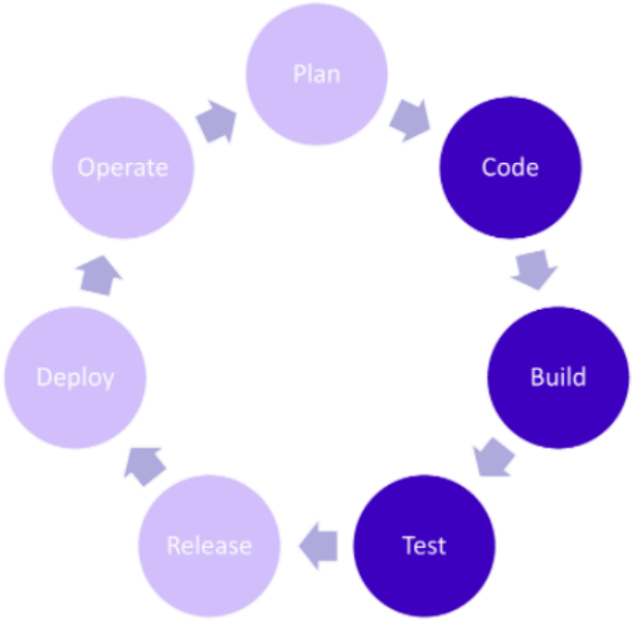
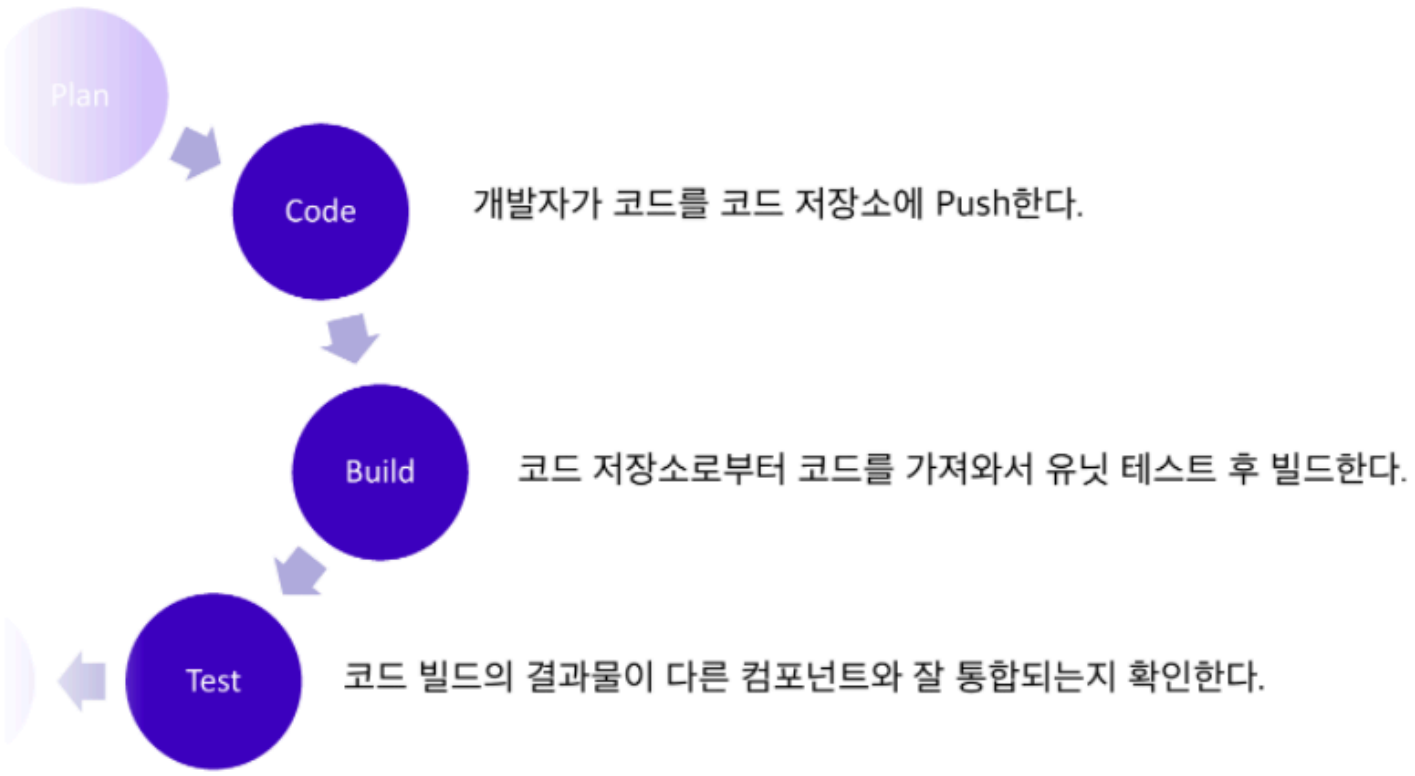
지속적 통합 (Continuous Integration)




