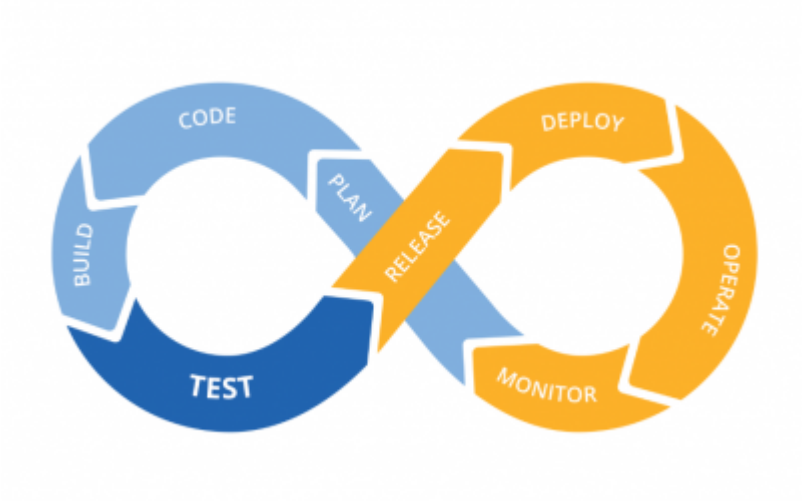
# DevOps

- description : DevOps
- author : 밤즌
- email : bjlee@repia.com
- lastupdate : 2022-03-16

## Intro

### 데브옵스(Devops)란?

- 데브옵스(DevOps)는 개발(Development)과 운영(Operations)의 합성어
- 개발과 운영의 경계를 허물고 하나의 팀으로서 소통, 협업 및 통합을 강조하는 개발 환경이나 문화를 뜻 함.




## Case Study

### 데브옵스(DevOps)의 이해

- 개발팀과 IT 운영팀의 작업에 대한 공유 방식으로 조직 내 여러 팀 간 의사소통과 협업을 활성화 시키는 운영 방식
  - 개발팀과 IT 운영팀이 나누어져 있는 상황에서, 개발팀은 프로그램의 개발만을 IT 운영팀은 기존 시스템의 운영만을 담당하므로 개발자와 운영자에 대한 구분과 불신이 나타나게 되고 이렇게 되면 개발한 프로그램에 대한 배포와 류를 잡아 프로그램을 안정화하는 과정이 오래 걸려 비효율을 초래하게 되는데 개발자와 IT 운영자를 통합하는 Devops를 적용하여 개발부터 운영까지 한번에 처리해보자는 것이다.
- 장점
  1. 개발팀과 운영팀간의 의사소통 증가로 생산성이 증가
  2. 한 곳에서 개발부터 검증, 배포까지 전체를 담당하게 되어 개발과 배포 속도가 빨라짐
  3. 구성원에게 개발 책임감과 코드의 소유권을 높여줘 개발 프로세스 간소화

- 단점
    1. 다양한 팀이 모여 업무 역할이 변경되기 때문에 활성화되기 위해서는 충분한 시간이 필요
    2. 코드를 자주 배포할 필요가 없다면 비용만 늘어남
    3. 포괄적인 자동화 도구가 필요함

Devops는 도구가 아니라 일을 이렇게 하자는 방법론이지만 주요 구성 내용과 도구의 예시는 다음과 같다.

- 코드 리포지토리(Code Repositories)
    - 코드 수정 내용 보관 및 타 개발자들이 변경된 코드 적용
    - ex) Git, SVN
- 아티팩트 리포지토리(Artifact Repositories)
    - 프로젝트 수행 시 만든 산출물(설계 문서, .jar 파일 등)을 버전별로 보관
    - ex) JFrog, Nexus Repository
- 컨테이너(Containers)
    - 가상 환경에 접속한 후 해당 환경에 맞추어 세팅해주면 개별 시스템별로 세팅해줄 필요 없이 컨테이너에 맞춘 대로 개별 시스템에 동일하게 세팅이 가능해짐
    - ex)Docker, Microsoft Hyper-V
- CI/CD
    - CI(Continuous Integration) 지속적인 통합
    - CD(Continuous Deploy or Continuous Delivery) 지속적인 배포
    - 품질관리를 위해 지속적인 빌드, 테스트 과정 지원, 지속적인 배포 가능한 기능을 포함
    - ex)Jenkins, Travis CI

