|  |
| --- |
|  |
| **DevOps** |
|  |
|  |
| **인턴 김 지 영** |
|  |
| **2018-09-14** |

|  |
| --- |
|  |

**목차**

[**1.** **DevOps란?** 2](#_Toc524938578)

[**1.1.** **DevOps가 나타난 배경** 2](#_Toc524938579)

[**1.2.** **DevOps정의** 2](#_Toc524938580)

[**2.** **DevOps 구조** 3](#_Toc524938581)

[**3.** **DevOps 특징** 3](#_Toc524938582)

[**4.** **DevOps 장점&단점** 4](#_Toc524938583)

[**4.1.** **DevOps 장점** 4](#_Toc524938584)

[**4.2.** **DevOps 단점** 4](#_Toc524938585)

[**5.** **DevOps 사용법** 5](#_Toc524938586)

[**5.1.** **DevOps 사용 준비** 5](#_Toc524938587)

[**5.2.** **DevOps 개발자 필요 역량** 6](#_Toc524938588)

[**5.3.** **DevOps 기반 팀의 개발 싸이클** 6](#_Toc524938589)

[**6.** **참고** 6](#_Toc524938590)

# **DevOps란?**

## **DevOps가 나타난 배경**

**기존 개발 체계**

기존 개발 체계는 개발팀과 운영팀이 나누어져 있어서 개발팀에 의해서 개발이 끝나면 시스템 테스트를 거쳐 운영팀에게 전달되고, 운영팀은 그 시스템을 배포하고 운영한다.

**기존 개발 체계 문제점**

1. 책임 회피

시스템의 기능을 변경하는 과정에서 오류가 발생했을 때, 운영팀은 개발팀이 잘못된 코드를 줬다고 주장하고, 개발팀은 테스트에서 문제없었던 것이므로 운영팀이 뭔가를 잘못 설정한 것이라고 주장한다. 서로 책임을 회피하며 문제의 원인을 찾지 못하고 비효율적인 시간을 사용하게 된다.

1. 운영 이슈에 대한 전달문제

운영팀에서는 서비스를 운영하고 있기 때문에 여러 VoC를 모아 개발팀에게 서비스 개선을 요청한다. 하지만 개발팀 입장에서는 이러한 요구사항들이 추가적인 일이 되기 때문에 거절하게 된다. 따라서 애플리케이션에 VoC를 반영하지 못하게 된다.

1. 개발팀과 운영팀의 관계 악화

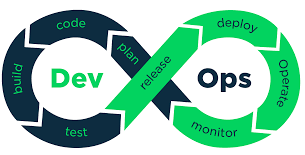
서비스가 운영 배포된 후에도, 빠른 비즈니스 환경 변화에 따라가기 위해서 애플리케이션에 많은 변경이 요구된다. 많은 변경으로 인해서 제대로된 테스트를 거치지 못하고 배포되는 애플리케이션이 발생하게 된다. 이러한 애플리케이션은 잦은 장애를 발생시키고, 이렇기 때문에 운영팀의 입장에서는 잦은 배포가 자신들이 개발팀의 실수를 뒷처리 하는 것으로 인식되기 때문에 계속해서 개발팀과 운영팀의 관계는 악화되어 간다.

이러한 개발팀과 운영팀 간에 커다란 관점 차이를 극복하기 위해 개발팀과 운영팀을 병합할 수 있는 DevOps라는 문화가 만들어졌으며, DevOps는 효율적인 IT운영을 할 수 있도록 돕는다.

## **DevOps정의**

* 소프트웨어 **개발(Development)**과 **운영(Operations)**의 합성어이다.
* 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화를 말한다.
* 소프트웨어 개발 조직과 운영 조직간의 상호 의존적 대응이다.
* 조직이 소프트웨어 제품과 서비스를 빠른 시간에 개발 및 배포하는 것을 목적으로 한다.

