DevOps

DevOps의 시작

- **2009년 벨기에**
  Patrick Debois가 DevOpsDays 컨퍼런스(http://devopsdays.org/)를 조직하면서 처음으로 사용하기 시작했다. 이후 민첩한 소프트웨어 개발 및 운영을 위한 방법으로 DevOps가 알려지면서 많은 개발조직들에게 전파되기 시작했다.

개발팀(Dev)의 목표

- 잦은 배포와 업데이트
- 애플리케이션을 통한 쉽고 빠른 새로운 기능(리소스) 제공

운영팀(Ops)의 목표

- 프로덕션* 앱의 안정성
- 애플리케이션이 아닌 인프라 관리
- 모니터링 및 제어

DevOps가 아닌것은?

- **역할과 구조**

  DevOps는 새로운 부서(팀)가 아니다. 그렇다고 개발자가 서비스 운영을 책임지는 것도 서비스 운영자가 개발 업무를 맡는 것도 아니다.

- **도구와 솔루션**

  사람과 프로세스가 아닌 도구나 솔루션을 구매하면 DevOps를 구현할 수 있다고 생각하는 것도 잘못된 것이다. 자동화 도구를 통하여 배포 및 운영업무에서 운영자의 수동적 업무가 줄어들기는 하나 이 역시 서버 관리자가 필요하다. 결국 운영이라는 것을 어딘가에서는 해야 한다는 것이다.

새로운 개발문화로써의 DevOps

- **협력**

  새로운 개발 문화로써 인정하고 이를 조직에 정착시키는 것이 핵심이다. 개발자와 운영자는 서로 존중하며, 신뢰를 해야한다. 각자의 전문성과 책임을 인정해야 한다. 운영자는 새로운 기능을 배포하는데 개발자를 믿어줘야 하고, 개발자는 운영자가 제안하는 인프라를 신뢰하고 따라야 한다. 서로가 투명하게 공유하고 협력해 나가는 것이다. 개발자는 자신의 코드가 어떠한 영향을 주는지 알려야 하고, 운영자는 개발자에게 운영할 시스템에서 배포할 방법을 제공하는 것이다.

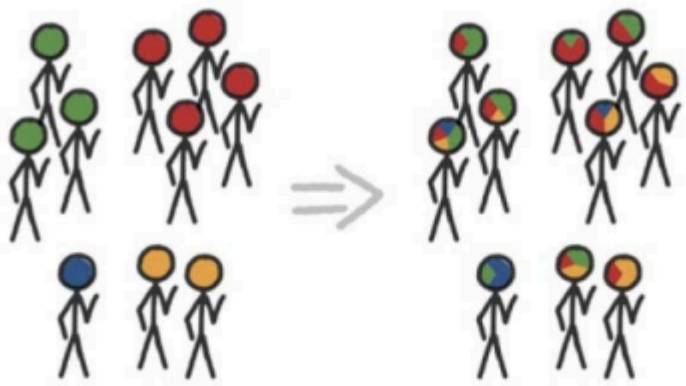
- **통합**

  하나의 프로젝트에서 개발팀은 릴리즈에 막혀 실패하고, 운영팀은 릴리즈를 막아내어 안정성 유지로 보너스를 받는 보상체계는 결국 팀 간의 장벽만 높여놓게 된다. 또한 개발에서 운영까지를 하나의 통합된 프로세스로 묶어내고 툴과 시스템을 표준화하고 통합하여 의사소통의 효율성을 확보하고 매뉴얼 작업을 자동화 함으로써 코드 통합, 테스트, 릴리즈 과정을 자동화 시키는 것이 필요하다.

- **체계**

  만약 서비스가 중단된다면, 누구든지 문제점을 진단하고 시스템을 복구하여 운영할 수 있는 절차를 알고 있어야 한다. 아래 그림과 같이 개발과 운영이 나뉘어진것이 아닌 DevOps라는 체계가 구성되는 것이다.

