Cloud Native

Cloud Native

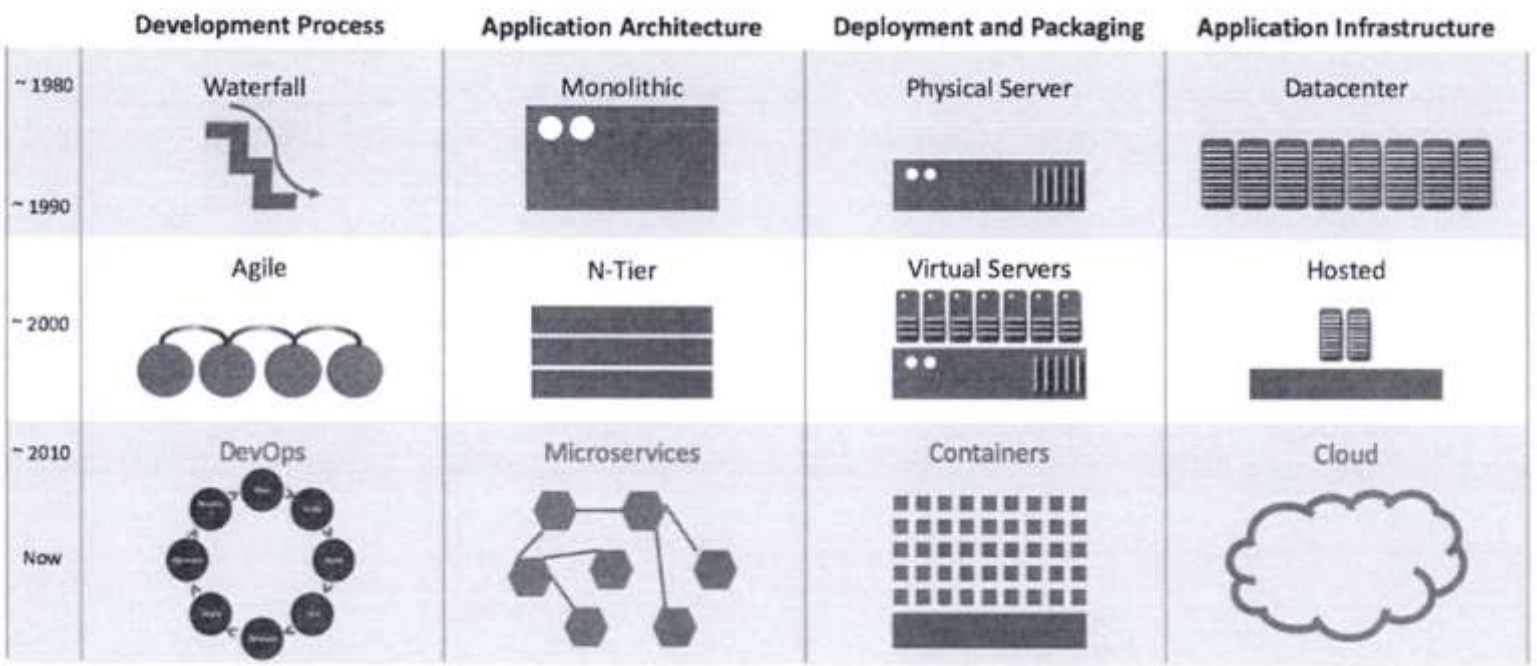
- Cloud Native
  - ✓ Cloud Native 기술은 조직이 Public, Private 그리고 Hybrid Cloud와 같은 현대적이고 동적인 환경에서 확장 가능한 애플리케이션을 개발하고 실행할 수 있게 해주는 기술로 Container, Service Mesh, Micro Service, Immutable Infra, Declarative API가 이러한 접근 방식의 예시이며 이러한 기술은 회복성, 관리 편의성, 가시성을 갖춘 느슨하게 결합된 시스템을 가능하게 해주ㅁ 견고한 자동화 기능을 함께 사용하면 엔지니어는 영향이 큰 변경을 최소한의 노력으로 자주 예측 가능하게 수행할 수 있음
  - ✓ Cloud Native Computing Foundation은 벤더 중립적인 오픈 소스 프로젝트 생태계를 육성하고 유지함으로써 위와 같은 패러다임 채택을 촉진해서 최신 기술 수준의 패턴을 대중화하여 이런 혁신을 누구나 접근 가능하도록 함

Cloud Native

- IT서비스 개발 및 구현 방식의 변화
  - ✓ 개발 방식
    - 35.9% DevOps/DevSecOps
    - 31.78% Agiie/Scrum
    - 13.02% Kanban
    - 10.02% Waterfall
    - 5.01% Water/Scrum/Fall
    - 4.20% Lean

Cloud Native

❖ IT서비스 개발 및 구현 방식의 변화



Cloud Native

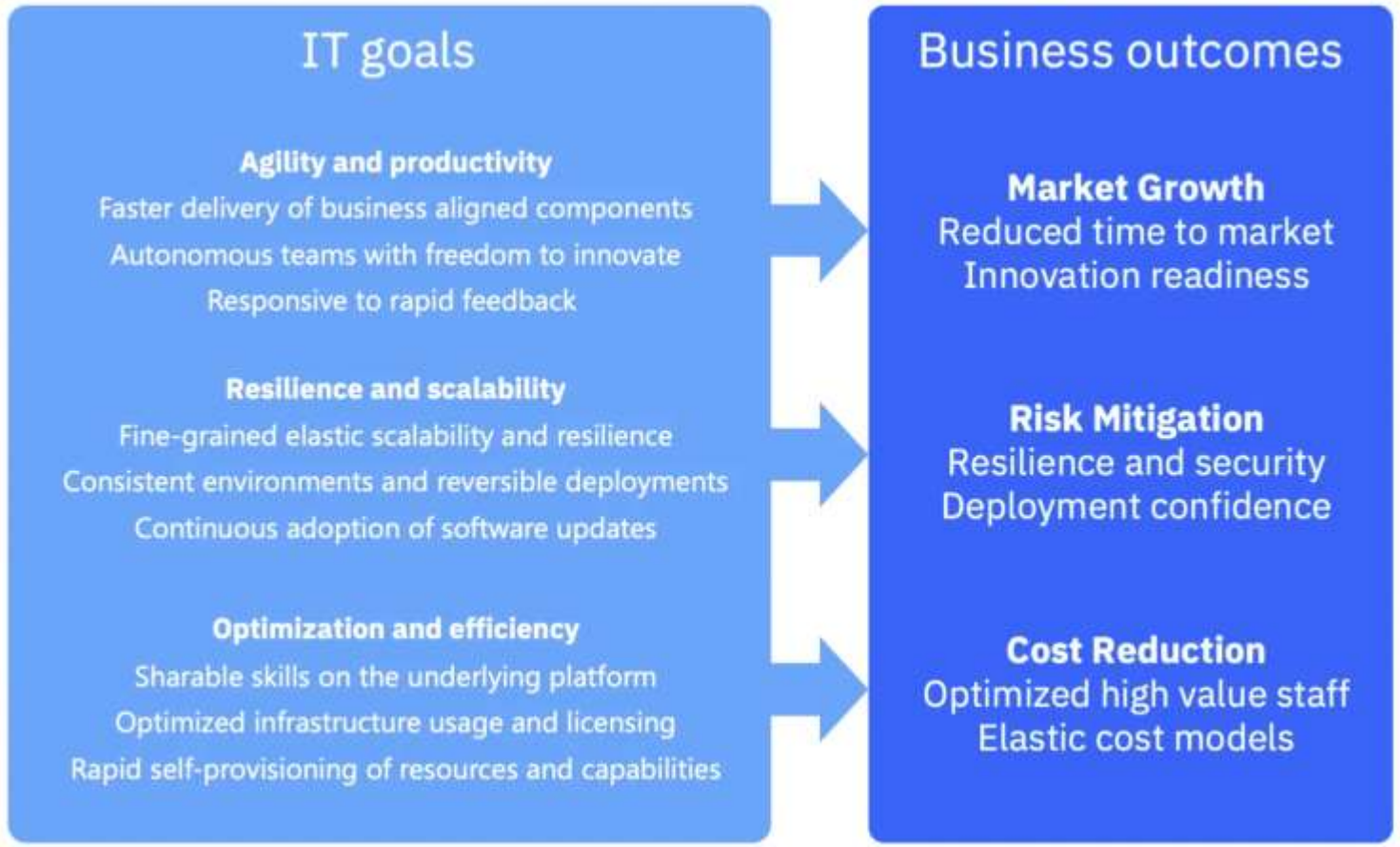
- Cloud Native 적용이 필요한 이유
  - ✓ 서비스 배포 시간 단축: Containers + Micro Service를 통해 개발 팀과 운영 팀 사이의 의사 소통 향상
    - 상호 이해폭 확대
    - DevOps 문화의 내재화 촉진
    - 조직 내 다양한 팀 간의 마찰 감소
    - 지속적인 적용(Continues Delivery)을 통한 빠른 배포 가능
    - 변경 프로세스의 복잡성 감소
    - 변경에 따른 인지된 위험 감소
  - ✓ 애플리케이션 및 서비스 현대화
    - 컨테이너로 애플리케이션을 배포하여 운영 인프라에 대한 종속성 감소
    - 결과적으로 이전의 On-Premise 애플리케이션을 Cloud로 이관 가능
    - Kubernetes는 모든 인프라에 컨테이너를 배포할 수 있는 단일 통합 플랫폼 제공