CI/CD에 대하여 - Part 1. CI 지속적 통합

# Continuous Intergration

지속적 통합은 기존에 있던 기능과 새로운 기능의 통합을 의미한다.
(CI tool: CI를 자동화 해주는 소프트웨어)




## 도입함으로서 얻을 수 있는 장점

- 버그를 일찍 발견할 수 있다.
- 빌드 및 테스트와 같이 사람이 해야 할 일들을 자동화할 수 있다.
- 테스트가 완료된 코드에 대해 빠른 전달이 가능하다.

- 지속적인 배포가 가능해진다.
- 개발자의 생산성을 향상시킬 수 있다.

# 테스트가 중요한 이유

- 테스트를 통해 결함과 버그를 조기에 발견 할 수 있으며, 이는 개발자의 생산성을 향상할 수 있습니다.
- 제품의 결함과 버그를 발견하고 수정하는 것은 소프트웨어의 품질을 보증하고, 더 안정적이고 사용하기 쉽게 만듭니다.

소프트웨어 개발 분야의 설계 및 개발 방법론에 있어서 저명한 마틴 파울러는 지속적 통합에 대해 다음과 같이 정의합니다.

> 팀 구성원이 각자의 작업을 자주 통합하는 소프트웨어 개발 방식
> *- Martin Fowler*

# 지속적 통합의 원칙

## 1. 레파지토리 단일화

- 프로젝트에서 제품을 빌드하기 위해 함께 조정해야 하는 수많은 파일이 포함되어 있기 때문입니다.
- 이 모든 파일이 단일 소스 레파지토리가 아닌 곳에 뿔뿔이 흩어져 있다면 추적하는 것이 굉장히 힘들어질 것입니다.

## 2. 빌드 자동화

- 사람들에게 이상한 명령을 입력하게 하거나 대화 상자를 클릭하게 하는 것은 시간 낭비입니다.
- 빌드가 수동으로 진행된다면, 수 많은 실수를 낳을 수 있습니다.

## 3. 셀프 테스팅 빌드

- 빌드 프로세스에 자동화된 테스트를 포함 함으로써 버그를 더 빠르고 효율적으로 파악할 수 있습니다.
- 지속적 통합의 일부인 테스트 주도 개발(TDD)을 통해 손상된 빌드를 즉시 확인, 수정할 수 있습니다.

## 4. 매일 메인라인 커밋

메인라인: 시스템의 현재 상태를 의미하며 회사마다 어떤 브랜치를 메인라인으로 두는지는 조금씩 다르지만, 여기서는 master 브랜치를 메인라인으로 취급하도록 하겠습니다.

- 각자의 진행 상황을 추적하는 데 도움이 됩니다.
- 짧은 주기로 커밋을 하므로, 충돌이 발생한 후 빠르게 충돌 상황을 감지할 수 있습니다.
    - 해당 시점에는 충돌이 많이 발생하지 않아 문제를 쉽게 해결할 수 있습니다.
    - 만약 통합의 빈도가 길어 충돌을 늦게 발견하게 된다면 문제를 해결하기 매우 어려울 수 있습니다.

## 5. 모든 팀원이 무슨 일이 일어나고 있는지 알아야 합니다.

- 지속적 통합은 커뮤니케이션에 관한 것이므로 모든 사람이 시스템 상태와 시스템에 적용된 변경사항을 쉽게 확인할 수 있어야 합니다.

위에서 언급한 원칙 외에도, 그 밖에 Martin Fowler가 제시한 나머지 원칙도 함께 소개합니다.

## 6. 모든 커밋은 통합 서버의 메인라인에서 빌드 돼야 합니다.

- 개발자간 개발 환경에 차이가 있기 때문입니다.

