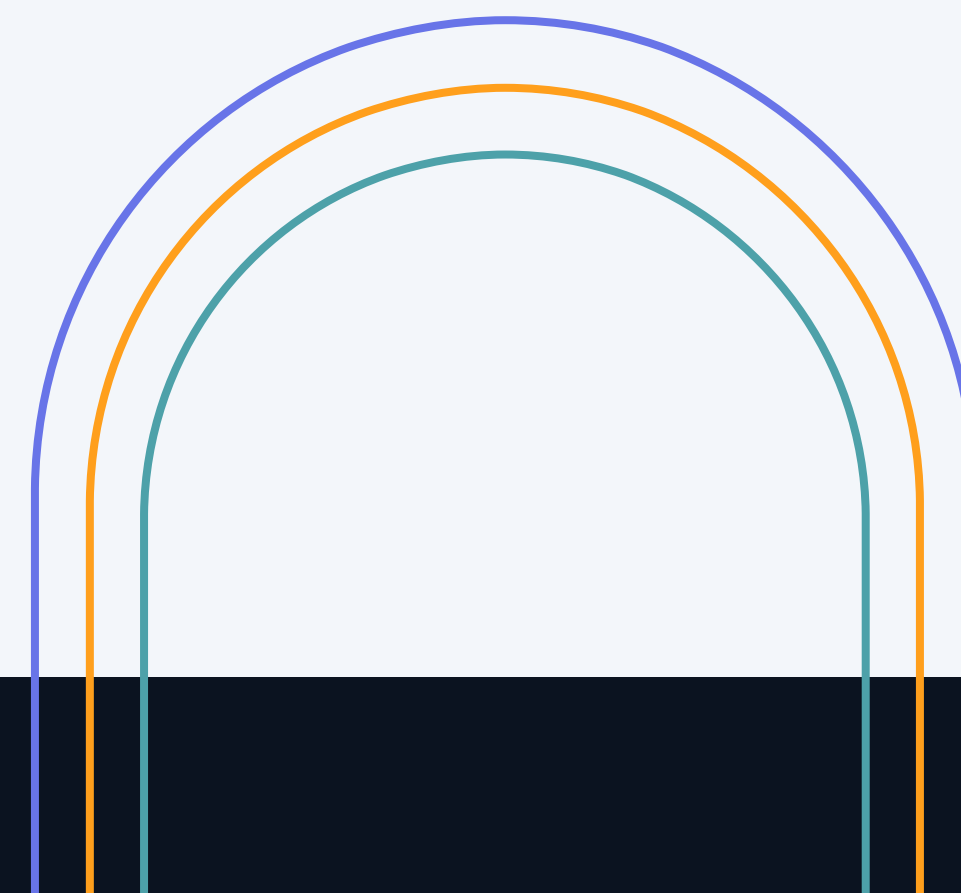
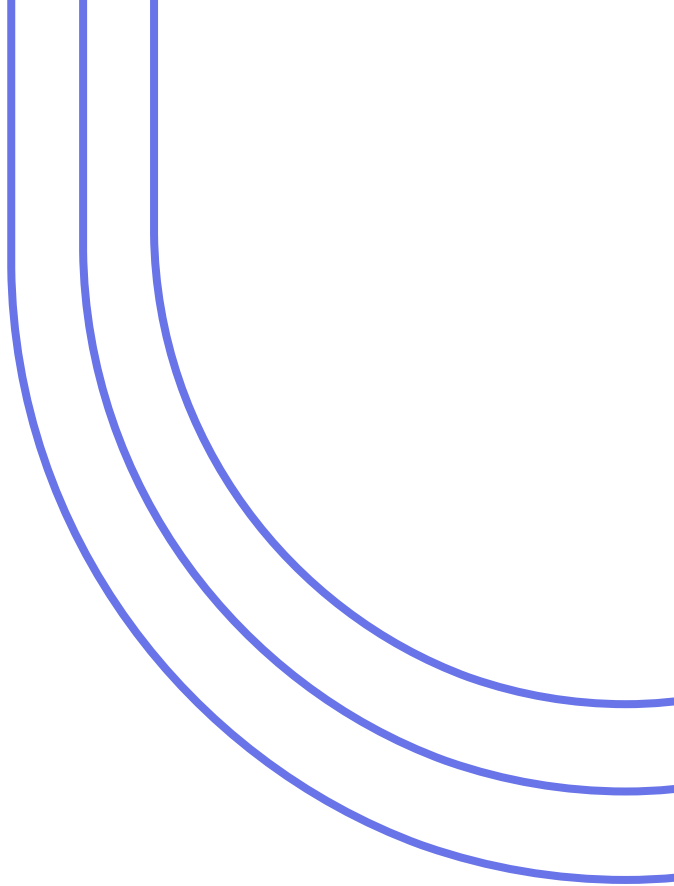