Cloud Native

- Cloud Native 적용이 필요한 이유
  - ✓ 신속한 신규 서비스 개발 사이클
    - 풍부한 기술 생태계
      - 대규모 커뮤니티
      - Kubernetes 및 CNCF slack에 약 35,000명이 참여
      - 충성도 높은 커뮤니티 활동가들
    - 오픈 소스 기반
      - 개발자에게 친숙한 개발 환경 및 운영 협업 체계
      - 풍부한 개발 인력 Pool
  - ✓ 사업 성장을 위한 조직 문화 혁신 촉진
    - Cloud Native는 조직의 혁신을 가속화하기 위해 새로운 문화, 기술 및 프로세스를 제공
    - DevOps, CI/CD, Containerization은 서비스 개발 조직의 현대화 촉진
    - 이전보다 훨씬 빠르게 조직 문화 및 서비스 문화 변화 촉진

Cloud Native

❖ Cloud Native 적용이 필요한 이유

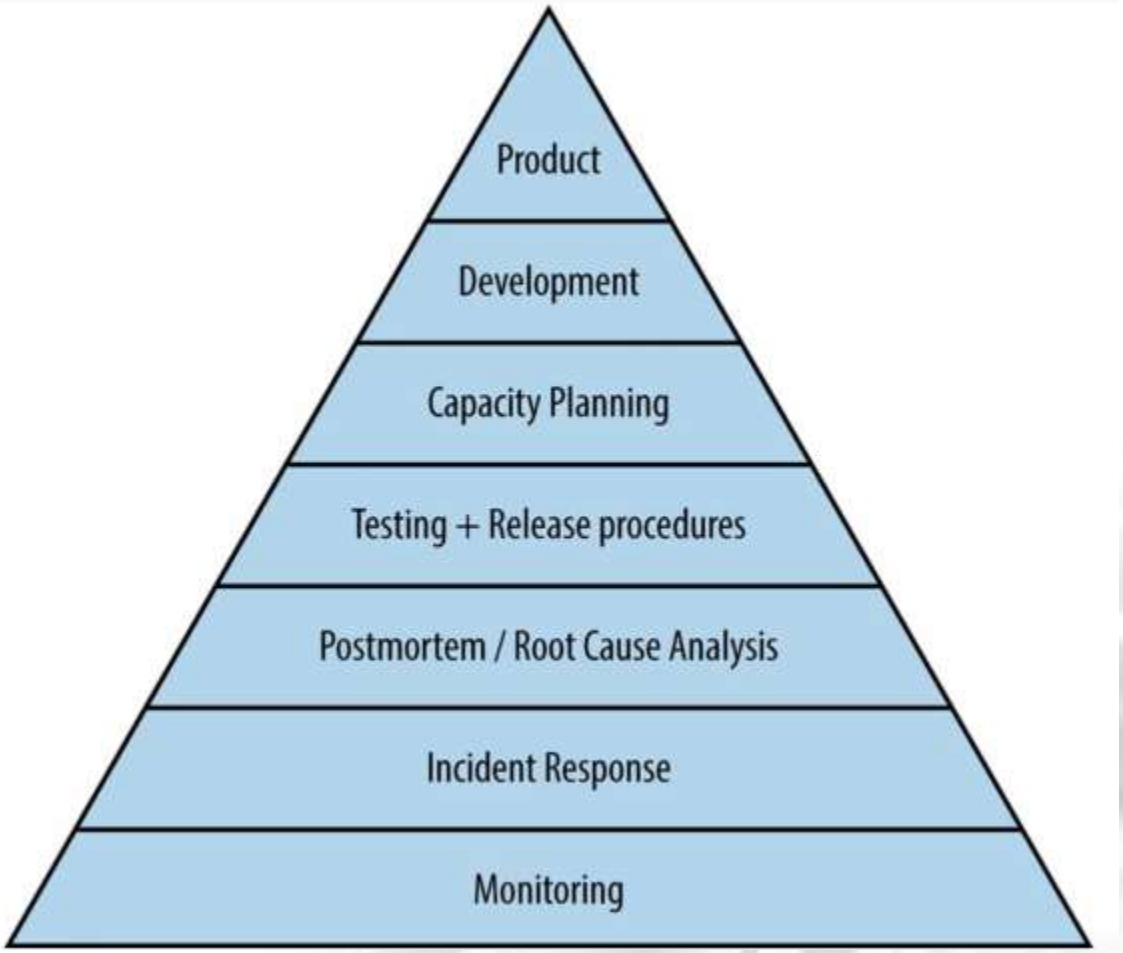


Cloud Native

- Cloud Native 적용이 필요한 이유
  - ✓ IT 목표
    - 민첩성 과 생산성
    - 복원력과 확장성
    - 최적화 와 효율성
  - ✓ 사업 성과
    - 시장 성장
    - 위험 완화
    - 비용 절감

