

**DevOps**

DevOps — это методология разработки, которая помогает наладить эффективное взаимодействие разработчиков с другими IT-специалистами. Это набор процессов и инструментов, которые позволяют компании создавать и улучшать продукты быстрее, чем при использовании традиционных подходов к разработке программного обеспечения.

Термин DevOps — это комбинация слов «разработка» (development) и «эксплуатация» (operations), которая отражает процесс интеграции этих дисциплин в единый непрерывный процесс. Разработчики и тестировщики отвечают за Development, а администраторы — за Operations.

**Семь этапов жизненного цикла DevOps**

Жизненный цикл DevOps можно разделить на семь этапов, которые регулируют процесс непрерывной разработки программного обеспечения от начала до конца. Чтобы понять принципы DevOps, нужно знать каждый этап этого жизненного цикла, а также процессы и требования каждого этапа.

**Непрерывная разработка и доставка**

Разработка ПО начинается с планирования и написания кода. Согласно методологии DevOps, для этого используется процесс регулярной доставки, цель которого — непрерывное улучшение продукта.

Методология DevOps основана на базовых ценностях Agile-разработки и стимулирует регулярные и частые выпуски ПО. Стандартный способ решения этой задачи — автоматизация интеграции и развертывания кода. Этот процесс называет непрерывной интеграцией/непрерывным развертыванием (CI/CD).

На всех этапах разработки команда использует анализирует обратную связь и предлагает возможные решения на этапе планирования

После планирования начинается разработка исходного кода и создание ассетов для дальнейшего развития продукта. Независимо от используемого языка программирования, главная задача — поддержание кодовой базы с использованием инструментов управления исходным кодом.

**Непрерывная интеграция**

Непрерывная интеграция (CI) — это методика разработки, которая требует выгрузки кода в общий репозиторий несколько раз в день. Каждое изменение, или ветка, затем проверяется в ходе автоматической сборки, что позволяет командам быстрее обнаруживать проблемы, обеспечивая работоспособность и готовность основной ветки к дальнейшей разработке.

Методика непрерывной интеграции по своей сути подразумевает множество небольших, итеративных изменений, а не редкие и крупные обновления. Это позволяет разработчикам масштабировать процессы сборки, тестирования, слияния и загрузки в общие репозитории благодаря автоматизации процессов.

Основная цель непрерывной интеграции — быстрая доставка качественного кода. За счет небольших регулярных изменений и автоматизации команды разработчиков могут быстро находить и устранять ошибки, а также сокращать сроки проверки и выпуска обновлений.

**Непрерывное тестирование**

Непрерывное тестирование идет рука об руку с непрерывной интеграцией. Процессы CI/CD больше зависят от [автоматизированных тестов](https://create.unity.com/bugs-and-automation-e-book), а не от проверок кода вручную. Это позволяет обеспечить высокое качество развертываемого кода и исключить появление критических для функциональности игры ошибок до ее выпуска.

DevOps полагается на максимальную автоматизацию процессов. Чем больше ручного труда, тем больше времени тратится впустую и тем выше вероятность появления ошибок. Цель инструментов непрерывного тестирования DevOps — не только найти ошибки, но и сделать это как можно быстрее, чтобы их не пришлось устранять на этапе производства с помощью патчей или исправлений, что сделает процесс еще более сложным и длительным.

Автоматизированные тесты проводятся до начала сборки, а также до производственного этапа. Команды могут провести ручную проверку непосредственно перед производством, а также после завершения автоматизированного тестирования.

**Непрерывный мониторинг**

Непрерывный мониторинг предназначен для эффективной реализации жизненного цикла DevOps. Его конечная цель — удобство для пользователя. Обновления и использование ПО внимательно отслеживаются, а собранные аналитические сведения используются для поддержки работоспособности ПО.

На этапе непрерывного мониторинга команды стремятся обнаруживать и устранять системные ошибки как можно быстрее. Для этого необходима автоматизированная система регистрации ошибок. Автоматизация также помогает получать информацию о других аспектах, включая общую производительность, поведение пользователей, стабильность инфраструктуры разработки, и о многом другом.

