CI = 지속적 통합 (Continuous Intergration)
CD = 지속적 전달 (Continuous Delivery)
CI/CD = 지속적 배포 (Continuous Deployment)
Plan > Code > Build > Test > Release > Deploy > Operate

지속적 통합

(Continuous Intergration)

Code = 개발자가 코드를 코드 저장소에 Push한다.
Build = 코드 저장소로부터 코드를 가져와서 유닛 테스트 후 빌드한다.
Test = 코드 빌드의 결과물이 다른 컴포넌트와 잘 통합되는지 확인한다.
결과 확인 후 개선하여 다시 코드 Push
즉, 기존에 있던 기능과 새로운 기능의 통합을 의미한다.

- **필요성**
  버그를 일찍 발견할 수 있다.
  테스트가 완료된 코드에 대해 빠른 전달이 가능하다.
  지속적인 배포가 가능해진다.

지속적 전달

(Continuous Delivery)

Release = 배포 가능한 소프트웨어 패키지를 작성한다.
Deploy = 프로비저닝을 진행하고, 서비스를 사용자에게 노출한다.
Operate = 서비스에 생길 수 있는 현황을 파악하고 문제를 감지한다.

지속적 배포

(Continuous Deployment)

지속적 배포란, 지속적 통합과 지속적 전달의 합집합으로 모든 코드의 변경이 배포로 이어진다는 것이 핵심이다.
전달(Delivery)의 목적은 배포(Deployment)이므로 두 용어를 혼용해서 사용하기도 한다.

릴리즈와 출시

릴리즈와 출시는 다른 의미로 사용할 것이다.
출시 = 상품이 시장에 나오거나, 내보냄 (비즈니스적 관점에서의 용어)
릴리즈 = 배포 가능한 소프트웨어 패키지 (애자일 소프트웨어 개발에서의 의미)

- **a. Plan**

  소프트웨어의 목적과 방향성을 확립하고 계획을 수립한다.

- **b. Code**

  개발자가 코드를 코드 저장소에 Push한다.

- **c. Build**

  코드 저장소로부터 코드를 가져와서 유닛 테스트 후 빌드한다.

- **d. Test**

  코드 빌드의 결과물이 다른 컴포넌트와 잘 통합되는지 확인한다.

- **e. Release**

  배포 가능한 소프트웨어 패키지를 작성한다.

- **f. Deploy**

  프로비저닝을 진행하고, 서비스를 사용자에게 노출한다.

- **g. Operate**

  서비스에 생길 수 있는 현황을 파악하고 문제를 감지한다.

CI/CD 파이프라인은 "개발자가 새로운 소프트웨어 Plan을 계획 후 배포하기 위해 Code를 개발 및 저장소에 Push하고, 코드 저장소로부터 코드를 가져와서 코드 Build와 유닛 테스트 후 결과물이 다른 컴포넌트와 잘 통합되는지 Test하며, 배포 가능한 소프트웨어 패키지를 Release하고 프로비저닝을 거쳐 사용자에게 노출하여 Deploy하고, 이 후에 생기는 현황을 파악하면서 문제가 없는지 지속적으로 Operate 한다."