Cloud Native

- CNCF Trail Map

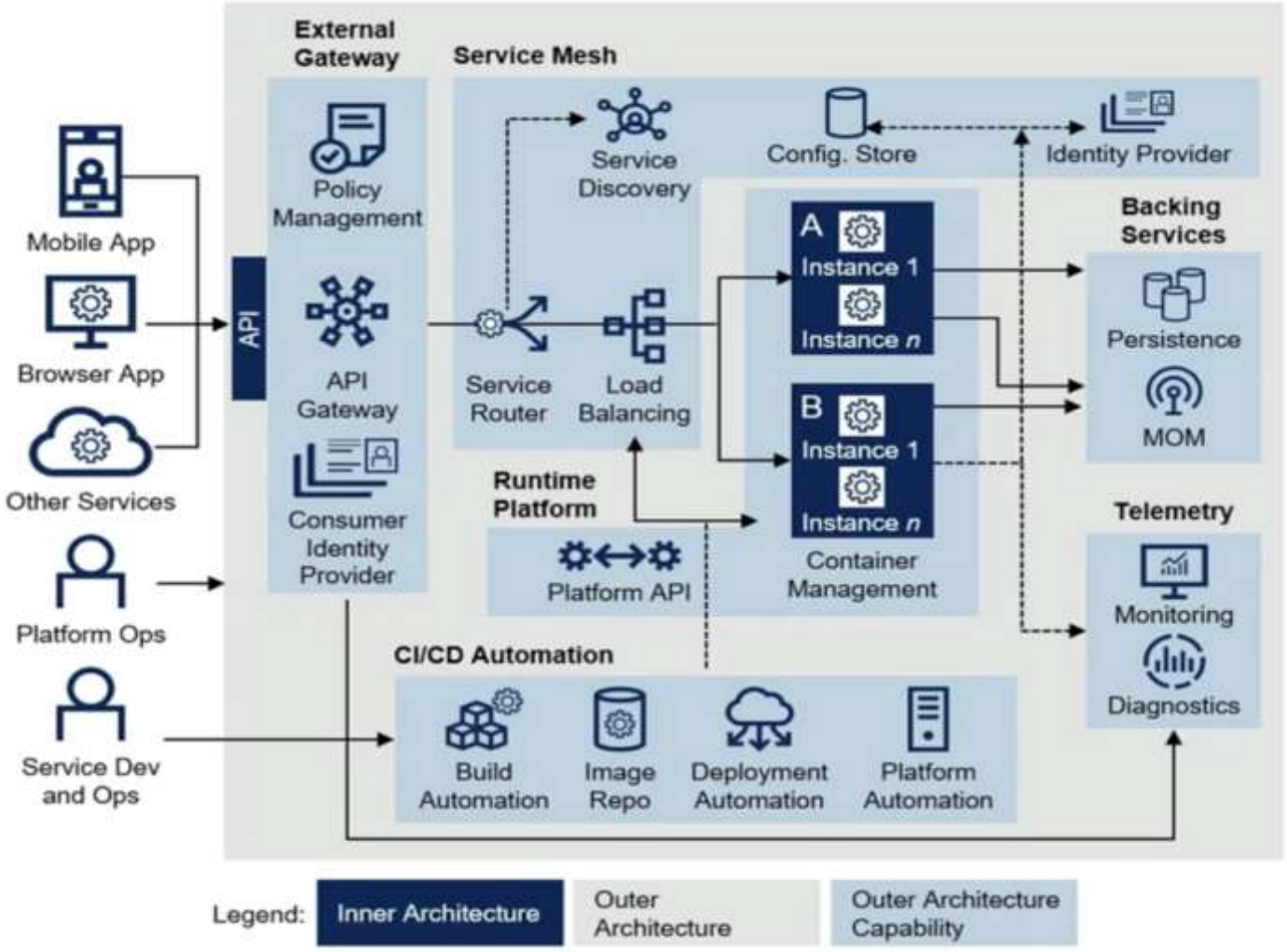


Cloud Native

- CNCF Trail Map

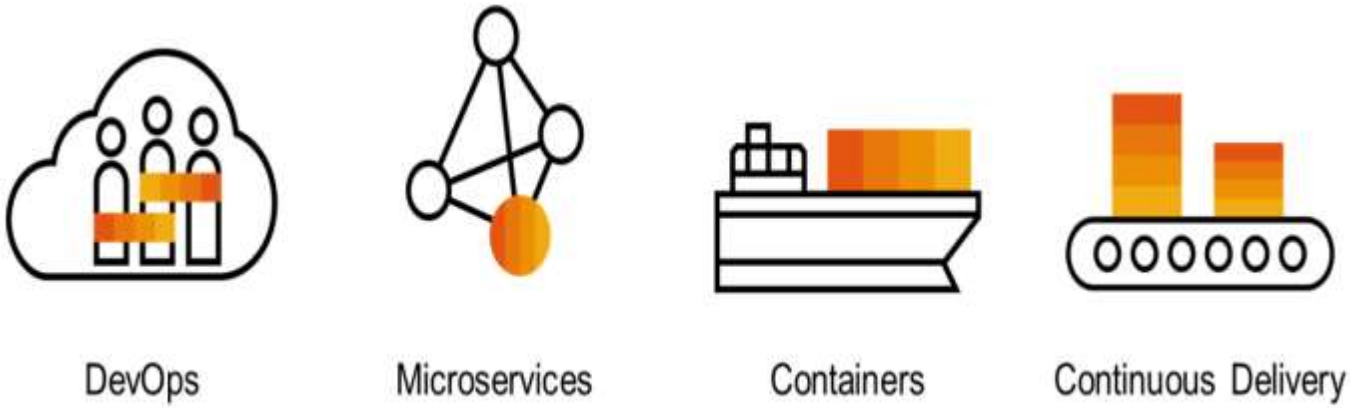
Cloud Native

- Architecture



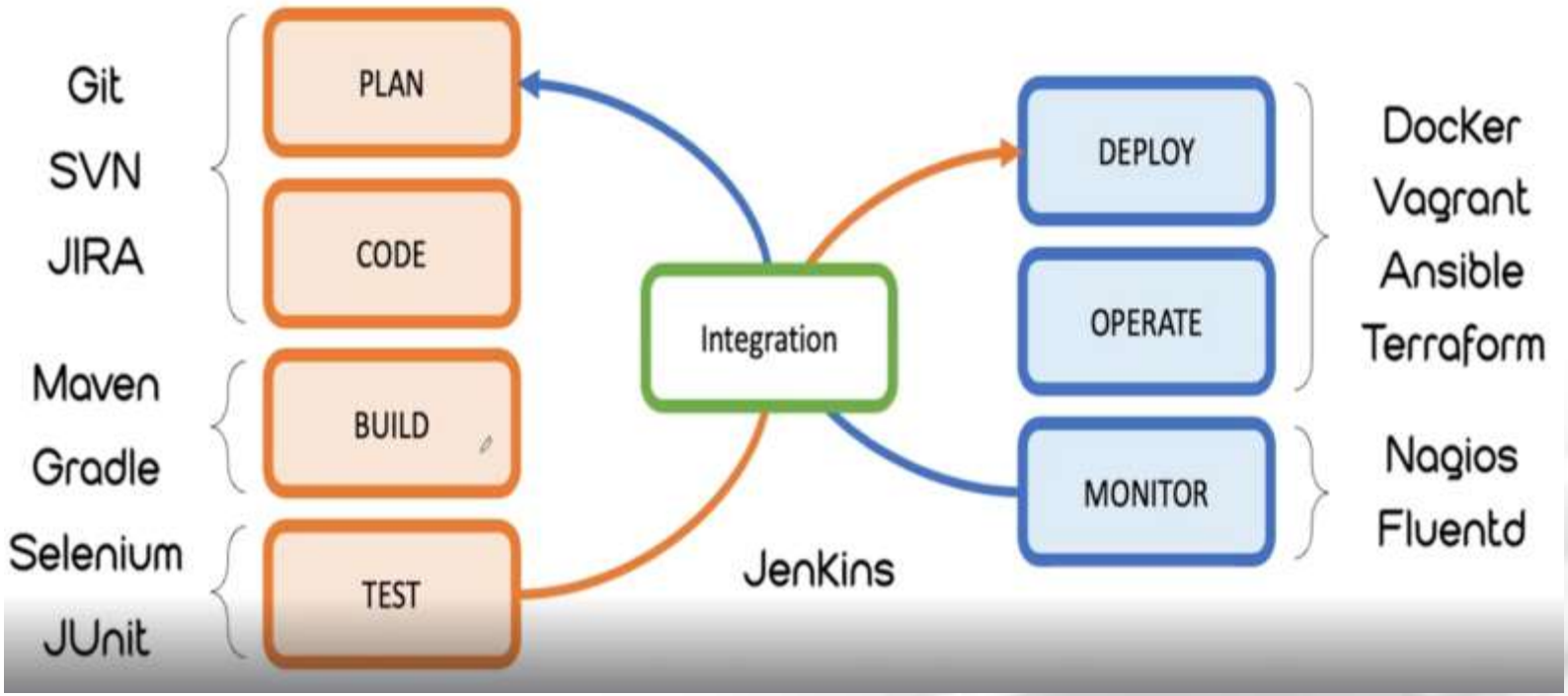
Pillars of Cloud Native

- Pillars of Cloud Native



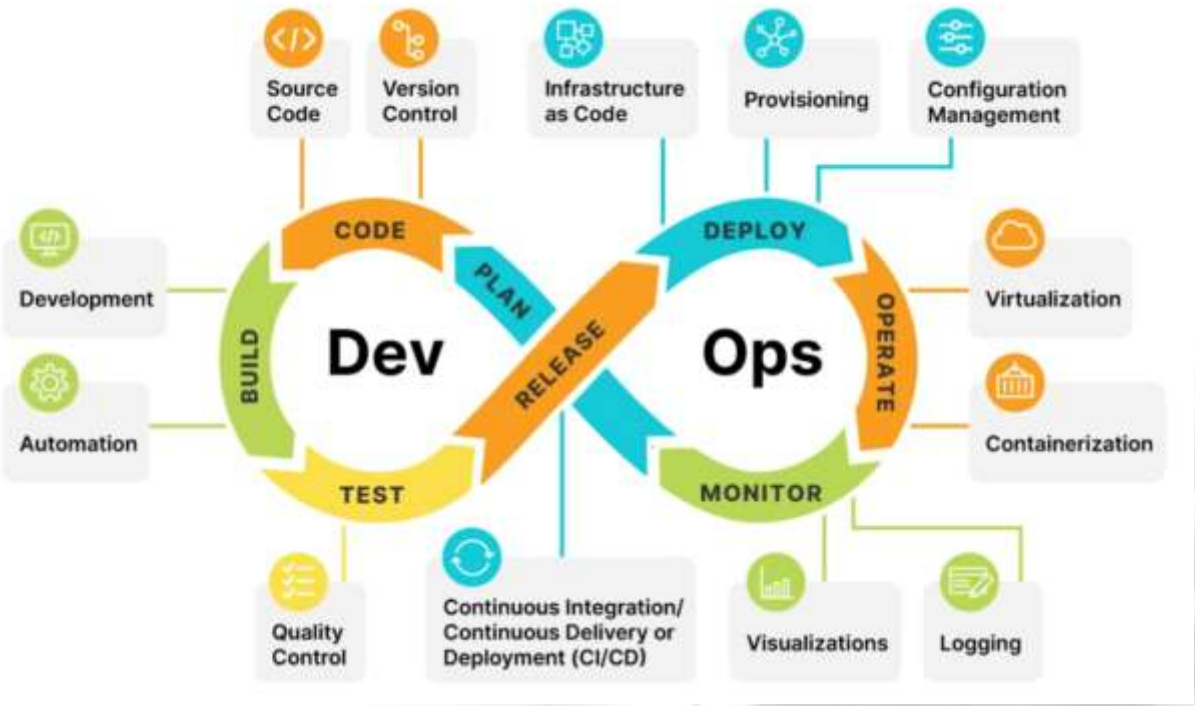
Pillars of Cloud Native

- DevOps



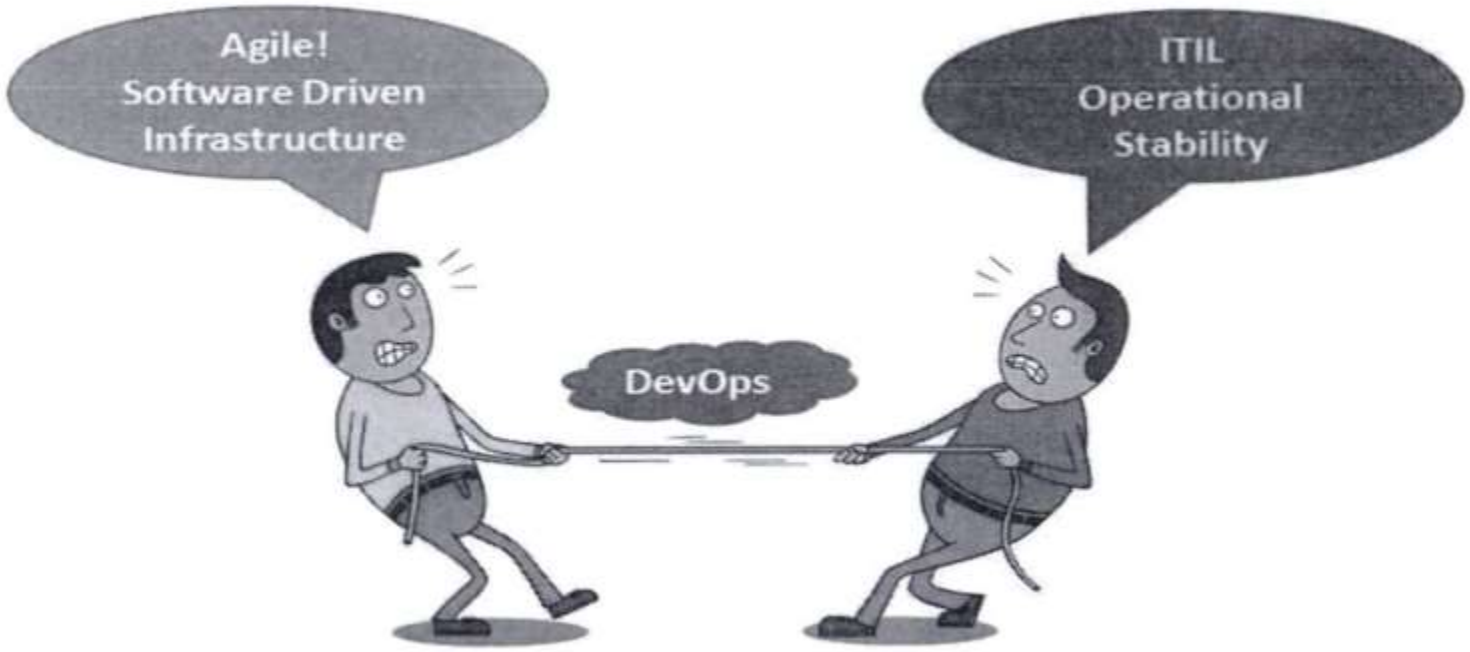
Pillars of Cloud Native

- DevOps



Pillars of Cloud Native

- DevOps
  - ✓ 서비스 개발 vs 서비스 운영



Pillars of Cloud Native

- DevOps
  - ✓ 애자일 방법론
    - 애자일은 2001년부터 시작
    - 소프트웨어 프로젝트를 일련의 선형적 순서로 구성하는 워터폴(Waterfall) 방식의 프로젝트 관리 기법의 단점을 보완하기 위해서 등장