Лекция 10. CI & CD



- 
- 01.** Зачем нужен CI/CD?
 - 02.** Непрерывная интеграция
 - 03.** Непрерывная доставка
 - 04.** CI/CD: принципы, внедрение,
инструменты
 - 05.** DevOps

Содержание



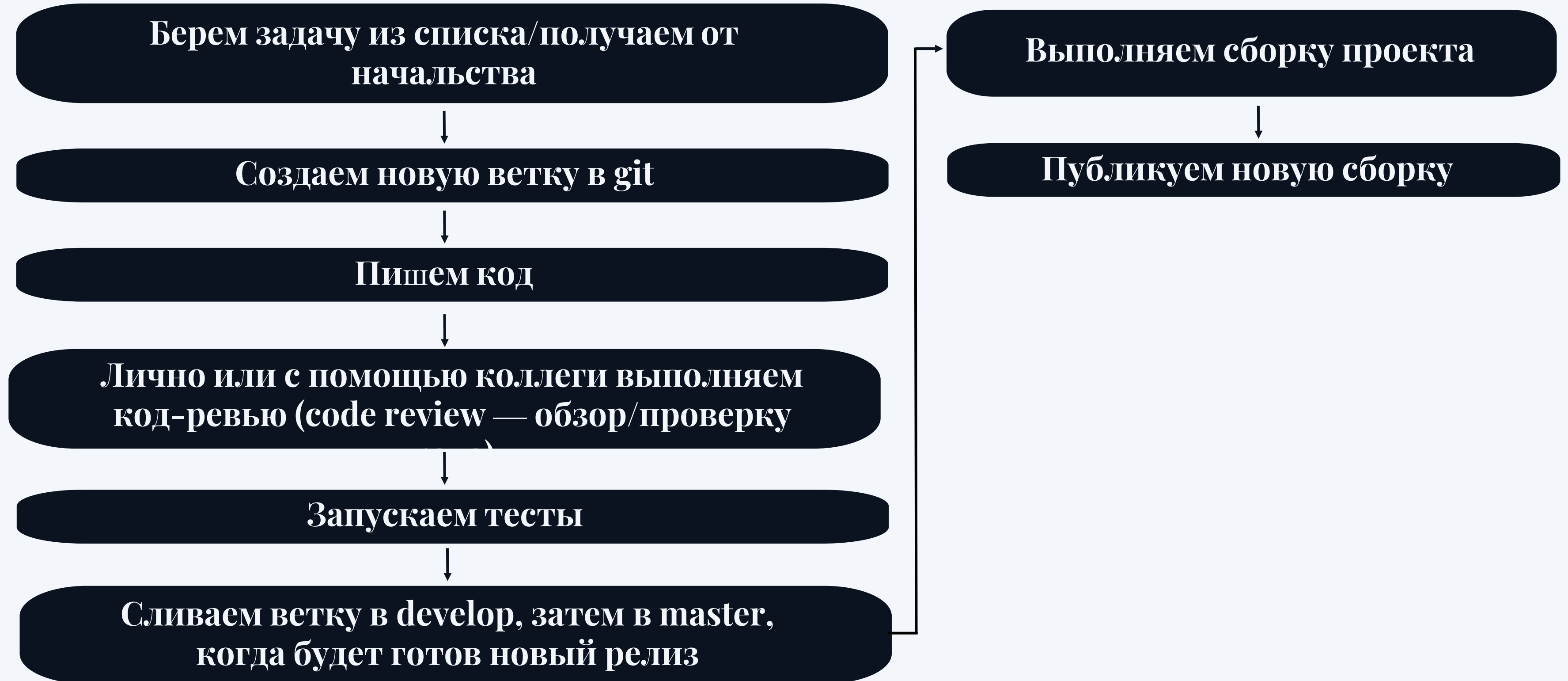
01.



