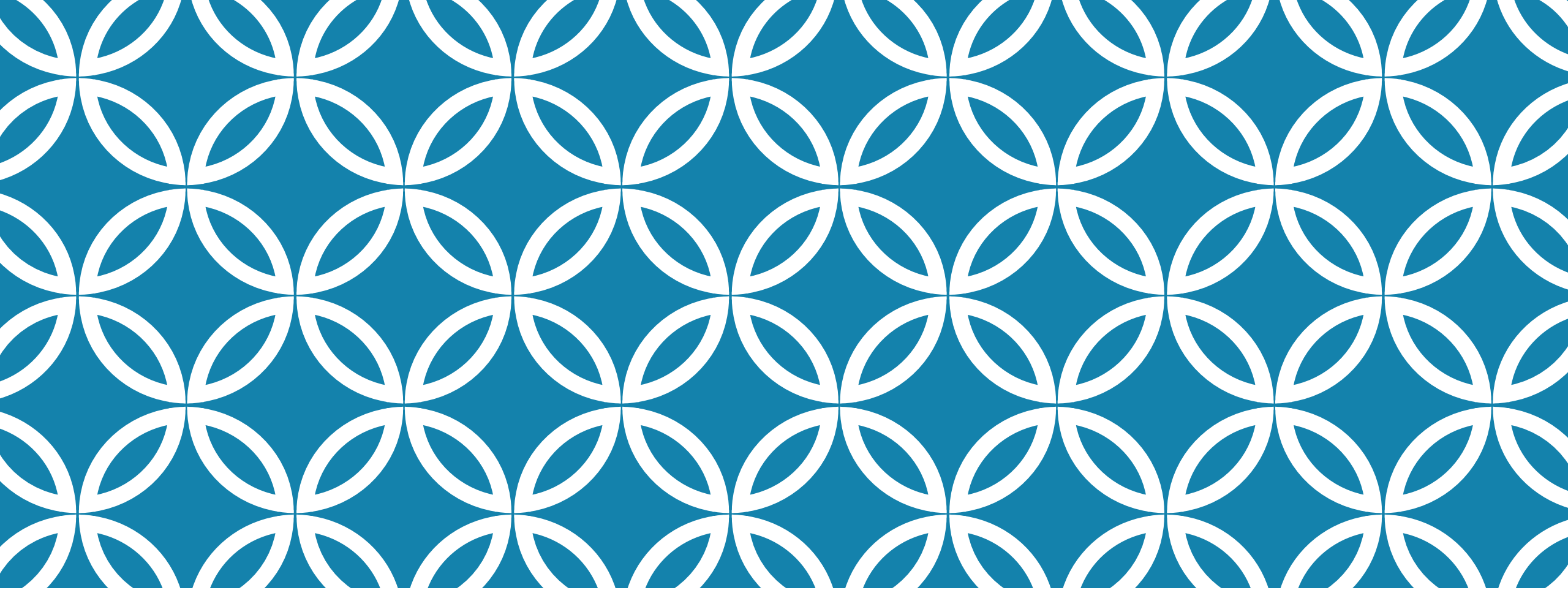


ЛЕКЦИЯ 10 | CI/CD

ЗАЧЕМ НУЖЕН CI/CD?

Классический процесс решения задачи, подходящий для большинства компаний:

1. Берем задачу из списка/получаем от начальства.
2. Создаем новую ветку в git.
3. Пишем программный код.
4. Лично или с помощью коллеги выполняем код-ревью (*code review* — обзор/проверку кода).
5. Запускаем тесты..
6. Сливаем ветку в develop, затем в master, когда будет готов новый релиз.
7. Выполняем сборку проекта.
8. Публикуем новую сборку.

ЗАЧЕМ НУЖЕН CI/CD?

Важно затрачивать как можно меньше времени на развёртывание. Для автоматизации этого процесса и придумана концепция CI/CD.

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 трудоёмкость интеграции и сделать её более предсказуемой за счет раннего обнаружения и устранения ошибок и противоречий. Основным преимуществом является сокращение стоимости исправления дефекта, за счёт раннего его выявления.

Непрерывная интеграция впервые названа и предложена Гради Бучем в 1991 году.

ТРЕБОВАНИЯ К ПРОЕКТУ, В КОТОРЫЙ НЕОБХОДИМО ВСТРОИТЬ CI

1. Исходный код и всё, что необходимо для сборки и тестирования проекта, хранится в репозитории системы управления версиями (Git, Subversion, Mercurial и т.д.)
2. Операции копирования из репозитория, сборки и тестирования всего проекта автоматизированы и легко вызываются из внешней программы

ОРГАНИЗАЦИЯ CI

На выделенном сервере организуется служба, в задачи которой входят:

1. Получение исходного кода из репозитория.
2. Сборка проекта (компиляция).
3. Выполнение тестов.
4. Развёртывание готового проекта (иногда эту часть рассматриваю уже как Continuous Delivery, о нём мы поговорим немного позже).
5. Отправка отчетов (на электронную почту).

Сборка может осуществляться:

1. По внешнему запросу.
2. По расписанию.
3. По факту обновления репозитория и по другим критериям.