    - 소프트웨어 개발자 그룹이 애자일 소프트웨어 개발에 대한 선언문(The Manifesto for Agile Software Development)을 작성
    - 반복적인 개발 주기와 자기 조직화 팀을 강조하는 일련의 관행
    - Agile manifesto
      - 개인과 개인 간의 상호 작용이 프로세스 및 툴보다 우선
      - 작동하는 소프트웨어가 포괄적인 문서보다 우선
      - 고객과의 협업이 계약 협상보다 우선
      - 변화에 대응하는 것이 계획을 따르는 것보다 우선

Pillars of Cloud Native

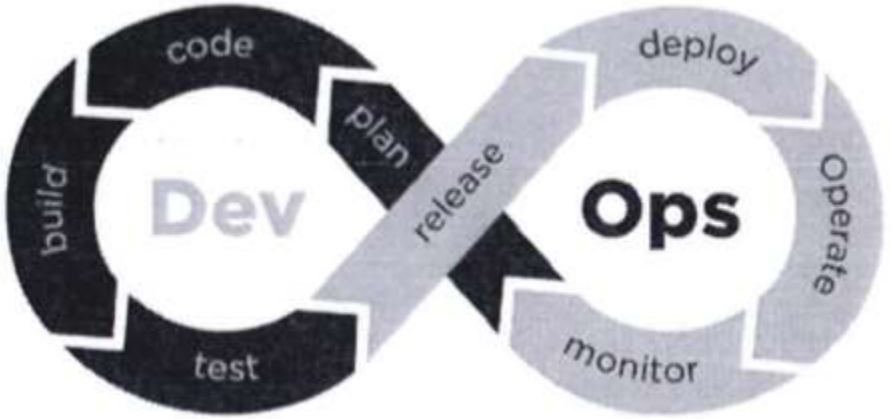
- DevOps
  - ✓ ITIL(IT Infrastructure Library)
    - 1980년대 영국 정부의 CCTA(Central Computer and Telecommunications Agency)에서 IT 인프라 운영의 최고 사례를 모아 발간
    - 2000년대 ITIL V2 부터 IT 인프라 운영관리 de facto로 전세계 IT서비스 기업의 운영 방법론으로 자리 잡음(ITSM)
    - 2019년 ITIL V4 부터 Agile, Lean, DevOps등의 방법론과의 연계성을 강화한 Service Value System 개념으로 전체 체계(Framework) 변화