Зачем нужен CI/CD?

01. Зачем нужен CI/CD?

Классический процесс решения задачи,
подходящий для большинства компаний:



**Важно затрачивать как можно меньше
времени на развёртывание.**

**Для автоматизации этого процесса и
придумана концепция CI/CD.**

02.



Непрерывная интеграция

02. Непрерывная интеграция

Continuous Integration (CI, непрерывная интеграция/доставка) — это практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки несколько раз в день и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.

Переход к непрерывной интеграции позволяет:

снизить
трудоемкость
интеграции

сделать её более
предсказуемой за
счет раннего
обнаружения и
устранения ошибок и
противоречий

сократить
стоимость
исправления
дефекта, за счёт
раннего его
выявления

Непрерывная
интеграция впервые
названа и
предложена Гради
Бучем в 1991 году

02. Непрерывная интеграция



Требования к проекту, в который необходимо встроить CI

Исходный код и всё, что необходимо для сборки и тестирования проекта, хранится в репозитории системы управления версиями (Git, Subversion, Mercurial и т.д.)

Операции копирования из репозитория, сборки и тестирования всего проекта автоматизированы и легко вызываются из внешней программы

02. Непрерывная интеграция





Преимущества

- проблемы интеграции выявляются и исправляются быстро, что оказывается дешевле
- немедленный прогон модульных тестов для свежих изменений
- постоянное наличие текущей стабильной версии вместе с продуктами сборки — для тестирования, демонстрации, и т. п.
- немедленный эффект от неполного или неработающего кода приучает разработчиков к работе в итеративном режиме с более коротким циклом

Недостатки

- затраты на поддержку работы непрерывной интеграции
- потенциальная необходимость в выделенном сервере под нужды непрерывной интеграции
- необходимо обратить внимание на скорость CI: разработчики должны получать результат CI максимум за 10 минут, иначе продуктивность падает из-за потери концентрации и частого переключения между задачами

03.