## 데브옵스(DevOps)의 이점

1.속도
Micro Service Architecture(단일 애플리케이션을 작은 서비스의 집합으로 구축하며,
각 프로세스는 자체 프로세스에서 실행되고 잘 정의된 인터페이스를 통해 다른 서비스와 연결되는 설계기법) 및 CD(Continuous Delivery)등을 사용하여
팀에서 서비스를 주도적으로 운영하여 수정된 코드들을 더 빠르게 릴리스 할 수 있다.
2.신속한 제공
CI(Continuous Integrity) 및 CD(Continuous Delivery)등을 통해 빌드에서 배포까지 자동화시켜
릴리스의 빈도와 속도를 개선하여 제품을 더 빠르게 업데이트 할 수 있다.
새로운 기능 및 버그 수정 속도가 빨라 고객의 요구에 더 빠르게 대응할 수 있게 된다.
3.안정성
CI(Continuous Integrity) 및 CD(Continuous Delivery)등을 통해 변경사항을 안전하게 작동하는지 업데이트마다 테스트 해주어
애플리케이션에 안전성 및 인프라 변경의 품질의 보장해준다.
4.확장
규모에 따라 인프라와 개발 프로세스를 운영관리하여 주며 자동화와 일관성의 지원 이점을 살려
복잡한 시스템이나 변화하는 시스템을 효율적으로 관리할 수 있도록 해준다.
또한 코드형 인프라(버전 관리 및 CI(Continuous Integrity)과 같은 코드와 소프트웨어 개발 기술을 사용하여 인프라를 예측하고 관리하는 방식으로
클라우드의 API 중심 모델 사용을 통해 개발자와 시스템 관리자가 수동으로 리소스를 설정 및 구성할 필요 없이
프로그래밍 방식으로 큰 규모로 인프라와 상호 작용하도록 해주는 방식)등을 사용하여

개발, 테스트, 프로덕션 환경을 반복 가능하고 효율적인 방식으로 관리 할 수 있도록 하여준다.

5.협업 강화
개발팀과 운영팀은 서로 긴밀하게 협력하며, 책임을 강조하는 가치를 통하여 효과적인 팀을 구축한다.
6.보안
자동화된 규정 준수 정책 및 세분화된 제어 및 구성 관리 기술을 통하여 제어를 유지하고, 규정을 준수하며 작업을 진행해 나아가는데
보안도 지켜 나아갈 수 있도록 하여준다.

