DevOps의 성공적인 적용을 위한 가이드 라인 (1)

정두현 / ㈜씨에스리
dhc97@naver.com
컴퓨터시스템응용기술사
정보시스템수석감리원

1. DevOps의 이해

최근 트랜드로 등장하는 DevOps는 국내 SW산업의 문제점인 SI와 SM간의 상반되는 관점으로 인하여 SW개발의 한계성을 극복하기 위한 방안으로 제시되고 있다. 이러한 DevOps는 개발팀과 운영팀을 통합하여 성공적으로 프로젝트 이행뿐 아니라 연속적으로 운영조직까지의 통합으로 SI를 통해 만들어진 SW가 지속성 및 효과성, 그리고 가치를 극대화할 수 있다. SW개발은 사용자의 편의성과 효율성을 제공하여 기존 업무의 라이프 사이클을 단축시켜 의사결정을 빠르게 제공하고 이를 기반으로 기업 전략의 우위를 유지할 수 있다.

그런데 개발 프로젝트 조직과 운영조직은 목표하는 바가 다르다

가. 개발 프로젝트 조직의 목표

- 기능성: 추가 요구되는 기능을 제공한다
- 일정: 정해진 일정에 완료 되어야 한다.
- 신뢰성, 사용성: 정확한 결과를 항상 제공하고 사용자의 편의성을 제공한다.
- 효율성: 기존 업무보다 시간 단축 및 반복업무의 자동화를 제공한다.

나. 운영 조직의 목표

- 가용성: 서비스는 장애 없이 지속되어야 한다.
- 유지보수성: 변경이 용이하고 기능 추가가 용이하다.
- 이식성: 인프라의 변경 또는 H/W의 변경에 영향도가 적다.

이러한 조직간의 목표 차이는 SI의 완료 또는 인수테스트, 서비스 적용 시 많은 문제점을 발생하고 있다.
즉 안전하지 않거나 테스트가 충분하지 않았다면 운영조직에서는 서비스 적용 및 오픈에 대한 위험성을 지적하고 적용하는 시점을 늦추거나 추가 보완을 요구할 수 있다.
그러나, 개발프로젝트 조직은 일정 내에 충족되었다면 프로젝트를 완료하려고 할 것이다.

이러한 문화의 차이로 인하여 많은 개발조직의 품질관리팀장들은 한결같이 우려의 목소리를 내고 있다. 일정 내에 프로젝트가 완료되어 성공하였더라도 그 프로젝트의 산출된 SW가 활용되지 않는다면 전체적은 관점에서는 실패라고도 볼 수 있기 때문이다.

이러한 두 조직간의 실제적인 한계성에 상세하게 살펴보자면 다음과 같다

2. 개발 조직과 운영조직의 통합의 한계성

가. 일정 내 완료와 가용성의 위협

- 일정 내에 완료되었더라도 기능이 완료되었어도 품질을 보증할 수 없는 경우 가용성에 위협을 가질 수 있다.
- 실제 프로젝트가 완료 시점에서 인력의 변화가 있는 경우가 많은데 즉 프리랜서들이나 개발 담당자들의 이탈로 인한 잠재적인 리스크가 증가될수 있다.

나. 문화의 차이의 한계성

- 개발프로젝트 조직은 정해진 일정까지만 진행함으로 향후 발생되는 이슈에 대한 조치가 미흡할 수 있다.
- 운영조직은 안정성이 보장되지 않은 SW는 적용하기 보다는 충분한 테스트를 통해 일정이 늦춰지더라도 위험성이 최소화 된 SW를 적용하려고 한다.

다. 도구의 한계성

- 운영조직은 우선 사용되는 개발 도구 및 CI등의 환경을 개발프로젝트 조직과의 연계성을 고려하여 우선 공유 해야 한다.
- 개발조직은 개발도구와 배포 도구 등의 한시적인 사용 및 개발하는 단계의 부수적인 도구로 인식한다.

라. 연결 교차점의 부재

- DevOps의 가장 중요한 개발 완료단계와 테스트 단계 그리고 운영조직에서 접근하는 운영서버의 배포 및 버전업 단계의 연결 시점이 명확하지 않는 경우가 잦음
- 현실적으로 프로젝트 개발조직은 잦은 요구사항 변경으로 인해 오픈 전까지 개발하는 경우가 많으며 TDD(테스트 기반 개발)등이 이루어 지지 않은 경우에는 QA조직이 별도로 추가되고 전체 연계 라이프 사이클이 길어질 수 있다.