Непрерывная доставка

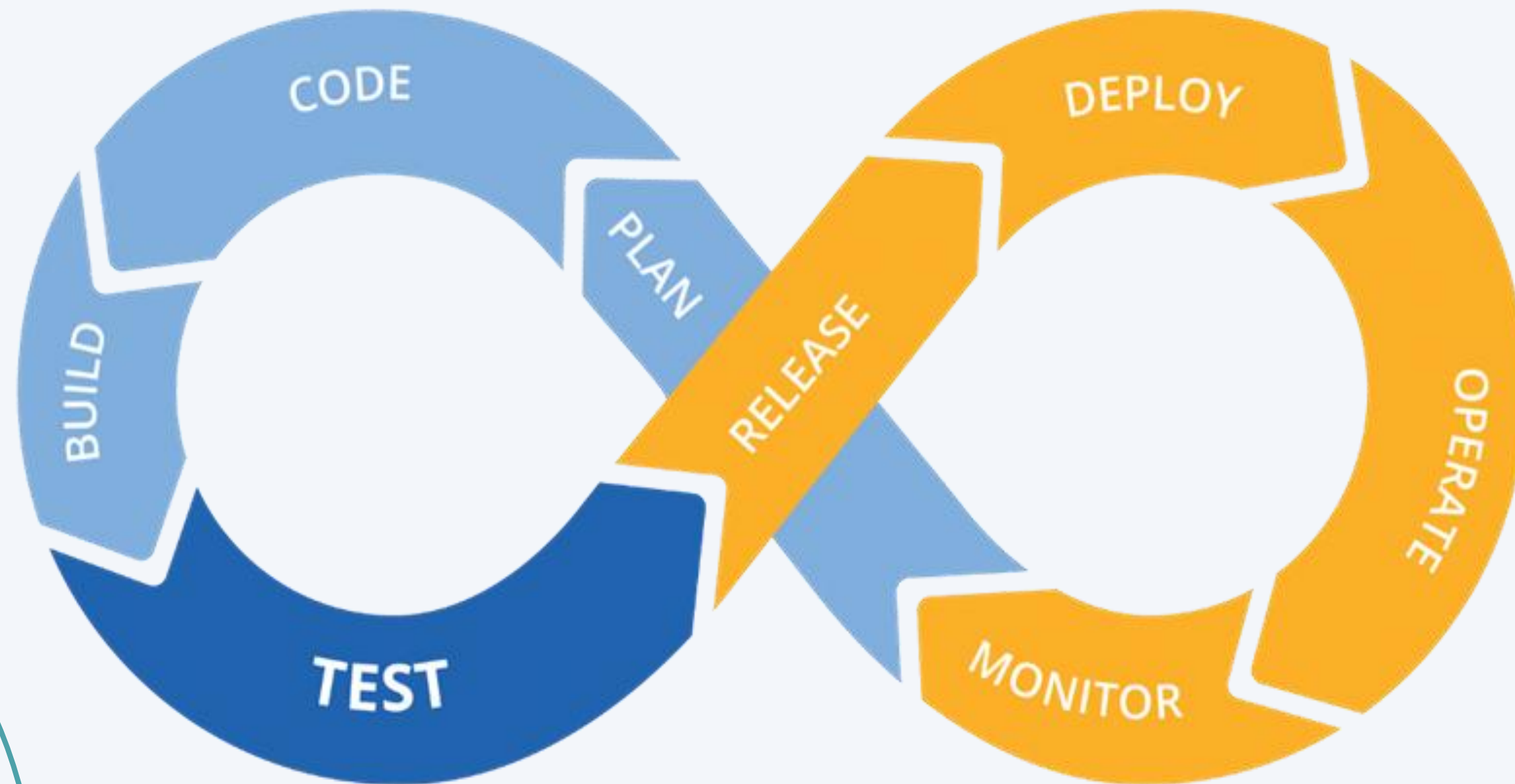
03. Непрерывная доставка

Непрерывная доставка (Continuous Delivery, CD) – это практика автоматизации всего процесса релиза ПО. Идея заключается в том, чтобы выполнять CI, плюс автоматически готовить и вести релиз к промышленной среде.

Как правило, в процессе непрерывной доставки требуется выполнять вручную как минимум один этап: одобрить развертывание в промышленную среду и запустить его.

Непрерывное развертывание располагается «на уровень выше» непрерывной доставки (CI). В данном случае все изменения, вносимые в исходный код, автоматически развертываются в продакшн. Как правило, задача разработчика сводится к проверке запроса на включение (pull request) от коллеги.

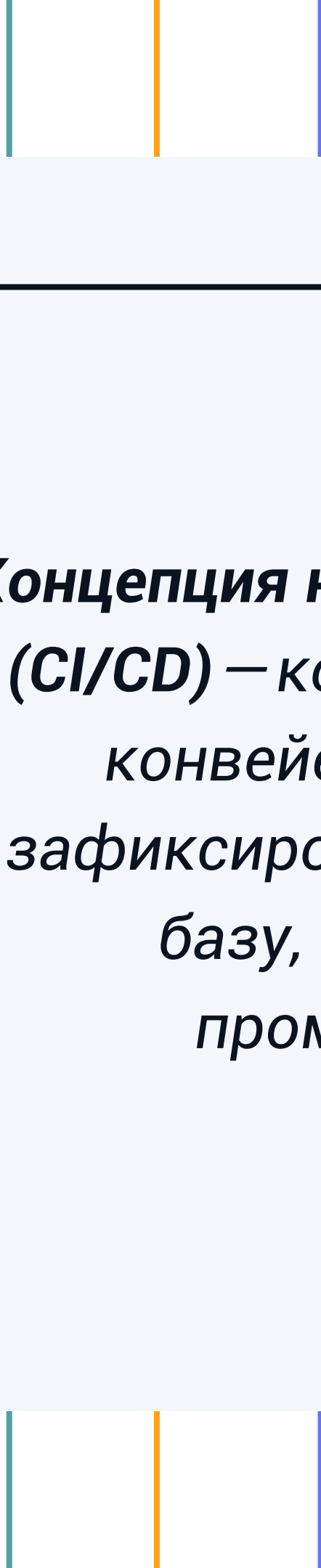
Общая схема CI & CD




04.