# 데브옵스 VS 기존의 접근 방식

| DevOps 진행 | 기존의 접근 방식 |
|---|---|
| – 협업 중심.<br>성공적인 DevOps는 신속하고 신뢰성있는 소프트웨어의<br>개발과 전달을 보장하기 위해 개발팀과 IT 운영팀의 성공적이고 지속적인 협업 능력에 의존한다. | – 사일로 중심.<br>기존의 접근 방식은 협업에 대해서는 “ 다짜고짜 떠넘기기 ” 에 의존한다.<br>IT 운영자는 실제 운영 환경에서 소프트웨어 배포와 관리를 담당하며,<br>개발팀은 최소한의 지원을 제공합니다. |
| – 상당한 수준의 (또는 전체적) 구조화와 자동화.<br>DevOps 진행은 개발 환경에서의 작동이 프로덕션 환경에서의 작동을 보장하는 환경 프로비저닝과 구성에서 속도, 일관성, 반복성을 제공하는 자동화에 의존합니다.<br>또한 구조적 접근 방식은 오류 복구를 반복 가능한 자동화만큼 빠르게 바꿔주므로 롤백과 복구가 편해집니다. | – 눈송이 중심 및 대부분 수동.<br>기존의 접근 방식은 프로젝트를 제대로 수행하거나 신뢰 가능하게 반복하거나 신속하게 진행하기 어렵습니다.<br>또한 개발 및 프로덕션 인프라스트럭처와 구성 패리티를 신속하고 일관되게 프로비전하는 능력이 부족하므로 많은 문제에 봉착하는 경향이 있습니다. |
| – 셀프 서비스 중심.<br>DevOps 중심 조직은 협업 및 자동화 프레임워크를 구축해 개발자와 IT 운영자에게 상대방에게 방해되지 않고 독립적으로 일하도록 자율권을 줍니다.<br>예를 들어 개발자는 IT 운영자의 직접적인 프로비저닝을 기다리지 않고 신속하게 개발/테스트 환경을 자체적으로 프로비전할 수 있습니다. | – “ 정보 기술(IT) 티켓 ” 중심.<br>기존의 엔터프라이즈 접근 방식에서 IT 운영자는 쉽게 자동화 가능한<br>IT 티켓 관리 작업을 수행하고 반복적이고 복잡한 수동 프로비저닝과 구성을 진행해야 합니다.<br>그 결과 프로비저닝, 배포, 크기 조정, 다른 소프트웨어 전달 및 관리 활동이 상당히 복잡해지고 지연됩니다. |
| – 비즈니스 중심.<br>DevOps 조직은 공동으로 비즈니스 성공을 추진하는데 집중한다. 그러므로 소프트웨어 전달 성공에 대한 책임도 공동으로 집니다. | – 기능 중심.<br>기존의 접근 방식은 개발자와 IT 운영자가 그들 기능에 집중하되<br>전체적인 성공에 대해서는 별다른 책임을 부여하지 않았습니다.<br>그 결과 여러 에러 상황이 발생해 많은 비난을 받고 조직내에서 마찰이 발생합니다. |
| – 변경 맞춤형.<br>DevOps 진행은 신속하고 반복 가능하게 자동화하도록 디자인됩니다. 신속한 오류 복구뿐만 아니라 신속한 변경 처리도 수행하게 빌드됩니다. 신속하게 진행하기 위해 빌드되는 것입니다. | – 변경 반대.<br>기존의 접근 방식에서는 손댔다가 빨리 복구할 수 없게 될까봐 프로덕션 배포를 변경하지 않습니다.<br>전통적인 접근 방식에서는 변경과 업데이트를 최소화하고 조직에게는 천천히 진행할 것을 간접적으로 권장합니다. |

- 데브옵스(DevOps) VS 기존의 접근 방식
- 데브옵스(DevOps)와 애자일(Agile)

# Term

- CI,CD
- 사일로 현상(Organizational Silos Effect): 조직 부서 간에 서로 협력하지 않고 내부 이익만을 추구하는 현상.
- 프로비저닝(Provisioning): 사용자의 요구에 맞게 시스템 자체를 제공 하는 것
  시스템 자원을 할당, 배치, 배포해두었다가 필요할 때 즉시 사용할 수 있는 상태로 미리 준비해두는 것
  인프라 자원이나 서비스, 또는 장비가 될 수도 있음.
- 패리티(parity): 데이터 한 블록 끝에 에러를 검출하기 위해 추가하는 1비트의 검사 비트

# Ref

https://aws.amazon.com/ko/devops/what-is-devops/?nc1=h_lshttps://aws.amazon.com/ko/devops/what-is-devops/?nc1=h_ls

데브옵스, DevOps, 밤즌

☐ 이 페이지 초안 작성 부탁합니다. (다른 페이지를 보거나, 주위에 물어서요~~) Convention과 Consensus가 중요해요 ) (할당된 사용자 밤즌. 마감일 2022-03-18)

From:
http://125.132.25.164/dokuwiki/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다. - 2023.12**

Permanent link:
**http://125.132.25.164/dokuwiki/doku.php?id=wiki:pm:devops&rev=1648013416**

Last update: **2022/03/23 14:30**