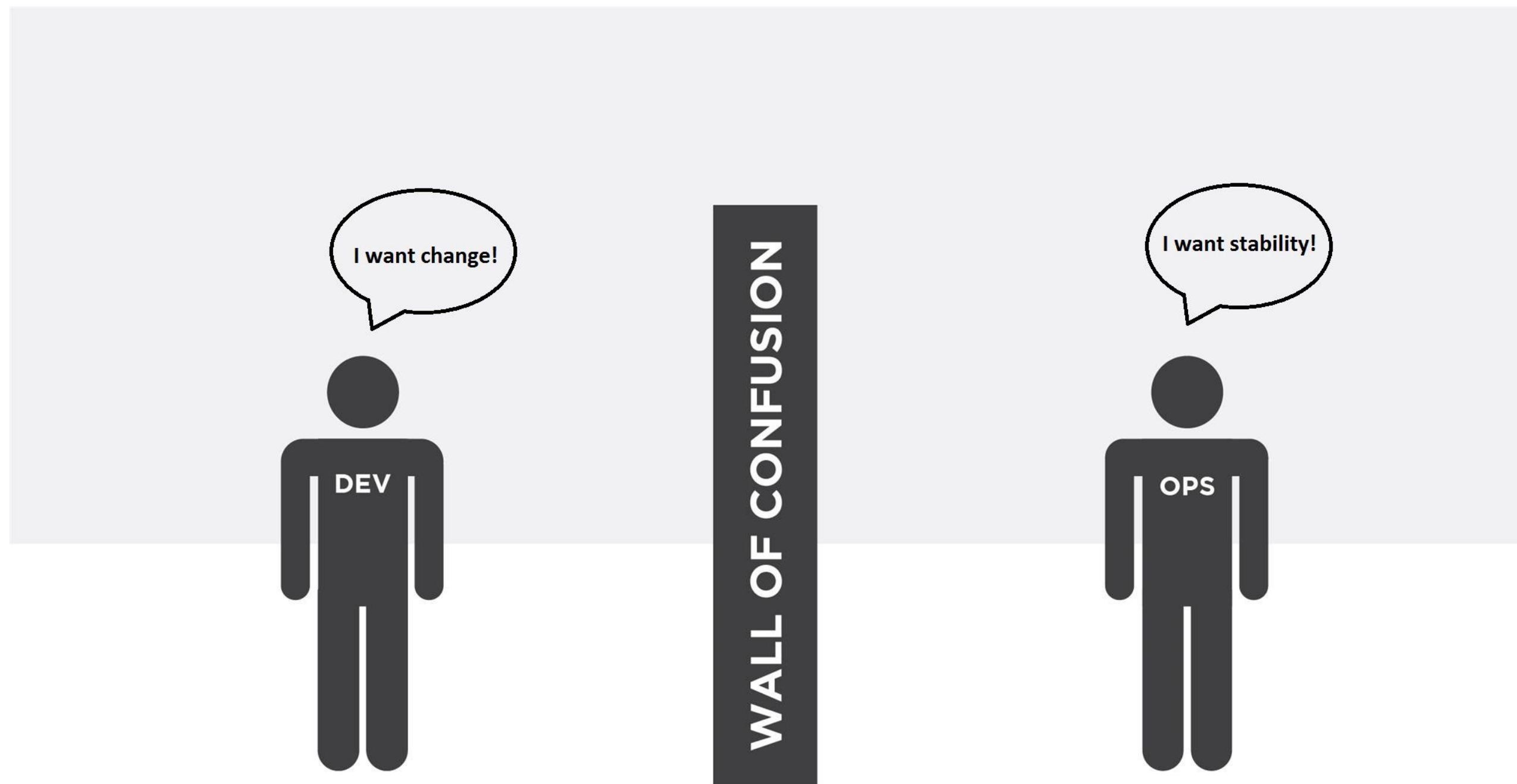
- разрабатывают продукт и новый функционал
- хотят как можно скорее его увидеть на серверах