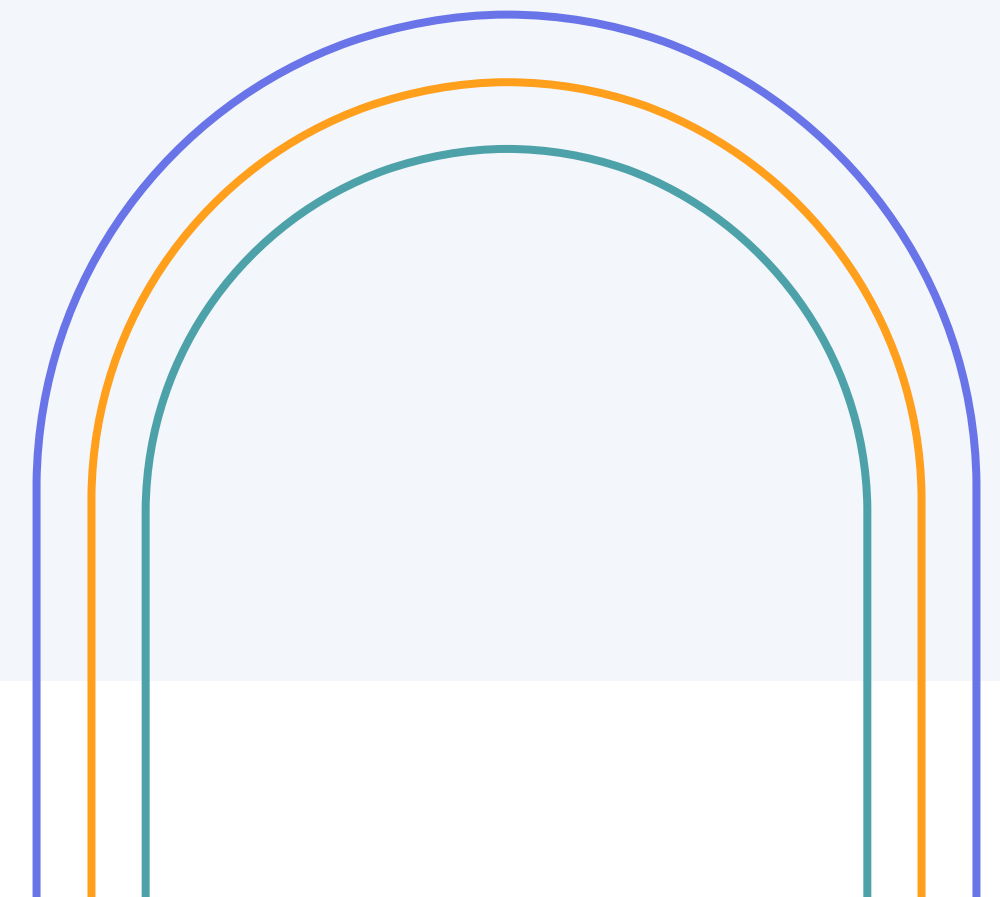


CI/CD: принципы, внедрение, инструменты





Концепция непрерывной интеграции и доставки (CI/CD) — концепция, которая реализуется как конвейер, облегчая слияние только что зафиксированного кода в основную кодовую базу, так и установку приложения в промышленную/тестовую среду.



Принципы CI&CD

1

разделение ответственности
заинтересованных сторон
(разработчики, QA, аналитики, Dev-
Ops инженеры, владельцы продукта)

2

снижение риска (каждый отвечает
за свою роль и пытается снизить
риски для бизнеса в своей работы)

3

короткий цикл обратной связи

Внедрение

Реализации среды в CI/CD (чаще всего
заводятся отдельные ветви в
репозитории)

Инструменты

локальные: GitLab CI, TeamCity, Bamboo, GoCD
Jenkins, Circle CI;

облачные: BitBucket Pipelines, Heroku CI, Travis,
Codeship, Buddy CI, AWS CodeBuild)

05.