그림 3_기능교차적 Devops 팀



출처: Devops for Developers, Michael Huttermann, Apress, 2012

그림 4_개발, 품질보증, 운영의 교집합

출처: http://en.wikipedia.org/wiki/DevOps#Devops_Days

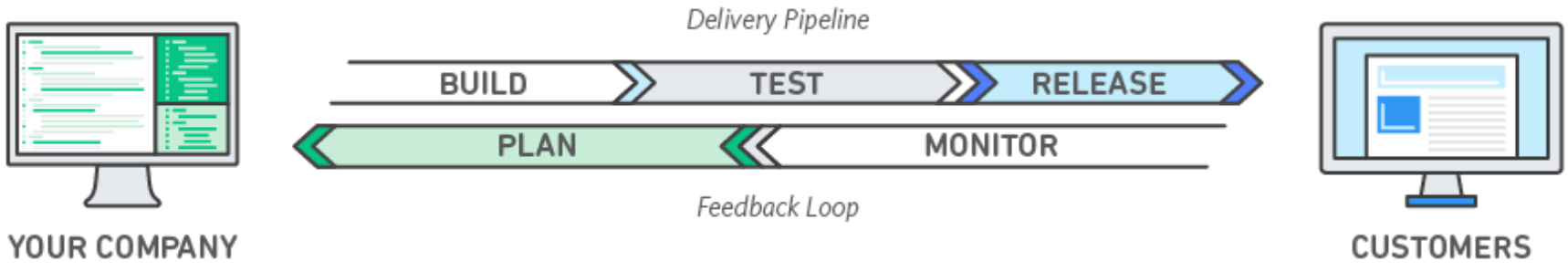
DevOps는 어떻게 정의될까?

주요 클라우드 서비스들이 말하는 공통점에서 그 답을 찾을수 있었다. AWS에서는 "조직의 역량을 향상시키는 문화 철학, 방식 및 도구의 조합"으로 보다 빠른 서비스를 Azure에서는 개발, 품질, 보안, 운영 등 단적인 역할들이 하나의 합집합으로써 비즈니스 목표를 빠르게 달성하는 것이고, Google Cloud에서는 소프트웨어 개발 및 배포 실적을 개선한다고 정의했다. 따라서 DevOps(Development + Operations)라는 합성어는 소프트웨어 개발자들과 IT종사자들 사이의 의사소통, 협업, 융합을 강조한 소프트웨어 개발 방법론이며, "개발"과 "운영"에 대한 용어와 개념의 교차 기능 조합으로 공유 소유권, Workflow 자동화 및 신속한 피드백과 같은 주요 원칙으로 정의할 수 있을 것이다.
즉, 잦은 배포와 업데이트를 하지만 쉽고 빠른 서비스 기능을 제공함과 동시에 앱의 안정성을 높이고 모니터링하며 인프라 관리까지 할 수 있는 조직이 되는 것으로 생각된다.

주요 서비스들의 DevOps 정의

- **AWS (Amazon Wep Service)**

  애플리케이션과 서비스를 빠른 속도로 제공할 수 있도록 조직의 역량을 향상시키는 문화 철학, 방식 및 도구의 조합입니다. 기존의 소프트웨어 개발 및 인프라 관리 프로세스를 사용하는 조직보다 제품을 더 빠르게 혁신하고 개선할 수 있습니다. 이러한 빠른 속도로 통해 조직은 고객을 더 잘 지원하고 시장에서 좀 더 효과적으로 경쟁할 수 있습니다.



- **Microsoft Azure**

  개발(Dev)과 운영(Ops)의 합성어인 DevOps는 고객에게 지속적 가치를 제공하도록 지원하는 사람, 프로세스, 기술의 합집합. 개발, IT 운영, 품질 엔지니어링, 보안 등 이제껏 서로 단절되었던 역할들이 서로 조율하고 협업하여 더욱 안정적이고 뛰어난

제품을 생산할 수 있도록 지원하며, 고객 요구사항에 보다 효과적으로 대응하고, 더욱 안심하고 애플리케이션을 빌드하며, 비즈니스 목표를 더 빨리 달성할 수 있도록 한다.