СБОРКА ПО РАСПИСАНИЮ

В случае сборки по расписанию (*daily build, ежедневная сборка*), они, как правило, проводятся каждой ночью в автоматическом режиме — *ночные сборки*.

Для различия дополнительно вводится система нумерации сборок — обычно, каждая сборка нумеруется натуральным числом, которое увеличивается с каждой новой сборкой.

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Плюсы	Минусы
1) проблемы интеграции выявляются и исправляются быстро, что оказывается дешевле 2) немедленный прогон модульных тестов для свежих изменений 3) постоянное наличие текущей стабильной версии вместе с продуктами сборок — для тестирования, демонстрации, и т. п. 4) немедленный эффект от неполного или неработающего кода приучает разработчиков к работе в итеративном режиме с более коротким циклом	1) затраты на поддержку работы непрерывной интеграции 2) потенциальная необходимость в выделенном сервере под нужды непрерывной интеграции 3) 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

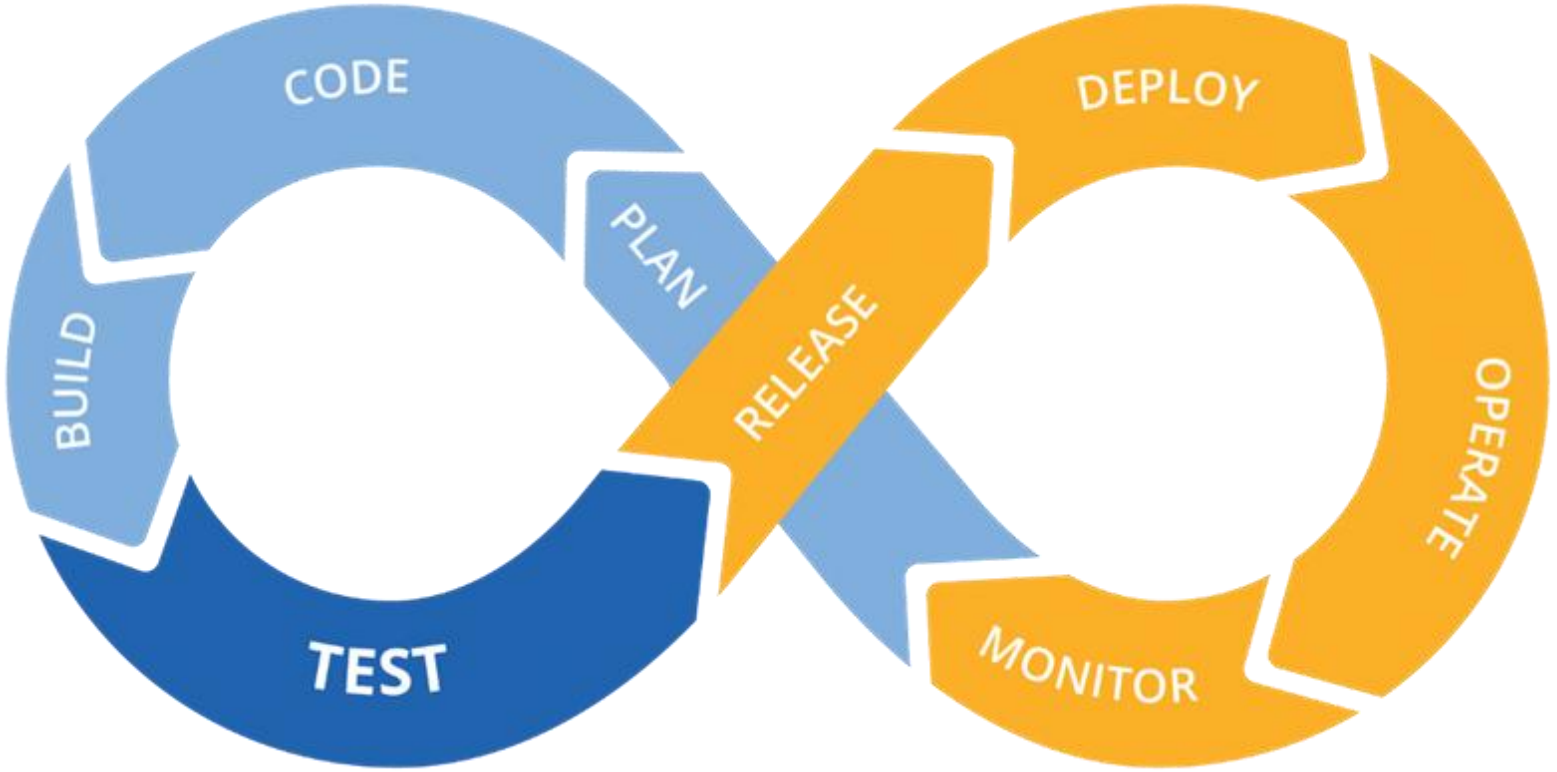
НЕПРЕРЫВНАЯ ДОСТАВКА

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.