## **Operations - эксплуатация**

- отвечают за стабильность серверов
- хотят, чтобы серверы реже выключались
- хотят, чтобы приложение было доступно клиенту бесперебойно

# Хронический конфликт



# Слайд с компакт-дисками



**Инфраструктура как код**

**Infrastructure as a code (IaC)**



# Методология

- максимально беспрепятственное взаимодействие внутри продуктовой команды
- повышение эффективности команды
- выкладывание новых фишки чуть ли не моментально и на лету
- каждый человек команды вовлечен в продукт

# Development + Operations



# Кто такой DevOps-специалист



**Так как же его можно называть?**

**Девопс-инженер**



**Девопс**



# Основные задачи

# Основные задачи

- Администрирование серверов

# Основные задачи

- Администрирование серверов
- Обеспечение их отказоустойчивости

# Основные задачи

- Администрирование серверов
- Обеспечение их отказоустойчивости
- Мониторинг и реагирование на проблемы



# Основные задачи

- Администрирование серверов
- Обеспечение их отказоустойчивости
- Мониторинг и реагирование на проблемы
- Автоматизация решения проблем

# Основные задачи

- Администрирование серверов
- Обеспечение их отказоустойчивости
- Мониторинг и реагирование на проблемы
- Автоматизация решения проблем
- Автоматизация выкладки приложений на сервера

# Основные задачи

- Администрирование серверов
- Обеспечение их отказоустойчивости
- Мониторинг и реагирование на проблемы
- Автоматизация решения проблем
- Автоматизация выкладки приложений на сервера
- Организация процессов работы с кодом

# Основные задачи

- Администрирование серверов
- Обеспечение их отказоустойчивости
- Мониторинг и реагирование на проблемы
- Автоматизация решения проблем
- Автоматизация выкладки приложений на сервера
- Организация процессов работы с кодом
- Описание серверов, сервисов и сети в виде кода

# Далее на курсе

- Использовать Docker
- Работать с Ansible
- Конфигурировать Gitlab CI и Jenkins
- Экономить деньги компании

# Первая задача

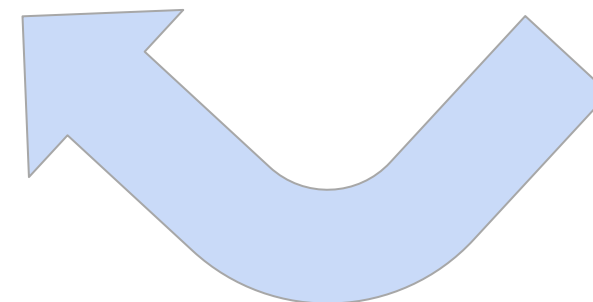
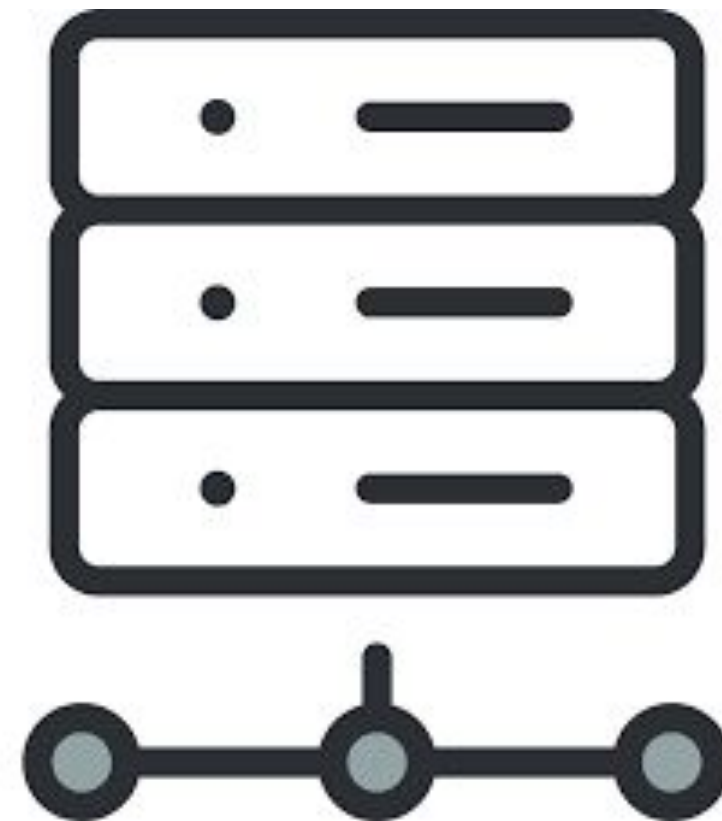
# Наш первый день на новой работе