Pillars of Cloud Native

- DevOps
  - ✓ 개요
    - DevOps는 IT 서비스 설계, SW 개발, 릴리즈 및 운영에 이르기까지 전체 서비스 수명 주기에 함께 참여하는 IT 서비스 운영 및 개발 엔지니어의 업무 방식
    - DevOps는 팀이 애플리케이션 개발에서 프로덕션 운영에 이르는 전체 프로세스를 소유하는 방법론
    - 일련의 기술 구현을 넘어 문화와 프로세스의 완전한 변화를 요구
    - DevOps는 전체 기능이 아닌 작은 구성 요소에서 작업하는 엔지니어 그룹을 요구하여 일반적인 오류 소스인 핸드 오프를 줄임
    - DevOps는 소프트웨어의 개발(Development)과 운영(Operations)의 합성어로서 소프트웨어 개발자와 정보 기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화
    - DevOps는 소프트웨어 개발 조직과 운영 조직간의 상호 의존적 대응이며 조직이 소프트웨어 제품과 서비스를 빠른 시간에 개발 및 배포하는 것이 목적
    - 기술이 아닌 철학이며 변경이 아닌 변화

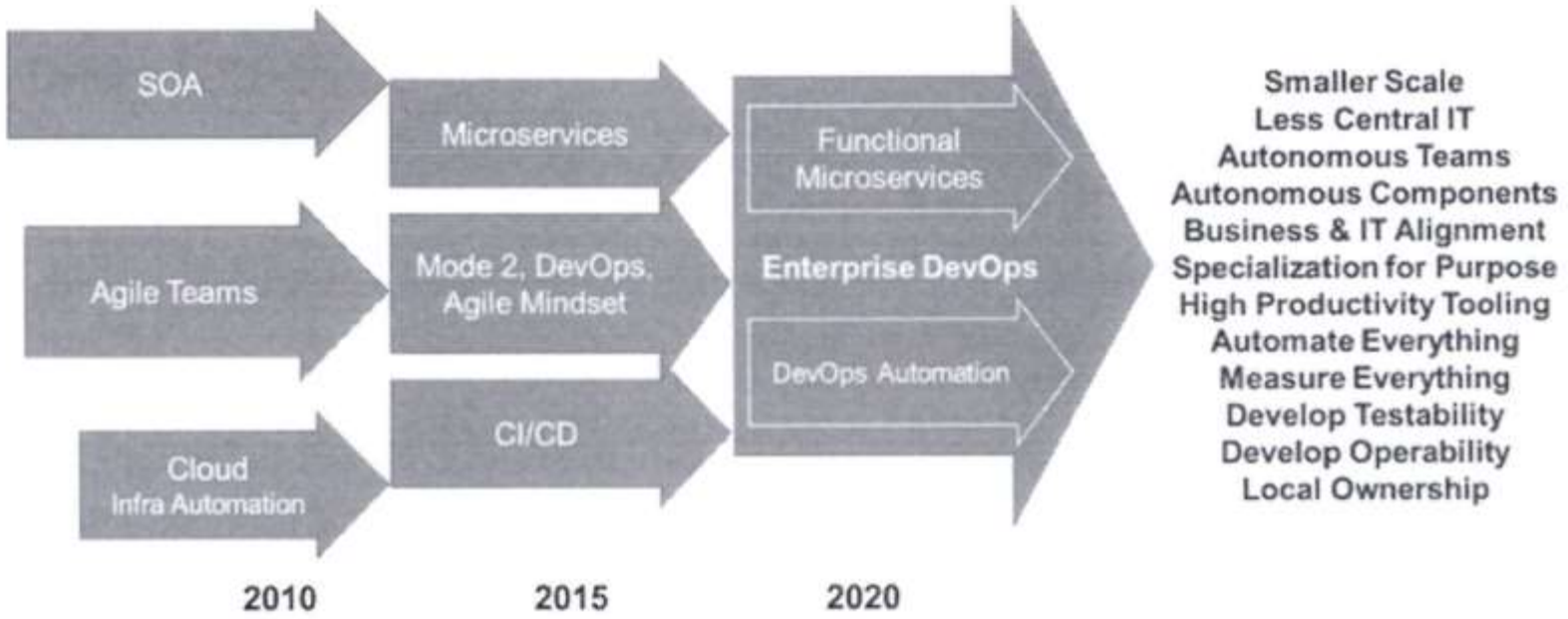
Pillars of Cloud Native

- DevOps
  - ✓ DevOps Tool Chain
    - 계획 - 목적을 수행하기 앞서 방법이나 절차 등을 미리 생각하여 계획
    - 코드 - 코드 개발 및 검토, 버전 관리 도구, 코드 병합
    - 빌드 - 지속적 통합(CI) 도구, 빌드 상태
    - 테스트 - 테스트
    - 패키지 - 애플리케이션 디플로이 이전 단계
    - 릴리스 – 변경 사항 관리, 릴리스 승인, 릴리스 자동화
    - 구성 - 인프라스트럭처 구성 및 관리, IaC (Infrastructure as Code) 도구
    - 모니터링 - 애플리케이션 성능 모니터링, 최종 사용자 경험