## 7. 빌드의 오류를 즉시 수정 할 수 있어야 합니다.

- 빌드가 중단되는 것이 나쁜 것은 아니지만, 이러한 상황이 매번 항상 발생하는 경우 사람들이 커밋 전에 로컬에서 업데이트 및 빌드에 대해 충분히 주의하지 않고 있음을 시사합니다.
- 지속적 통합 도구를 사용하면 빌드의 오류를 즉시 확인 할 수 있습니다.

## 8. 빌드가 빨리 되도록 유지해야 합니다.

- 지속적 통합의 요점은 신속한 피드백을 제공하는 것입니다.
- 오랜 시간이 걸리는 빌드는 지속적 통합의 가장 큰 장애물입니다.

## 9. 운영 환경과 동일한 환경에서 테스트가 진행돼야 합니다.

- 다른 환경에서 테스트하는 경우 환경의차이로 인해 테스트에서 발생하는 일이 프로덕션에서 발생하지 않을 위험이 있습니다.
- 누구나 최신 실행 파일을 쉽게 얻을 수 있어야 합니다.

## 10. 배포 자동화가 이루어져야 합니다.

- 지속적 통합을 수행하려면 커밋 테스트를 실행하기 위한 환경과 2차 테스트를 실행하기 위한 하나 이상의 환경이 필요합니다. 이러한 환경 간에 실행 파일을 하루에 여러 번 이동하기 때문에 이 작업을 자동으로 수행하고 싶을 것입니다. 따라서 응용 프로그램을 모든 환경에 쉽게 배포할 수 있는 스크립트를 갖는 것이 중요합니다.
- 매일 프로덕션에 배포하지 않을 수도 있지만, 자동 배포는 프로세스 속도를 높이고 오류를 줄이는 데 도움이 됩니다. 또한 테스트 환경에 배포하는 데 사용하는 것과 동일한 기능을 사용하기 때문에 저렴한 옵션이 될 수 있습니다.

# Action Items

- 지속적 통합이 있기 전에는 어떻게 릴리즈를 만들었을까요?
    - 출시 기한을 정해놓고 소프트웨어를 완성하는 폭포수 모델
    - 모든 수정사항을 한꺼번에 메인라인에 합쳐주는 커밋을 하고, 테스트한 다음 릴리즈를 한다.
    - 따라서, 오류가 발생하는 것을 뒤늦게 알게되고 다시 오류를 수정하고 테스트를 하면서 릴리즈 간격이 갈수록 벌어지게 된다.
    - 사용자가 항상 최신 버전을 다운받아 업데이트 해야함
        - 여전히 모바일 애플리케이션이 사용하는 전달 방식

- 지속적 통합을 통해 어떻게 버그를 일찍 발견할 수 있는걸까요?
    - 변경점이 많이 수정되고 나서야 커밋을 하지 않고, 매일 메인라인에 커밋해주면서 지속적인 통합을 한다.
    - 자동화된 통합과정으로 잦은 커밋을 할 수 있으며, 모든 진행상황을 쉽게 추적할 수 있게 된다.
    - 브라우저에 접속하기만 해도, 새 버전을 즉시 사용할 수 있다
        - 사용자가 최신 버전 업데이트를 신경쓰지 않음
        - 원하는 만큼 잦은 릴리즈를 할 수 있음(빠른 문제 해결)
    - 다양한 배포 방식을 적용하거나 A/B 테스트 가능

- 지속적 통합 과정에서 반드시 자동화가 이뤄져야 하는 부분은 어떤 부분인가요?
    - CI/CD 과정( P > C > [B > T] > D > R > O ) 가운데, **Build > Test**
    - 빌드와 테스트를 자동화하여 지속적으로 품질 관리(Quality Control)를 적용하는 프로세스를 실행하는 것이다.
    - 작은 단위의 작업, 빈번한 적용. 지속적인 통합은 모든 개발을 완료한 뒤에 품질 관리를 적용하는 고전적인 방법을 대체하는 방법으로서 소프트웨어의 질적 향상과 소프트웨어를 배포하는데 걸리는 시간을 줄이는데 초점이 맞추어져 있다.

- 지속적 통합을 도입함으로써 기존 개발 방식의 어떠한 문제를 해결해주었을까요?
    - 언젠가는 저장소가 개발자들의 베이스라인과는 너무 많이 달라지게 되는 "통합의 지옥"이라 불리는 상황에 빠지게 된다. 이 경우, 작업하는 시간보다 작업 내용을

통합하는데 소요되는 시간이 더 증가하게 되어, 최악의 경우 개발자들이 자신들의 변경 내용들을 취소하고 작업들을 완전히 처음부터 다시하는 것이 나을 수도 있다.