- **Google Cloud Platform**

  소프트웨어 배포 속도를 높이고 서비스 안정성을 개선하며 소프트웨어 이해관계자 간 공유 소유권을 구축하기 위한 조직적, 문화적 방법론으로 소프트웨어 개발 및 배포 실적을 개선

- **위키피디아**

  DevOps(Development + Operations)라는 합성어는 소프트웨어 개발자들과 IT종사자들 사이의 의사소통, 협업, 융합을 강조한 소프트웨어 개발 방법론이며, "개발"과 "운영"에 대한 용어와 개념의 교차 기능 조합으로 공유 소유권, Workflow 자동화 및 신속한 피드백과 같은 주요 원칙을 특징으로 합니다.

Dev팀과 Ops팀의 목표는 어떨까?

개발팀은 잦은 배포와 업데이트를 통하여 쉽고 빠른 새로운 기능을 제공하여야 하며 운영팀의 목표는 프로덕션앱의 안정성을 높이고 사용자 인프라 관리 및 모니터링이 우선시 된다.
따라서, 개발팀의 새로운 기능을 추가하기 위한 노력이 운영팀에서는 새로운 기능으로 인한 오류와 안전성 문제로 다가올 수 있다.
예로 들면 하나의 프로젝트에서 개발팀은 릴리즈에 막혀 실패하고, 운영팀은 릴리즈를 막아내어 안정성 유지로 보너스를 받는 보상체계가 돌아간다면 결국 팀 간의 장벽만 높여놓게 된다.

이렇게 개발팀과 운영팀의 목적과 방향성이 상충된다고 여겨진다.

DevOps 실현을 위해, 기술이 필요한 부분은?

개발자가 새로운 소프트웨어 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 Deploy하는 과정에서 쉽고 빠른 서비스를 위한 기술이 필요하다.

DevOps를 문화로 풀어야 할 부분

서비스 중에 문제가 생길 수 있고 애초에 계획 단계에서 부터 개발과 운영의 상충되는 부분으로 인해 지속적 배포에 걸림돌이 될 것 같다.

이러한 문제들은 DevOps라는 새로운 개발문화로써 인정하고 이를 조직에 정착시키는 것이 핵심이다. 개발자와 운영자는 서로 존중하며, 신뢰를 해야한다. 각자의 전문성과 책임을 인정해야 한다. 운영자는 새로운 기능을 배포하는데 개발자를 믿어줘야 하고, 개발자는 운영자가 제안하는 인프라를 신뢰하고 따라야 한다. 서로가 투명하게 공유하고 협력해 나가는 것이다. 개발자는 자신의 코드가 어떠한 영향을 주는지 알려야 하고, 운영자는 개발자에게 운영할 시스템에서 배포할 방법을 제공하는 것이다.
만약 서비스가 중단된다면, 누구든지 문제점을 진단하고 시스템을 복구하여 운영할 수 있는 절차를 알고 있어야 한다. 이렇게 된다면 개발과 운영이 나뉘어진것이 아닌 DevOps라는 체계가 구성되는 것이다.

프로덕션 앱이란?

소프트웨어를 출시할 때 베타 버전, 개발자 버전이 있듯, 프로덕션은 고객들이 사용할 수 있는 안정적인 공식 버전을 의미합니다.

Google Play 출시를 위한 버전은?

- 공개 테스트 : 모든 테스터에게 제공
- 비공개 테스트 : 개발자가 선택하는 제한된 인원에게 제공
- 내부 테스트 : 개발자가 선택하는 최대 100명의 테스터에게 제공
- 프로덕션 : 선택한 국가의 모든 사용자에게 제공