# Проблема

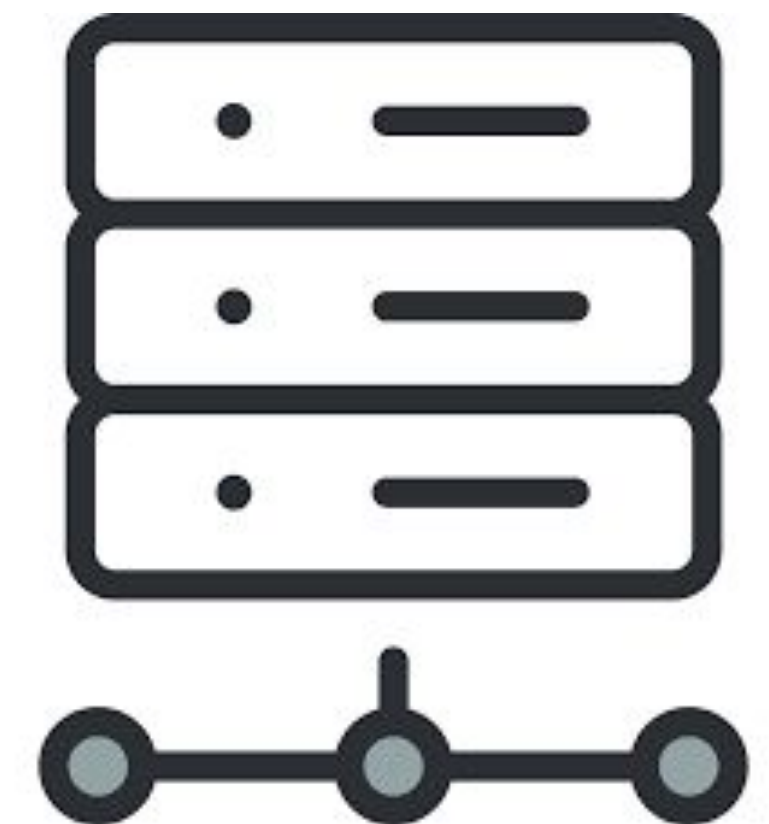
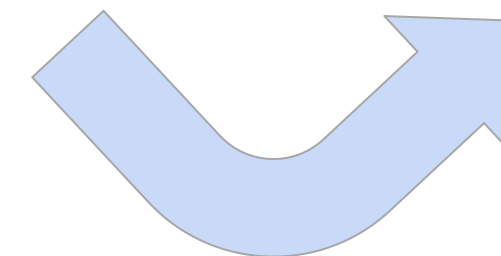
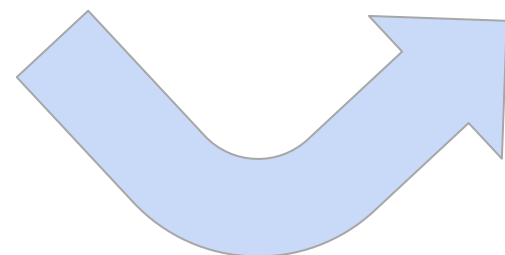
- аутсорс-разработчик компании занимается развёртыванием продукта на сервер вместо сисадминов
- на проекте не было команды эксплуатации