Непрерывная доставка располагается «на уровень выше» непрерывного развертывания (CI). В данном случае все изменения, вносимые в исходный код, автоматически развертываются в промышленную среду. Как правило, задача разработчика сводится к проверке запроса на включение (pull request) от коллеги.

ОБЩАЯ СХЕМА CI+CD



CI/CD: ПРИНЦИПЫ, ВНЕДРЕНИЕ, ИНСТРУМЕНТЫ

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

ПРИНЦИПЫ CI+CD

1. Первый принцип CI/CD: разделение ответственности заинтересованных сторон (разработчики, инженеры по качеству, аналитики, Dev-Ops инженеры, владельцы продукта).
2. Второй принцип CI/CD: снижение риска (каждый отвечает за свою роль и пытается снизить риски для бизнеса в своей работы).
3. Третий принцип CI/CD: короткий цикл обратной связи.
4. Реализации среды в CI/CD (чаще всего заводятся отдельные ветви в репозитории).
5. Инструменты для CI/CD (локальные: GitLab CI, TeamCity, Bamboo, GoCD Jenkins, Circle CI; облачные: BitBucket Pipelines, Heroku CI, Travis, Codeship, Buddy CI, AWS CodeBuild).

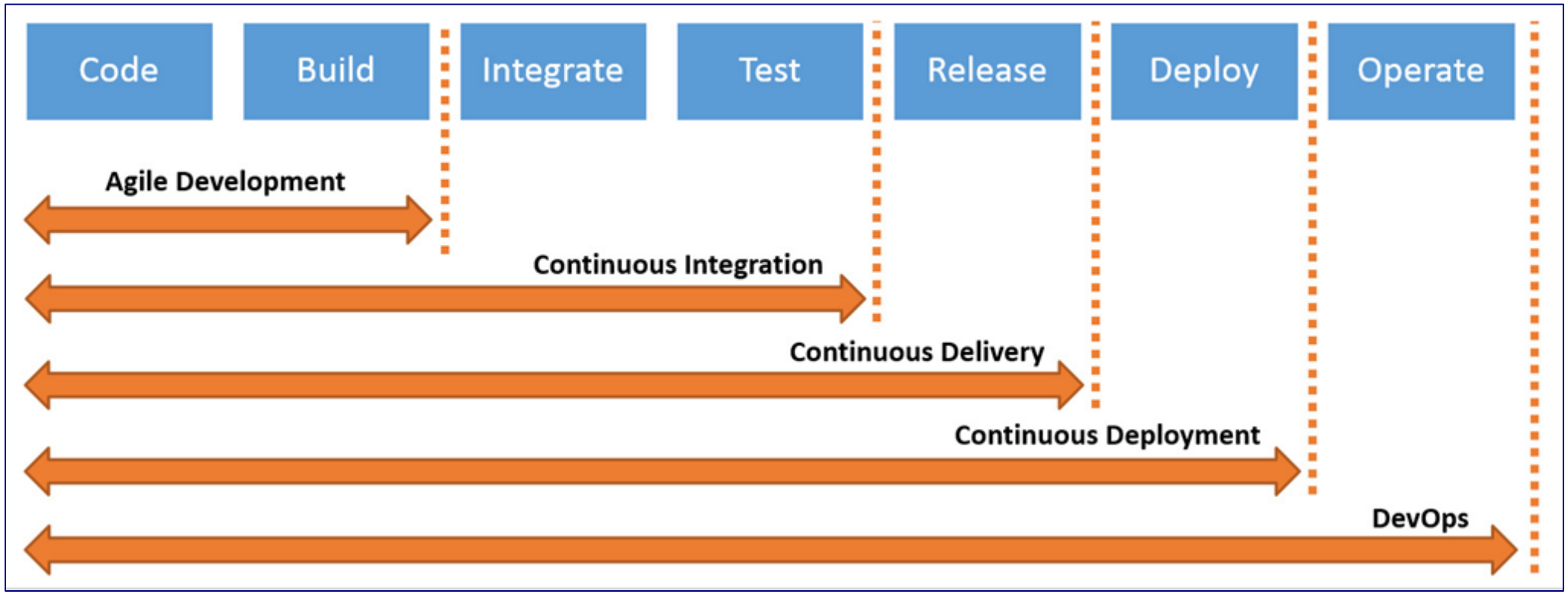
DEVOPS – ЧТО ЖЕ ЭТО?

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*

КАК ВСЁ РАБОТАЕТ ВМЕСТЕ? (ВЗАИМОДЕЙСТВИЕ CI/CD И DEVOPS)



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)

Интеграция DevOps предназначена для доставки продукта, непрерывного тестирования, тестирования качества, разработки функций и обновлений обслуживания для повышения надежности и безопасности и обеспечения более быстрого цикла разработки и развертывания.

ПРЕИМУЩЕСТВА DEVOPS

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

1. Значительном сокращении времени выхода на рынок;
2. Улучшении удовлетворенности клиентов;
3. Улучшении качества продукции;
4. Более надежных выпусках;
5. Повышении производительности и эффективности;
6. 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 развертывания;

- изменяемость;

- тестируемость;

- возможность мониторинга (система постоянного наблюдения за явлениями и процессами, проходящими в приложении).