Pillars of Cloud Native

- DevOps
  - ✓ DevOps 로의 변화

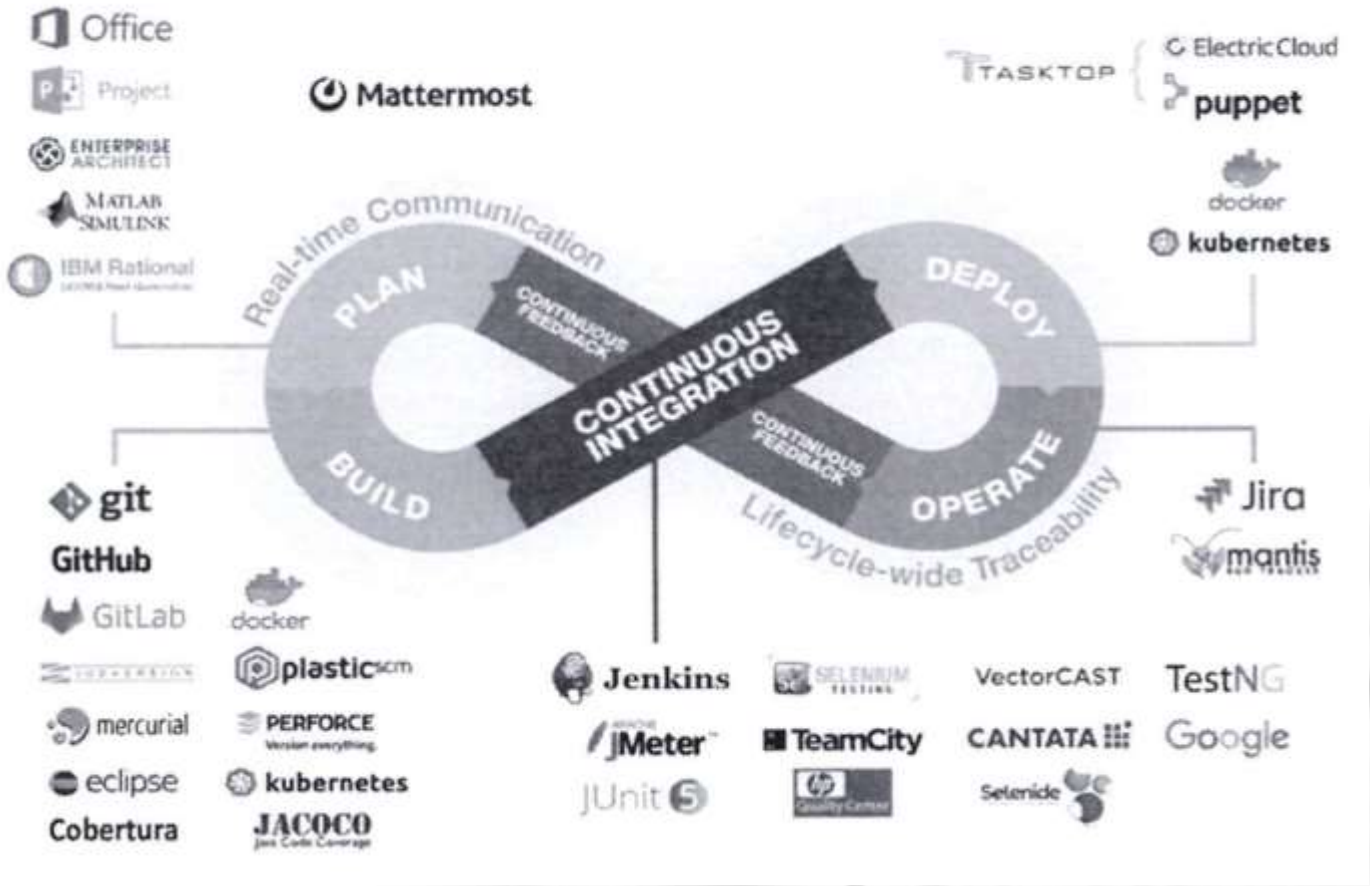


Pillars of Cloud Native

- DevOps
  - ✓ DevOps 이점
    - 속도 (Agile) - Time to Market, Microservices
    - 신속한 제공 (Delivery) - CI/CD
    - 안정성 (Reliability) - 모니터링 과 로깅
    - 확장 (Scale) - IaC(Infra as code, 코드형 인프라)
    - 협업 강화 (Collaboration) - CLAMS
    - 보안(Security) - Policy as code, 코드형 정책

Pillars of Cloud Native

- DevOps
  - ✓ DevOps 생태계

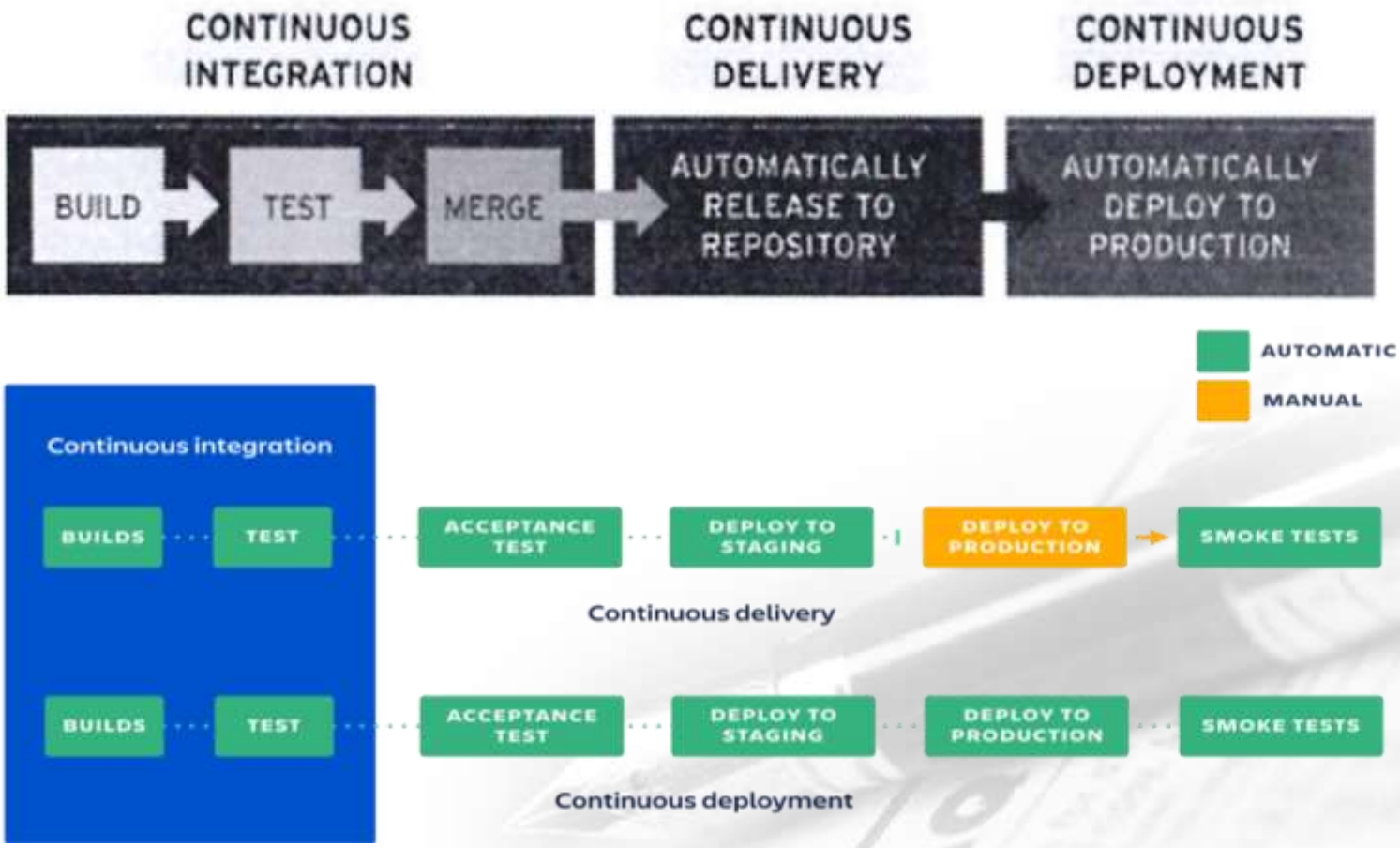


Pillars of Cloud Native

- CI/CD
  - ✓ 개요
    - CI/CD의 기본 개념은 지속적인 통합(Continuous Integration), 지속적인 서비스 제공(Integration Delivery), 지속적인 배포(Continuous Deploy)
    - CI/CD는 애플리케이션 개발 단계를 자동화하여 애플리케이션을 보다 짧은 주기로 고객에게 제공하는 방법
    - CI/CD는 새로운 코드 통합으로 인해 개발 및 운영 팀에 발생하는 문제(인테그레이션 헬 – integration hell)를 해결하기 위한 방법