3. 성공적인 DevOps의 이해

가. 연결성

- 요구 분석 단계에서부터 개발 및 배포 및 유지보수까지를 하나의 싸이클로 보고 각 단계를 연결하여 담당 책임자를 연결한다.
- 즉 R&R을 구성하여 각 요구 기능은 설계자, 개발자, 운영자가 연결 함으로서 연대적인 책임을 지도록 유도한다.

나. 표준화

- 적게는 네이밍 규칙부터 크게는 배포 시점까지를 통합하여 제공할 수 있는 개발 가이드 라인의 제공이 필요함
- 인터페이스 규칙, 레가시 접속방법, SOAP, RESTful등 통신방식과 API의 표준화된 가이드를 제공해야 한다.
- 운영 환경과 일치되는 개발 환경을 사전에 제공 또는 가이드하여 개발 완료 후 적용및 운영팀의 활용이 용이하게 제공해야한다.

다. 기능 교차점

- 개발 완료 후 테스트 및 배포의 라이프 사이클을 유기적으로 연결하여 커뮤니케이션 및 인수 시험 과 배포 수행 에 대한 포커스를 공유하여 자신이 개발한 기능의 적용 범위와 환경에 최대한 연동될 수 있도록 제공되어야 한다.

라. 반복 업무의 자동화

- DevOps의 주요 개발을 돕는 도구들이 제공되고 있다.특히 운영조직에서는 별도의 업무가 아닌 모니터링이 용이 할 수 있도록 편리한 도구를 제공하여 신규 개발 프로젝트의 적용의 어려움 및 복잡성을 단순화 시킬 수 있다.
- 테스트 및 배포에 대하여 자동화를 통하여 적용함으로써 배포 시간의 단축 및 서비스 다운타임의 최소화가 될수 있다.

마. 사용자의 참여

- 고객의 요구 사항에 대한 지속적인 참여를 유도하여 실질적으로 기능이 구현되는지 확인이 필요하며 기능간의 우선순위와 운영 조직의 가용성 보장에 대한 Trade Off 직접 할수 있도록 사용자의 적극적인 참여를 유도한다.

4. 성공적인 DevOps를 위한 오픈소스 도구들

-성공적인 DevOps를 위한 자동화 도구를 통해 효율성을 극대화 한다.

가. 이슈관리 도구

- 가정 범용적이며 이슈뿐 아니라 요구사항까지 각 Task를 정리하여 각 요구 업무별 ID를 부여하여 통합적으로 관리한다.

나. 통합IDE

- 이클립스와 같이 Plugin을 풍부히 제공되는 도구를 이용하여 표준 개발환경을 배포한다. 배포를 위한 구조를 설계하여 기존 서비스에 최소한 영향을 주도록 제공한다.

다. 형상관리 도구

- 최근 SVN, GIT등을 통해 소스를 버전 별로 관리하여 배포 시점에 완성된 소스를 점검하고 이상 없으면 배포 처리한다.

라. 배포관리 도구

- 보통 개발서버와 운영서버에는 FTP로 올리는 경우가 많다. 이를 통합 CI도구인 젠킨스나 허드슨을 사용한다면 지정된 시간에 DailyBuild를 수행할 수 있다.
- 배포 시 원복이 가능하도록 백업되어 실시간으로 배포 실패 시 원복 할 수 있도록 제공한다.
- 특정 시점에 배포된 대상을 시계열 또는 순차적으로 저장하여 해당 버전으로 변경 및 검토 필요 시 제공 또는 그 시점으로 변경이 가능하다.

마. 라이브러리 관리도구

- JAVA및 개발 언어에는 라이브러리가 존재한다. 솔루션 또는 특정기능, JAVA의 추가 기능들이 라이브러리로 제공되며 이를 스크립트 기반으로 Nexus등록하고 활용이 가능다.

- Maven을 이용하여 라이브러리를 명세화하고 임시적으로 다운로드 및 처리할 수 있다.

- 이러한 도구와 패러다임의 변화를 통해 DevOps를 성공적으로 도입할 수 있다. 그렇다면 실제적인 업무에 대하여 어떻게 활용할 수 있는지 다음시간에 제공하겠다.

궁금한 사항은 dhc97@naver.com 으로 메일 주세요 "끝"