# **DevOps 구조**



[그림1]DevOps 구조 출처 : http://sblog.netraweb.com/article/

DevOps에서는 개발팀과 운영팀이 합쳐져서 한 팀 내에서 개발에서 테스트, 배포, 운영까지 전체 애플리케이션 수명주기에 걸쳐 작업한다.

# **DevOps 특징**

* 개발팀과 운영팀이 더 이상 사일로를 경험하지 않는다.
* 속도가 느리고 수동으로 수행되던 프로세스를 자동화한다.
* 애플리케이션을 안정적으로 빠르게 운영하고 개선하는 데 도움이 되는 기술 스택과 도구를 사용한다. (도구로는 AWS나 Atlassian 등을 이용할 수 있다.)
* 이러한 도구들을 이용해 엔지니어는 다른 팀의 도움 없이 코드 배포 또는 인프라 프로비저닝과 같은 작업을 독립적으로 수행할 수 있다.

\*사일로(Silo) : 부서 또는 조직 단위로 업무 효율화 측면에서 IT 인프라 및 자체 솔루션을 구축해서 사용하다 보면 각 부서, 사업 단위 별로 데이터가 일치하지 않는 증상을 말한다.

\*프로비저닝(provisioning) : 사용자의 요구에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요할 때, 시스템을 즉시 사용할 수 있는 상태로 미리 준비해 두는 것.

\*AWS(Amazon Web Service) : 아마존 웹 서비스는 아마존 닷컴이 제공하는 각종 원격 컴퓨팅 서비스이다. 다른 웹 사이트나 클라이언트 측 응용 프로그램에 대해 온라인 서비스를 제공하고 있다.

\*Atlassian : 소프트웨어 개발자들을 타겟으로 하는 비즈니스 엔터프라이즈 소프트웨어를 만드는 소프트웨어 기업이다. JIRA와 같은 버그 추적 시스템, 이슈 추적 시스템, Confluence와 같은 위키 도구들을 제작한다. 애자일 소프트웨어 개발을 제공하는데 초점이 맞춰져 있다.

# **DevOps 장점&단점**

## **DevOps 장점**

1. 문제해결 시간 단축

팀이 단일화되었기 때문에 서로의 팀에게 책임을 미루는 경우가 없어진다. 서로의 책임을 공유하고, 투명하게 업무를 진행하여 빠른 피드백을 제공할 수 있다. 이러한 원활한 커뮤니케이션이 가능하게 되어 더 빠르고 효율적으로 문제를 해결할 수 있다.

1. 신속한 제공

제품을 더 자주 릴리스 할 수 있으며, 더 높은 품질과 안정성을 제공할 수 있다.

1. 협업, 신뢰 강화

* DevOps에서는 주인의식, 책임과 같은 가치를 중요시 하기 때문에 좀 더 효과적인 팀을 구축할 수 있다. 개발자와 운영 팀은 협력하고, 많은 책임을 공유하며 워크플로를 결합한다. 이를 통해 비효율성을 줄이고 시간을 절약할 수 있다.

## **DevOps 단점**

1. 전체 문화를 바꾸는 것 자체가 모험이 될 수 있다.

스타트업이 아닌 이상, DevOps를 사용하기 위해선 기존의 기업 문화를 장기적으로 보고 단계적으로 변경해야만 한다. 한 두명의 DevOps 엔지니어를 뽑는다고 해서 DevOps로 갑자기 바뀔 수 있는 것은 아니다.

