CI/CD concept

Continuous Integration – это непрерывное автоматическое внесение изменений (даже самых не значительных) при которых разработчики регулярно объединяют изменения программного кода в центральном репозитории, после чего выполняется сборка, тестирование и запуск.

Цели CI:

- Выявление изменений которые ломают работу приложения КАК МОЖНО РАНЬШЕ (дешевле фиксить).
- Создание готового файла/приложения который готов к работе и на 100% будет коректно выполнять свои функции.
- Автоматическое развёртывание приложений.
- Мониторинг здоровья проекта
- Визуализация цепочек сборки
- Розпаралеливание задач (ускоряет совместную работу)

Этапы Continuous Integration:

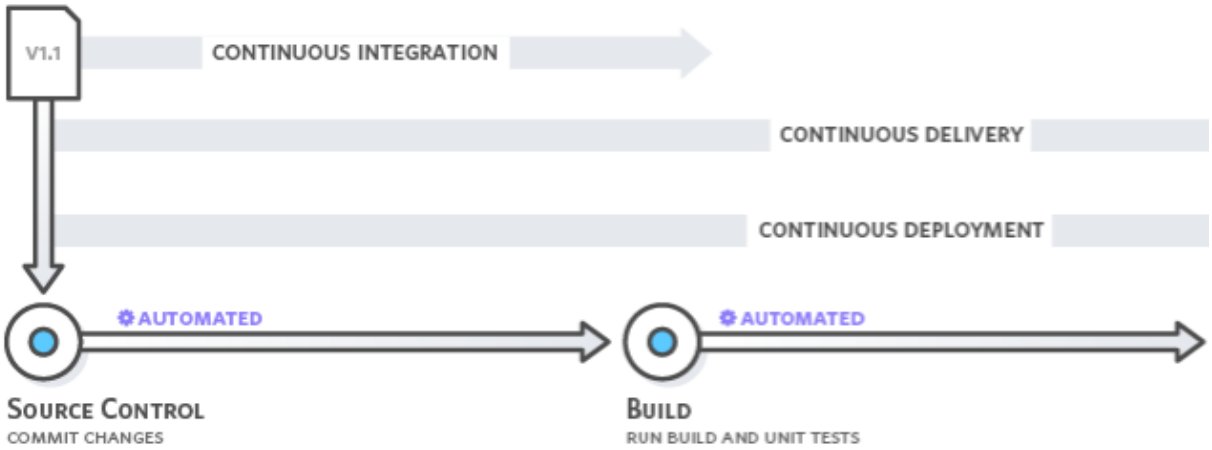
Code (Написание кода). Каждый из разработчиков пишет код своего модуля, проводит ручное тестирование, а затем соединяет результат работы с текущей версией проекта в основной ветке. Для контроля версий используется система Git, либо аналогичные решения. Когда участники команды опубликуют код своих модулей в основной ветке, начнется следующий этап.

Build (Сборка). Система контроля версий запускает автоматическую сборку и тестирование проекта. Триггеры для начала сборки настраиваются командой индивидуально — фиксация изменений в основной ветке проекта, сборка по расписанию, по запросу и т.д. Для автоматизации сборки используется Jenkins, либо аналогичный продукт. (build — процесс получения информационного продукта из исходного кода. Чаще всего включает компиляцию и компоновку, выполняется инструментами автоматизации)

Test (Ручное тестирование). Когда CI система успешно проверила работоспособность тестовой версии, то код отправляется тестировщикам для ручного обследования. При этом тестовая сборка получает номер кандидата для дальнейшего релиза продукта (например, v.1.0.0-1).

Release (Релиз). По итогам ручного тестирования сборка получает исправления, а итоговый номер версии кандидата повышается (например, после первого исправления версия v.1.0.0-1 становится v.1.0.0-2). После этого выпускается версия кода для клиента (например, v.1.0.0) и начинается следующий этап цикла

Continuous Integration относиться к стадии сборки и модульного тестирования процесса выпуска ПО. Каждое подтверждённое изменение кода запускает автоматический процесс сборки и тестирования.



С помощью **CI** изменения программного кода автоматически проходят сборку, тестируются и подготавливаются к запуску в рабочей среде. Непрерывная доставка расширяет практику непрерывной интеграции за счет того, что все изменения кода после стадии сборки развертываются в тестовой или в рабочей среде.