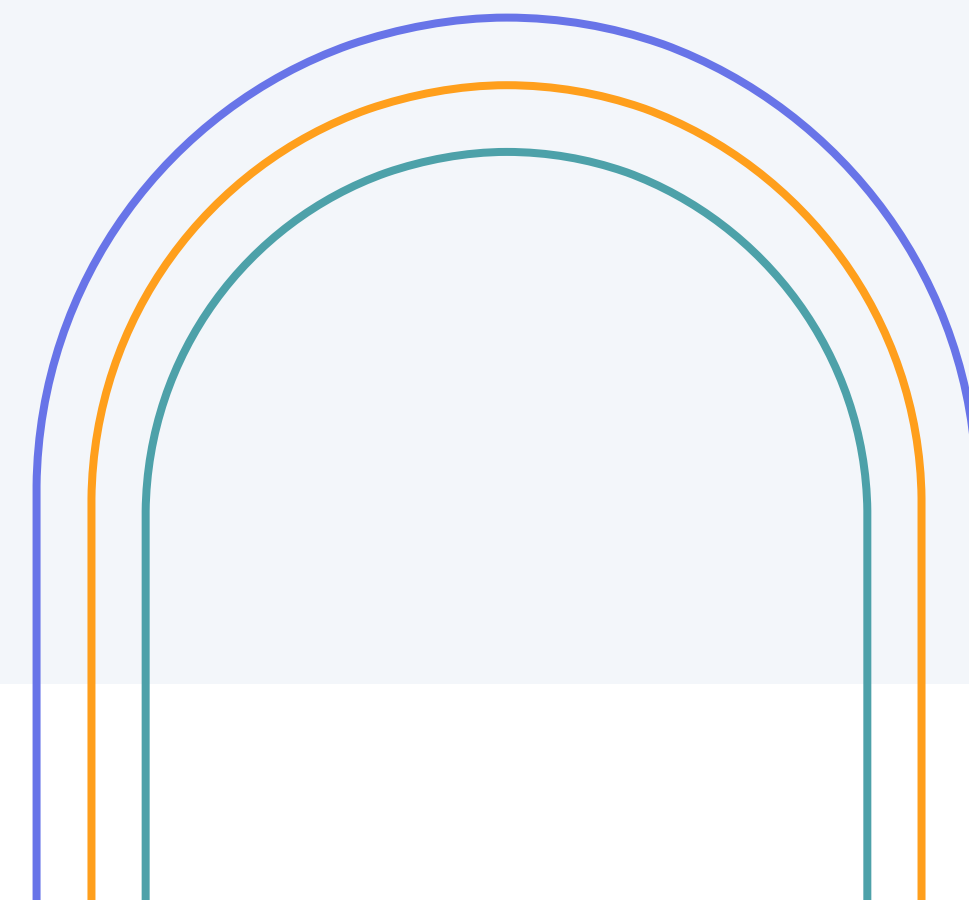
DEVOPS



Изначально DevOps не имел ничего общего с конкретной должностью в организации. Многие по-прежнему заявляют, что DevOps - это культура, а не профессия, согласно которой коммуникация между разработчиками и системными администраторами должна быть налажена максимально тесно.

Естественным путём DevOps из разряда “культуры” и “идеологии” переместился в разряд “профессии”. От DevOps-инженера ждут: (какие есть предположения, исходя из того, что DevOps – это *development + operations*):

- системное администрирование
- программирование
- использование облачных технологий
- автоматизация крупной инфраструктуры



Цели DevOps

1

Сокращение времени для выхода на рынок

2

Снижение частоты отказов новых релизов

3

Сокращение времени выполнения исправлений

4

Уменьшение количества времени на восстановления (в случае сбоя новой версии или иного отключения текущей системы)

Преимущества

Компании, которые используют DevOps, сообщили о значительных преимуществах, в том числе:

- значительном сокращении времени выхода на рынок;
- улучшении удовлетворенности клиентов;
- улучшении качества продукции;
- более надежных выпусках;
- повышении производительности и эффективности
- увеличении способности создавать правильный продукт путем быстрого экспериментирования

Однако всегда стоит взвешивать целесообразность решения.

DevOps и архитектура



Чтобы эффективно использовать DevOps, прикладные программы должны соответствовать набору архитектурно значимых требований, таких как:

- возможность автоматизированного развертывания;
- изменяемость;
- тестируемость;
- возможность мониторинга (система постоянного наблюдения за явлениями и процессами, проходящими в приложении).