# Проблема



# Решение



# Задача

- Забирать новый код от аутсорсеров
- Доставлять его к нам в develop-ветку
- Создавать merge-реквест

# Шаг 1

- **Склонировать главный репозиторий**

# Шаг 2

- Склонировать главный репозиторий
- **Получить код от аутсорсеров**

# Шаг 3

- Склонировать главный репозиторий
- Получить код от аутсорсеров
- **Сделать ветку develop**

# Шаг 4

- Склонировать главный репозиторий
- Получить код от аутсорсеров
- Сделать ветку develop
- **Проверяем наличие кода**

# Шаг 5

- Склонировать главный репозиторий
- Получить код от аутсорсеров
- Сделать ветку develop
- Проверяем наличие кода
- **Доставить код в наш собственный репозиторий**



# Автоматизация решения

# Автоматизация решения

- Написать скрипт, выполняющий команды

# Автоматизация решения

- Написать скрипт, выполняющий команды
- Добавить его в планировщик

# Автоматизация решения

- Написать скрипт, выполняющий команды
- Добавить его в планировщик
- Выбрать приемлемое время

# Практическое задание

Повторите шаги, показанные в уроке:

- Склонируйте главный репозиторий
- Получите код от аутсорсеров
- Сделайте ветку develop
- Проверьте наличие кода
- Доставьте код в ваш собственный репозиторий

# **Вводная к автоматической сборке и тестированию**

# Нужны автоматические проверки

- модульные тесты пишут разработчики
- статический анализ кода выполняют специальные программы
- существует много видов тестирования, у каждого своя задача

# Тестирование кода Python

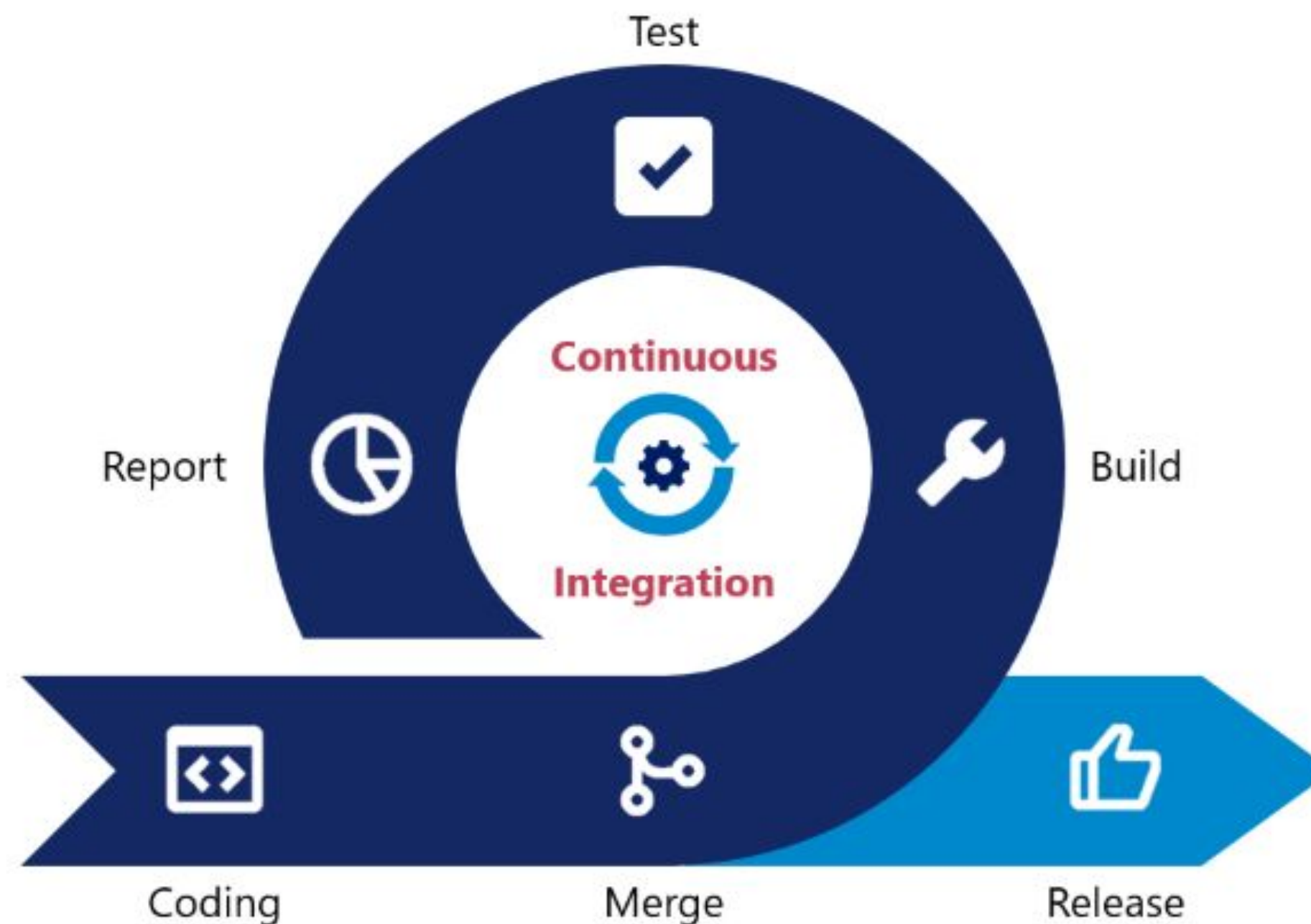
- **pytest** - модульные тесты
  - **flake8** и **pylint** - статический анализ
  - **mypy** - проверка типов переменных
- 
- экономят время на ненужную доставку кода
  - помогают программисту в поиске ошибок