Pillars of Cloud Native

- CI/CD
  - ✓ 개요

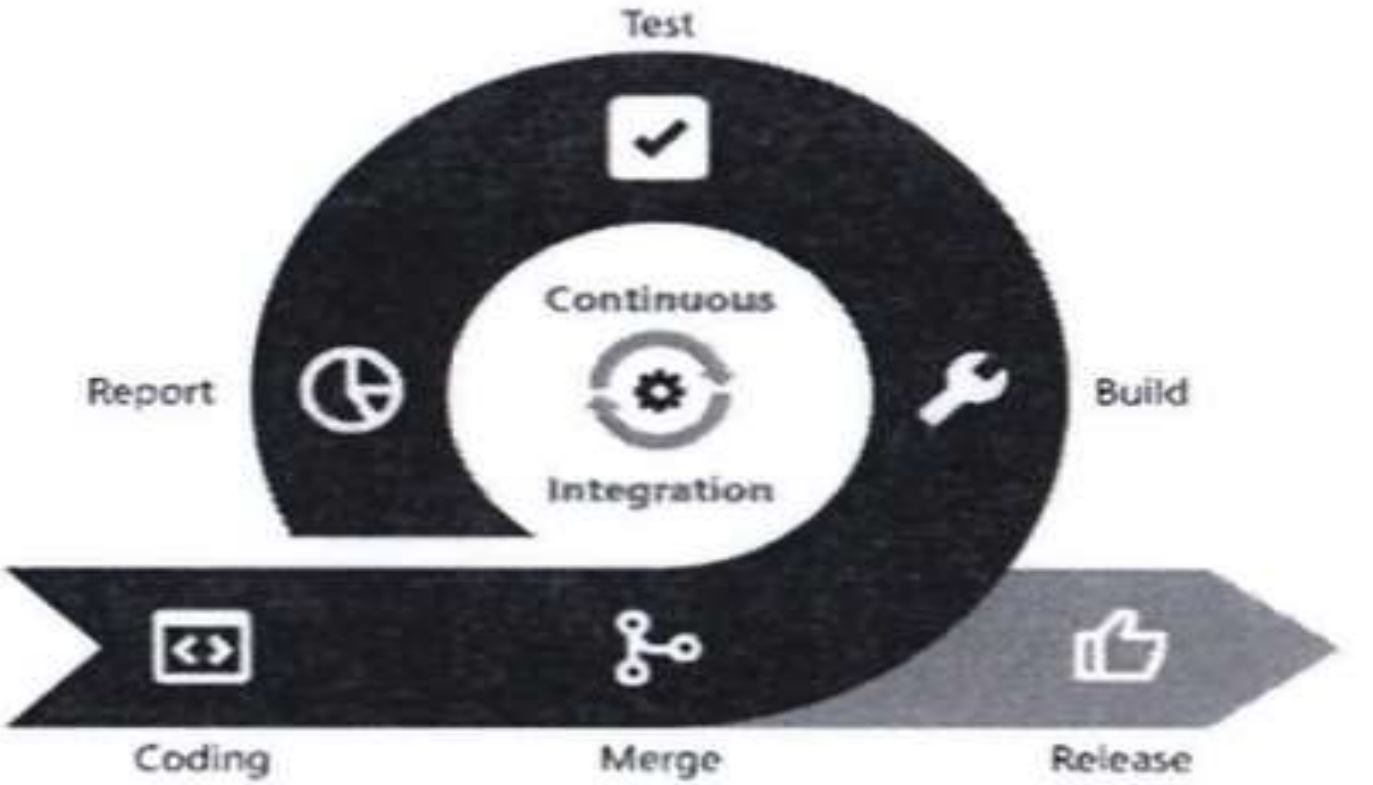


Pillars of Cloud Native

- CI/CD
  - ✓ 지속적인 통합(Continuous Integration)
    - 지속적인 통합(Continuous Integration)은 개발 팀이 작은 변경 사항을 구현하고 코드를 버전 제어 리포지토리에 자주 체크인하도록 하는 코딩 철학이자 일련의 관행
    - 최신 애플리케이션은 다양한 플랫폼과 도구에서 코드를 개발해야 하기 때문에 팀은 변경 사항을 통합하고 검증하기 위한 메커니즘이 필요
    - 기술 목표는 애플리케이션을 빌드, 패키지 및 테스트하는 일관되고 자동화된 방법을 설정하는 것
    - 통합 프로세스의 일관성을 유지하면 팀이 코드 변경을 더 자주 커밋할 가능성이 높아져 공동 작업과 소프트웨어 품질이 향상
    - 소프트웨어 공학에서 지속적 통합(continuous integration, CI)은 지속적으로 품질 관리(Quality Control)를 적용하는 프로세스를 실행하는 것
    - 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 맞추어짐

Pillars of Cloud Native

- CI/CD
  - ✓ 지속적인 통합(Continuous Integration)

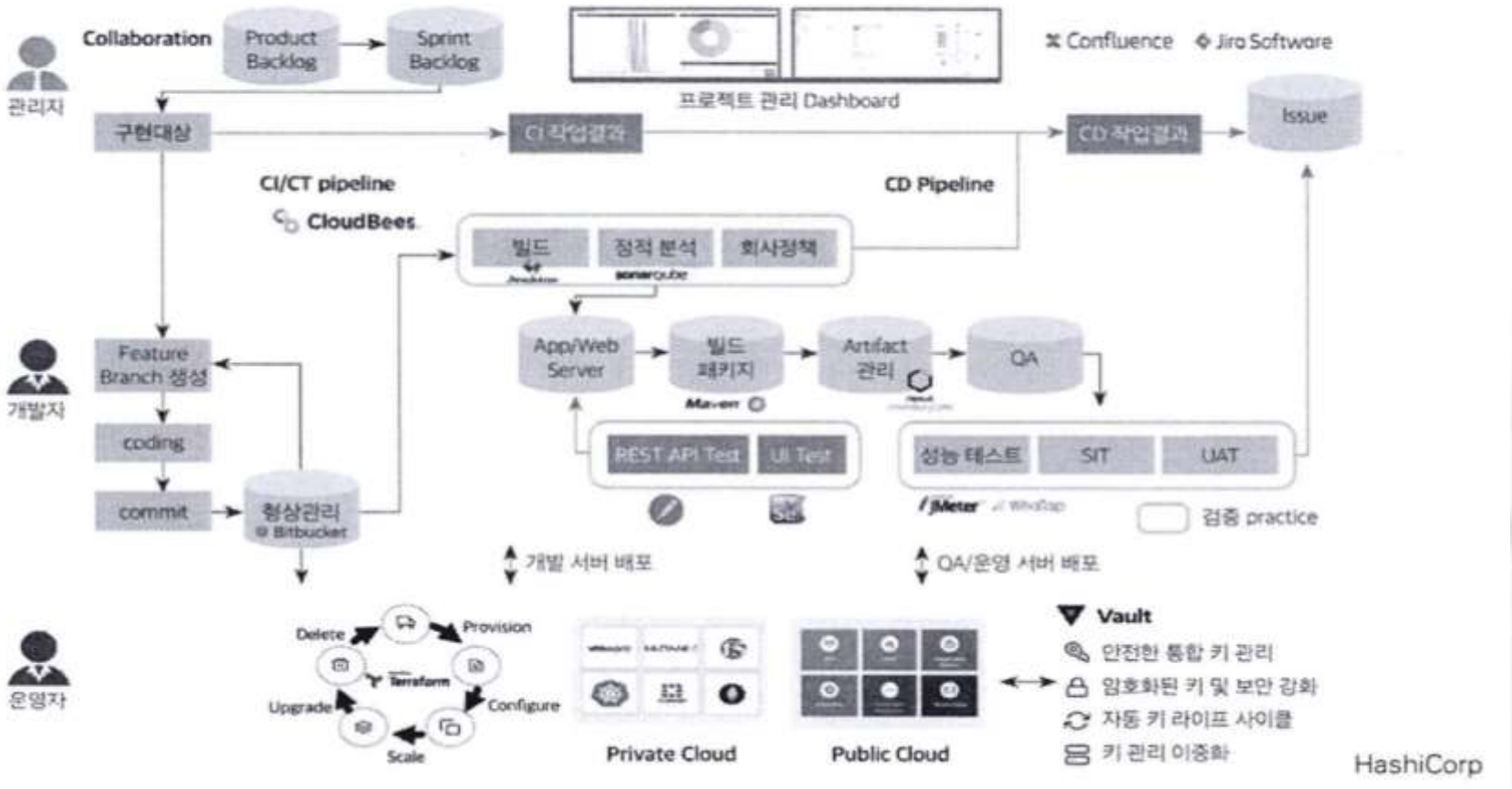


Pillars of Cloud Native

- CI/CD
  - ✓ 지속적인 제공(Continuous Delivery)
    - 개발자들이 애플리케이션에 적용한 변경 사항이 버그 테스트를 거쳐 리포지토리(GitHub 또는 컨테이너 레지스트리)에 자동으로 업로드되는 것
    - 운영 팀은 이 리포지토리에서 애플리케이션을 실시간 프로덕션 환경으로 배포
    - 개발 팀과 비즈니스 팀 간의 가시성과 커뮤니케이션 부족 문제를 해결
    - 지속적인 제공은 최소한의 노력으로 새로운 코드를 배포하는 것
  - ✓ 지속적인 배포
    - 다른 의미의 CD(Continuous Deployment)란 개발자의 변경 사항을 리포지토리에서 고객이 사용 가능한 프로덕션 환경까지 자동으로 릴리스하는 과정
    - 애플리케이션 제공 속도를 저해하는 수동 프로세스로 인한 운영 팀의 프로세스 과부하 문제를 해결
    - 지속적인 배포는 파이프라인의 다음 단계를 자동화함으로써 지속적인 제공이 가진 장점을 활용

