

Непрерывное тестирование

При регулярных аудитах кода наличие оптимизированной стратегии тестирования, встроенной в поток CI/CD, обеспечивает качественное замыкание цикла обратной связи. Непрерывное тестирование занимает центральное место в оптимизации потока CI/CD. Быстрое внедрение программных функций является приоритетом, но сохранение высокого уровня качества этих функций обеспечивается через постоянное тестирование.



Вот как оно может быть настроено:

- Модульные тесты являются фундаментом стратегии тестирования. Обычно они проводятся на компьютере разработчика и являются самыми экономичными и быстрыми. В общем случае, 70% усилий тестирования должны быть направлены на юнит-тестирование. Ошибки, выявленные на данном этапе, можно быстро и просто устранить. Также рекомендуется проводить статический анализ кода для модификаций и проверять покрытие кода, соблюдение стандартов кодирования и прочее. Компактные модульные тесты без зависимостей работают быстрее, поэтому программист может быстро выявить неудачные тесты.
- Этап сборки(build) – это начальная стадия тестирования взаимодействия между разными компонентами и индивидуальными компонентами. Также на этом этапе возможно выявление, не вызывает ли добавленный разработчиком код конфликтов с текущей функциональностью, а регрессионное тестирование помогает в этом.

- Этап тестирования на пре-продакшн stage. Стейджинговая среда представляет собой точное отображение продакшн среды. Здесь осуществляется комплексное тестирование системы, акцентируя внимание на пользовательском интерфейсе, бэкенд-логике и API. Также тут может проходить тестирование производительности, которое анализирует эффективность приложения при специфичной нагрузке.
И этот период также проводится UAT (User Acceptance Testing – приемочное тестирование) как часть подготовки к внедрению в производственную среду. Проверка на соответствие требованиям проводится для удостоверения, что продукт соответствует стандартам.
- После релизе, на этапе производства, для оценки новой версии приложения применяются методы, такие как A/B-тестирование с помощью FeatureFlags(обсуждали ранее) или канареечного анализа.

A/B тестирование с использованием Feature Flags или канареечного релиза

1. Feature Flags (Toggle):

Feature Flags позволяют вам динамически включать или отключать определённые функции вашего приложения без необходимости деплоя. Это может быть использовано для A/B тестирования путем включения новой функции для определённого процента пользователей.

Пример:

Допустим, у вас есть веб-сайт, и вы хотите тестировать новый дизайн главной страницы.

- Вы можете использовать Feature Flag, чтобы показать новый дизайн 50% посетителей и оставить старый дизайн для остальных 50%.
- По результатам пользовательской активности и обратной связи вы можете решить, стоит ли полностью внедрять новый дизайн или внести изменения.

2. Канареечный релиз:

В этом методе, новая версия продукта (или новая функция) изначально выпускается для малого процента пользователей. По мере того как она проходит испытание и демонстрирует стабильность, доля пользователей, видящих новую версию, увеличивается.

Пример:

Вы разрабатываете мобильное приложение и хотите добавить новую функцию, например, новую систему рекомендаций.

- Вы можете выбрать 10% вашей аудитории для тестирования новой функции, пока остальные 90% пользователей продолжают использовать старую версию.
- Если новая система рекомендаций получает положительные отзывы и не вызывает технических проблем, вы можете увеличить процент пользователей, которым доступна новая функция, до тех пор, пока все не получат обновление.

Оба метода позволяют избегать рисков, связанных с внедрением новых функций, и позволяют получить ценную обратную связь от пользователей на ранних стадиях развертывания.