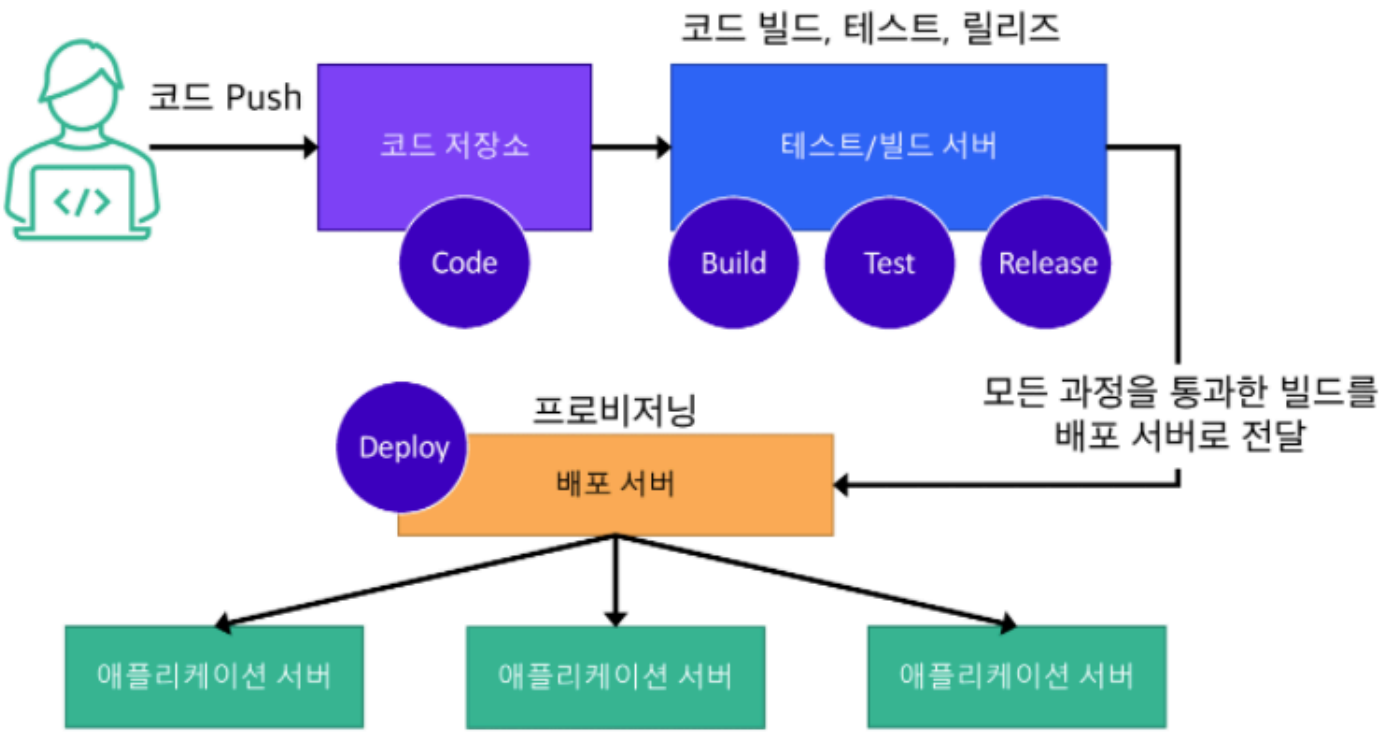
- 지속적인 통합은 "통합의 지옥"의 함정을 피하는 것을 내포하며 재작업에 들어가는 비용과 시간을 줄이는데 초점이 맞추어져 있다.

# 짧게 요약

- 프로젝트는 하나의 Repository에서, 빌드와 테스트를 자동화 하고, 매일 메인라인에 커밋 해주면, 오류를 즉시 수정할 수 있으며, 빠른 빌드와 운영 환경에서 테스트를 자동으로 해줌으로써 지속적 통합을 할 수 있다.
  따라서, 모든 팀원의 진행 상황을 볼 수 있으며 오류에 대한 빠른 대처가 가능하므로 소프트웨어의 질적 향상과 배포시간 단축을 할 수 있다.

## 클라우드 서비스의 전달 방식

고객의 요구에 민첩하게 대응하여 지속적 전달: 애자일(Agile) 모델




이러한 **전달/배포 파이프라인**을 구성하는 것 역시 DevOps의 영역!