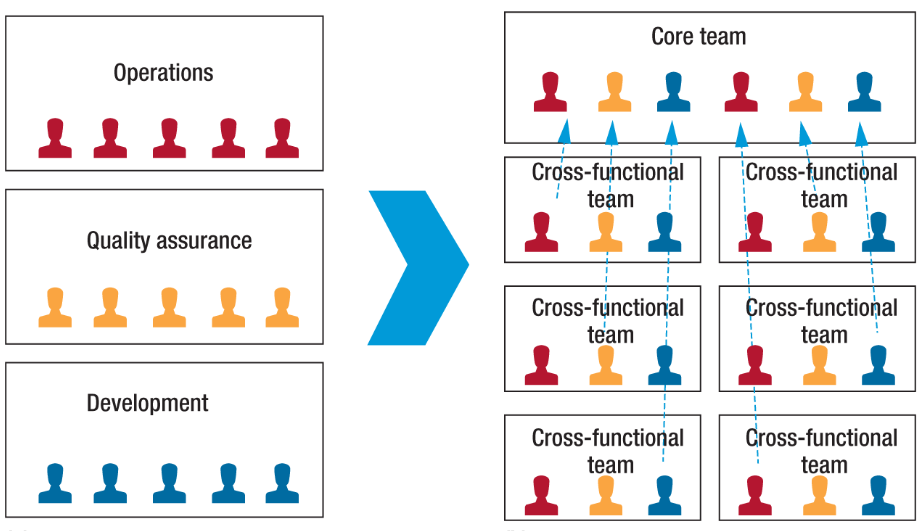
# **DevOps 사용법**

## **DevOps 사용 준비**

DevOps는 기존에 사일로에 묶여 있던 개발과 운영이라는 두 팀간의 장벽을 허무는 일이다. 두 팀의 장벽을 허물기 위해서 많은 준비들이 필요하다. 다음은 영국 정부에서 제공하는 ‘Good Habits for DevOps’이다.

1. Cross functional team

하나의 팀에 각각 다른 역할을 할 수 있는 팀원들로 구성한다.



[그림2]DveOps 팀 구성

1. Widely Shared Metris

사용자들이 서비스에 만족하는지, 운영에는 문제가 없는지 등에 대한 시스템 서비스 상태를 인지하기 위해 서비스에 대한 공통적인 지표를 만들어 놓는다.

1. Automating repetitive tasks

반복적인 작업을 자동화함으로써 팀원들이 반복적인 작업에 낭비하는 시간을 줄여 작업의 효율을 높인다. 또한, 자동화 시스템을 이용하여 테스트나 배포에 소비하는 시간을 줄여서 빠른 서비스 업데이트를 가능하게 하고, 전체 시스템에 대한 이해도를 높일 수 있다.

1. Post mortems (사후 부검)

서비스에 문제가 발생했을 때, 처리 후에 그 내용을 전체 팀원들과 공유한다. 문제에 대한 궁극적인 원인을 파악하여 다음에 같은 문제가 발생하지 않도록 할 수 있다.

1. Regular release

정기적으로 서비스를 출시한다. 정기적인 출시를 하게 되면, 언제 어떤 일을 해야하는지에 대한 계획을 확실히 세울 수 있어 빠르게 서비스의 기능을 개선할 수 있으며 고객의 VOC도 효율적으로 반영할 수 있다.

## **DevOps 개발자 필요 역량**

1. 코딩 능력
2. 다른 사람들과 잘 협업하고 커뮤니케이션 할 수 있는 능력

* DevOps는 개발과 운영 간의 소통 문제를 해결하고자 한 것이기 때문에, 다른 팀원들과의 원활한 커뮤니케이션이 매우 중요하다.

1. 프로세스를 이해하고 재정의할 수 있는 능력

## **DevOps 기반 팀의 개발 싸이클**

1. 사용자의 needs 분석. VoC(Voice of Customers) 수집
2. 사용자 요구사항 작성
3. 사용자 요구사항에 대한 범위와 우선순위 지정
4. 이해 관계자들에 대한 리포팅 및 관리
5. 다른 프로젝트와의 연관성 관리
6. 필요의 경우 오픈소스 또는 상용 평가 및 도입
7. 개발(디자인, 빌드, 테스트, 데모)
8. 테스팅
9. 배포
10. 보안 관리
11. 서비스 운영, 모니터링

# **참고**

<https://ko.atlassian.com/devops>

<https://ko.wikipedia.org/wiki/%EB%8D%B0%EB%B8%8C%EC%98%B5%EC%8A%A4>

<https://aws.amazon.com/ko/devops/what-is-devops/>

<http://bcho.tistory.com/817?category=502863>

<http://yookeun.github.io/think/2014/01/14/think-devops/>