# Непрерывная интеграция

## CI - Continuous Integration

- написание кода
- слияние
- сборка и тестирование
- отчеты
- выпуск новой версии



# GitLab CI

## Update .gitlab-ci.yml

🕒 4 jobs for `master` in 2 minutes and 36 seconds (queued for 2 seconds)

🚩 latest

🔗 3b31b248 ... 🔗

# Pipeline

Pipeline Jobs 4 Failed Jobs 1

# Stage

### Static Analysis

✅ flake8 🔄

✅ mypy 🔄


⚠️ pylint 🔄

### Test

✅ unit\_test 🔄

# Job

# Включение пайплайнов

 Settings

General

Integrations

Webhooks

Repository

CI / CD

Allow users to make copies of your repository to a new project

☒ Everyone With Access

**Pipelines**

Build, test, and deploy your changes

☒ Everyone With Access

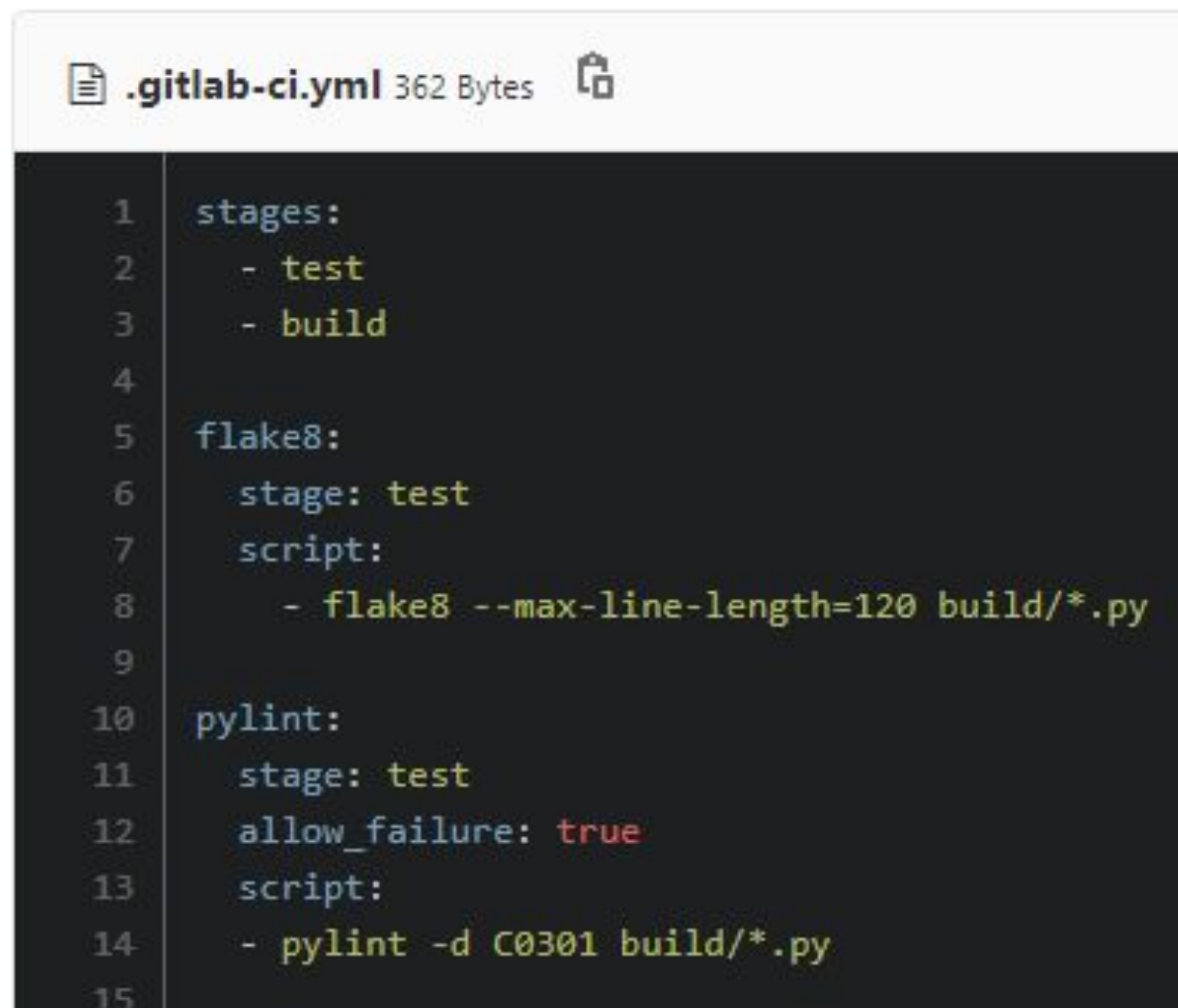
**Container registry ?**

Every project can have its own space to store its Docker images

Note: the container registry is always visible when a project is public

☒

# gitlab-ci.yml



The image shows a code editor window with a file named `.gitlab-ci.yml` (362 Bytes). The file contains the following YAML configuration:

```
1  stages:
2    - test
3    - build
4
5  flake8:
6    stage: test
7    script:
8      - flake8 --max-line-length=120 build/*.py
9
10 pylint:
11   stage: test
12   allow_failure: true
13   script:
14     - pylint -d C0301 build/*.py
15
```

# Что нам дал Gitlab CI

- **Автоматизировали** часть работы
- **Помогли** программистам в отладке кода

# Триггерные события

- **Автоматизировали** запуск при получении нового кода
- Запуск при **мердж-реквесте**
- Запуск в **специальной ветке** или **тэге**
- **Ручной запуск**