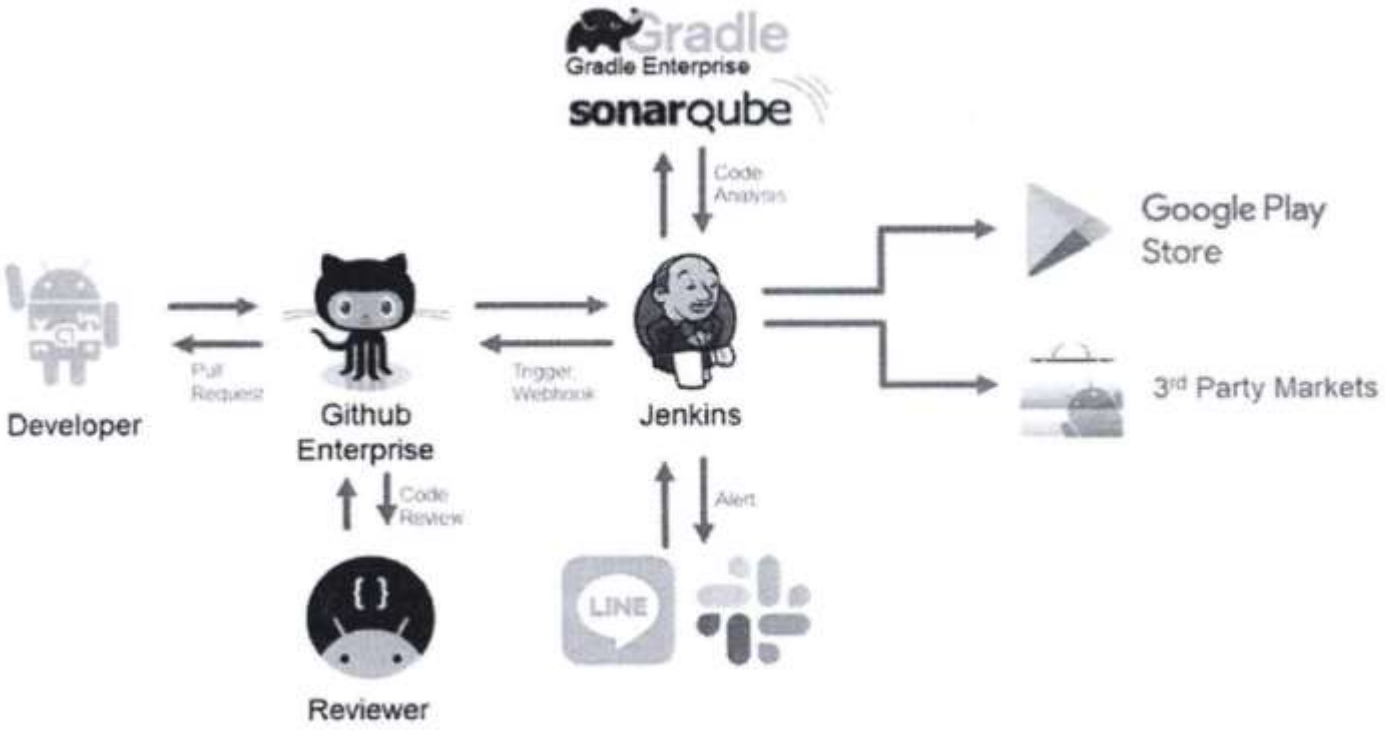
Pillars of Cloud Native

- CI/CD
  - ✓ CI/CD Flow
    - IaC(Infrastructure as Code) 측면



Pillars of Cloud Native

- CI/CD
  - ✓ CI/CD Flow
    - 실업무 측면

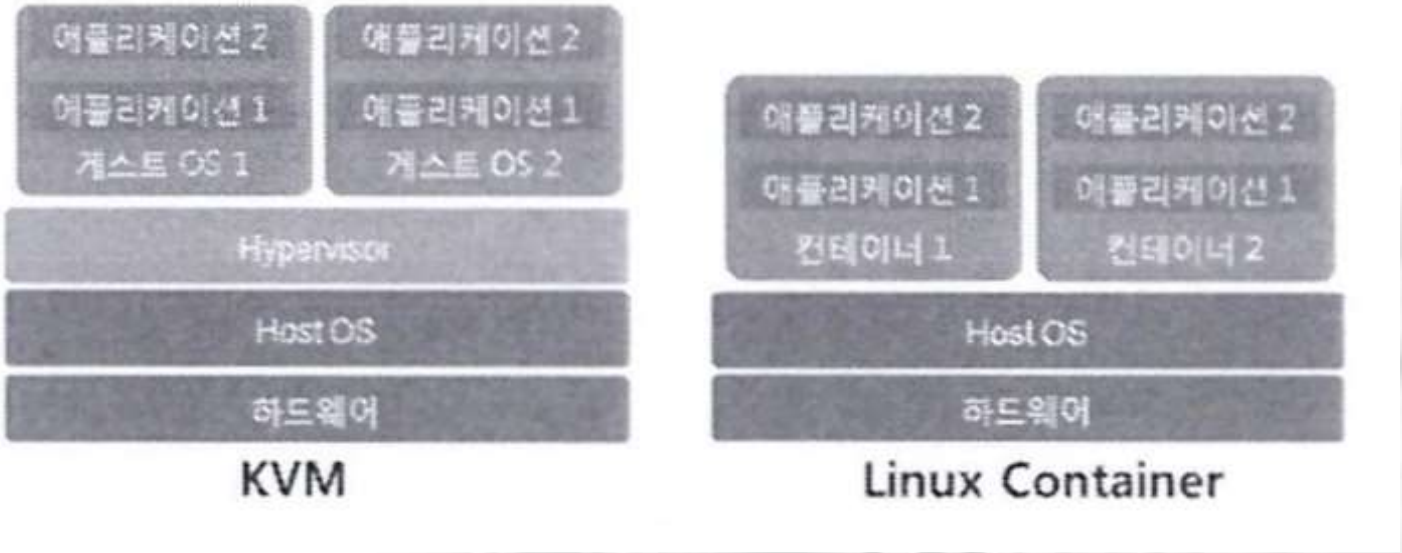


Pillars of Cloud Native

- Micro Service
  - ✓ 마이크로 서비스란 소프트웨어를 구축하기 위한 아키텍처이자 하나의 접근 방식으로 애플리케이션을 상호 독립적인 최소 구성 요소로 분할
  - ✓ 모든 요소를 하나의 애플리케이션에 구축하는 전통적인 모놀리식 접근 방식 대신 모든 요소가 독립적으로 연동되어 태스크를 수행
  - ✓ 소프트웨어 개발에 대한 이러한 접근 방식은 세분화, 경량화되어 있으며 다수의 애플리케이션 간에 유사한 프로세스를 공유하는 기능을 중시 함
  - ✓ 마이크로 서비스 아키텍처라고도 알려진 마이크로 서비스는 애플리케이션을 서비스 모음으로 구성하는 아키텍처 스타일
    - 높은 유지 관리 및 테스트 가능성
    - 느슨하게 결합됨
    - 독립적으로 배포 가능
    - 비즈니스 역량을 중심으로 구성됨
    - 소규모 팀이 소유함
  - ✓ 마이크로 서비스 아키텍처를 사용하면 크고 복잡한 애플리케이션을 빠르고 빈번하며 안정적으로 제공할 수 있는데 이를 통해 조직은 기술 스택을 발전시킬 수 있음

Pillars of Cloud Native

- ❖ Containers
  - ✓ 개요
    - Linux 컨테이너는 실행에 필요한 모든 파일을 포함하여 전체 실행(runtime) 환경에서 애플리케이션을 패키지화하고 분리하는 기술
    - 전체 기능을 유지하면서 컨테이너화 된 애플리케이션을 환경(개발, 테스트, 생산 등) 간에 쉽게 이동 가능
    - 컨테이너 파이프라인에 보안을 구축하고 인프라를 보호하여 컨테이너의 안정성과 확장성 및 신뢰성을 보장할 수 있음



Pillars of Cloud Native

- DevOps, CI/CD 실현 기술 요소