Помимо контроля за автоматизацией, специалисты DevOps отвечают за соответствие процессов разработки требованиям стандартов безопасности. На этом этапе также происходит ручная настройка управления выпуском.

**Непрерывная обратная связь**

Непрерывная обратная связь требует создания цикла обратной связи для сбора информации о производительности ПО от сотрудников команды и пользователей. Результаты затем передаются сотрудникам отдела DevOps для дальнейшего улучшения продукта. Источниками данных могут выступать опросы, анкеты, фокус-группы, социальные сети, форумы и многое другое.

Этот процесс направлен не только на обеспечение работоспособности ПО, но и на оценку общей удовлетворенности клиентов для формулирования бизнес-стратегии и достижения наилучших результатов. Непрерывную обратную связь следует использовать для выработки планов развития продукта и удовлетворения запросов аудитории.

**Непрерывное развертывание**

Непрерывное развертывание работает совместно с непрерывной интеграцией, дополняя цикл автоматизации и сводя к минимуму или полностью устраняя необходимость вмешательства человека в процесс развертывания. Автоматизированные инструменты DevOps отслеживают обновления кода и автоматически развертывают их в производственной среде после завершения этапа тестирования. Это экономит время и повышает удовлетворенность пользователей.

Непрерывное развертывание ускоряет циклы обратной связи с пользователями за счет автоматизации. Можно также использовать методы выпуска версий, недоступных всей аудитории или доступных только отдельным пользователям, с целью тестирования новых функций и сбора отзывов (так называемые «выключатели функций»).

Поскольку код обновляется малыми правками, это сокращает риски, сопряженные с масштабными изменениями кода. А автоматизация значительно упрощает весь процесс.

**Непрерывная поддержка**

Непрерывная поддержка направлена на сокращение простоев и предотвращение сбоев, которые расстраивают пользователей. Этот этап жизненного цикла DevOps сосредоточен на оптимизации приложений и сред для обеспечения стабильности и производительности. Он также завершает жизненный цикл DevOps, снабжая участников планирования непрерывной разработки отчетами об ошибках и отзывами пользователей для дальнейшего развития продукта.

Непрерывное взаимодействие разработчиков и пользователей позволяет эффективно передавать и обрабатывать отчеты об ошибках, отзывы о функциях и безопасности для улучшения продукта в рамках цикла DevOps.

Quality Gates переводится как ворота качества.  Это автоматические проверки проекта, кода или производства во время всего процесса работы над ним. Например, код проходит проверку, его пропускают дальше, если нет — отправляют на доработку.

В Quality Gates вы не можете перейти к следующему этапу, пока не завершите предыдущий.

**Как работает Quality Gates**

Чаще всего Quality Gates используют для проверки кода на ранних этапах разработки, пока он еще не объединился с другими кусочками кода. Это помогает увидеть ошибки в самом начале и быстро их исправить.

Quality Gates проходит по заранее определенным этапам. На каждом этапе код проверяют по разным критериям. Их заранее формулирует команда. Если соответствует —  то разработчик переходит к следующей фазе разработки.

Quality Gates нужен, чтобы снизить риски  в разработке и уменьшить количество багов. Еще ворота качества помогают сделать качественный проект, который можно будет без проблем поддерживать после его развертывания.

**Как работает Quality Gates в проекте**

Quality Gates для проектов основаны на чек–листах. Проект делится на этапы, на каждом этапе проходят контрольные проверки.  Эти проверки включают в себя вопросы по объему проекта, бюджету, заинтересованным сторонам, рискам и соответствию требованиям.

Представьте, что вы руководите CRM–проектом на этапе разработки. Ваша команда продумала архитектуру и  отправляет ее по электронной почте разработчикам. Разработчики приступают к конфигурированию и настройке CRM–приложения в соответствии со спецификацией. Но вдруг вы узнаете от другой проектной команды, что среда разработки и инфраструктура еще не созданы: нет поддержки базы данных. Это напрямую влияет на сроки проекта. Но уже ничего нельзя сделать. Если бы вы использовали Quality Gate, то на этапе проектирования обнаружили бы, что план проекта не соответствует требованиям.