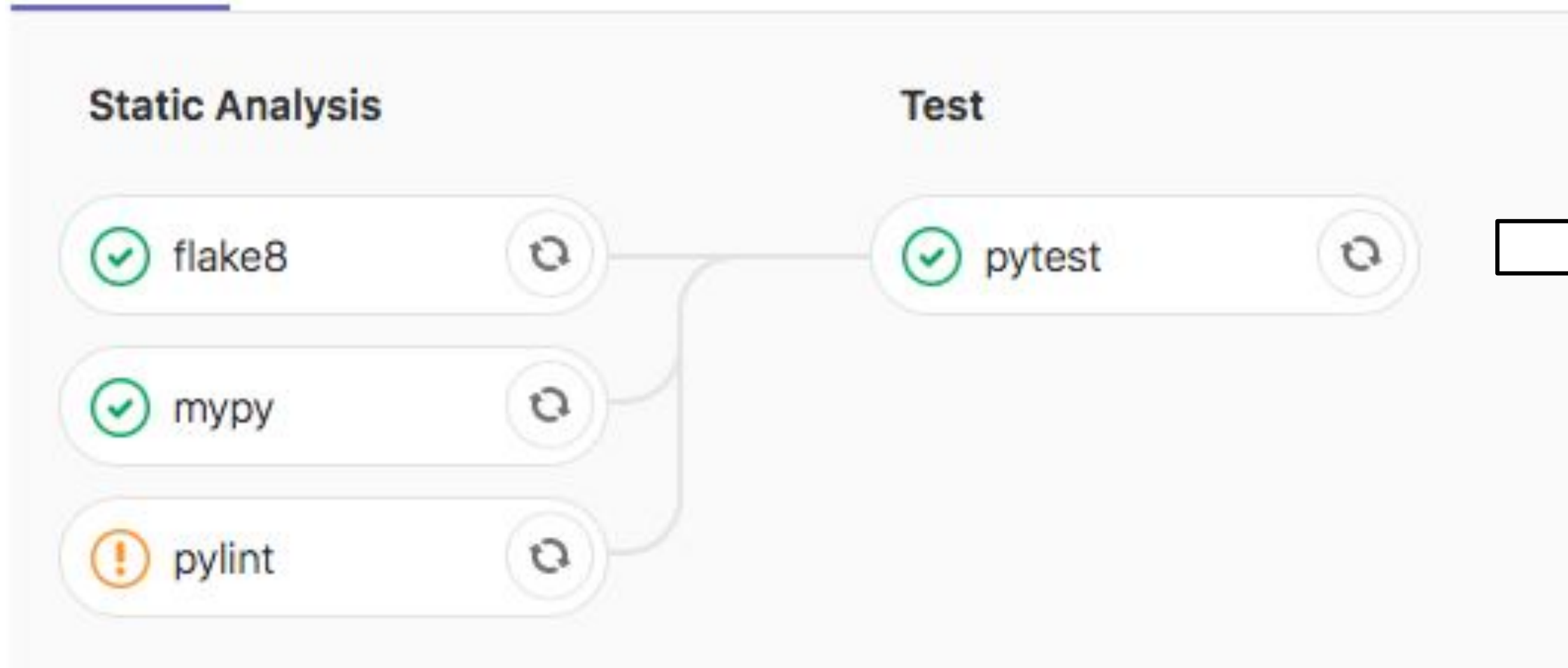
# Триггер на запуск в нужных ветках

```
1 build:
2   stage: build
3   script:
4     - ./build
5   only:
6     - master
7     - /^release_[0-9]+(?:?:.[0-9]+)+$/
8
```



# Что дальше

Pipeline Jobs 4 Failed Jobs 1





# Непрерывное развертывание

**CD - continuous delivery и continuous deployment**

- **написание кода**
- **слияние**
- **сборка и тестирование**
- **выпуск новой версии**
- **автоматическое развертывание в тестовой среде**
- **ручное/автоматическое развертывание в продакшене**

# Пример



# Что будет дальше

- Разберёмся, как нам доставлять код на сервера
- Как автоматически создавать сервера, когда они нужны
- Каким образом можно тестировать запущенный проект

# Домашняя работа

# Домашняя работа

Сделайте то же самое самостоятельно, в своем репозитории:

- добавьте файл с инструкциями для Gitlab CI
- опишите этап автоматического тестирования
- опишите этап сборки приложения
- добавьте дополнительную задачу, которая запустит абстрактный тест на наше приложение после сборки