Continuous Delivery – это практика разработки ПО при которой после любых изменений в программном коде выполняется автоматическая сборка, тестирование и подготовка к окончательному релизу. (отличается от Continuous Integration тем что все изменения после сборки разворачиваются в тестовой или рабочей среде)

Этапы Continuous Delivery:

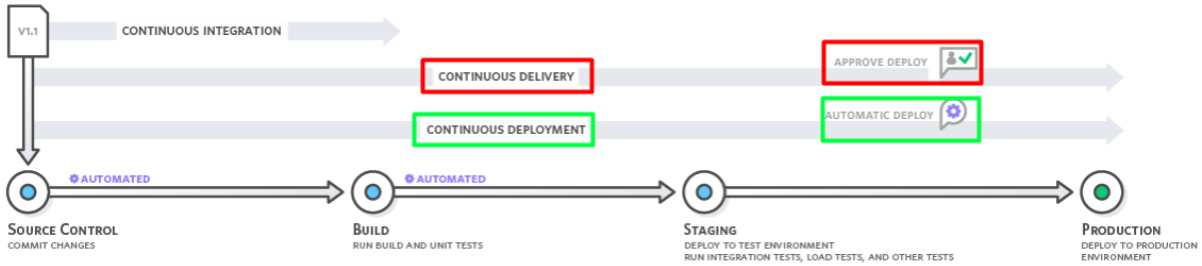
Deploy (Развертывание). На этом этапе рабочая версия продукта для клиентов автоматически публикуется на production серверах разработчика. После этого клиент может взаимодействовать с программой и ознакомиться с ее функционалом как непосредственно через готовый интерфейс, так и через облачные сервисы.

Operate and Monitoring (Поддержка и мониторинг). Конечные пользователи начинают работать с продуктом. Команда разработки поддерживает его и анализирует пользовательский опыт.

Plan (Планирование). На основе пользовательского опыта формируются запросы на новый функционал для продукта, готовится план доработок. После этого цикл замыкается и переходит в начальную стадию — написание кода. Далее начинается новая итерация CI/CD разработки.

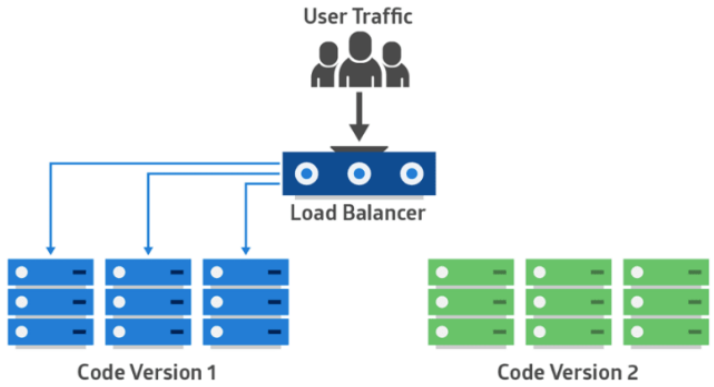
Continuous Delivery автоматизирует весь процесс выпуска ПО. Каждое подтверждение записи версии запускает автоматический процесс сборки, тестирования и размещения обновления. Окончательное решение о развертывании в реальной рабочей среде инициируется разработчиком.

Continuous Deployment – это тоже самое что Continuous Delivery но без вмешательства человека. А именно при Continuous Delivery для деплоя приложения в production stage необходимо подтверждение вручную. При Continuous Deployment это происходит автоматически без специального вмешательства.

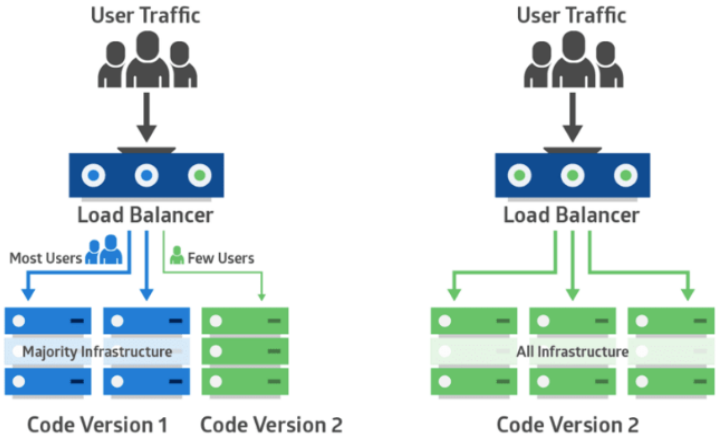


Deployment strategies

Blue-Green (Red-black, A/B) Deployment – для этого типа развёртывания необходимо иметь 2 полноценных окружения. Сначала обновление выкатывается на окружение без трафика (green) тестируется его функциональность и работоспособность, а уже после этого переключается поток трафика (при помощи LB). Этот тип deployment лучше всего подходит для cloud решений.



Canary Deployment – принцип похож на BLUE-GREEN но этот метод еще менее рисковый, так как обновление будет выкочено только на не значительное количество серверов, на которые будет направлена часть трафика для тестирования среды и только после удачной проверки обновление будет применено на остальные сервера.



Rolling Deployment – этот подход называется «постепенным выкатыванием» - так как обновления сначала применяется на минимальном количестве серверов/под, в случае позитивных результатов деплоя, начинается постепенное выкатывание обновления на всё новые и новые сервера, поды.