RED HAT® JBOSS®
MIDDLEWARE

redhat.

백서

JAVA 애플리케이션 현대화를 위한 플랫폼

클라우드 및 현대적인 에코시스템의 워크로드 설계

72%의 IT 비용이 유지 관리 작업에 사용됩니다.[2]

67%의 CEO들이 디지털 혁신을 비즈니스 전략의 핵심으로 설정할 계획입니다.[1]

59%의 CIO들이 현재 보유한 IT 전문성을 우려하고 있습니다.[20]

40%의 조직들이 향후 3년 이내에 IT 인프라를 현대화할 계획입니다.[18]

요약

조직은 현재 전략적인 기로에 서 있습니다. IDC 분석[1]에 따르면 기업 CEO의 2/3는 디지털 혁신 이니셔티브를 중심으로 회사의 전략을 집중할 것이라고 합니다. 기존 기능을 더욱 효과적으로 만드는 것은 디지털 혁신의 목표 중 일부에 지나지 않습니다. 더 큰 목표는 새로운 기능을 수행하고 기존의 데이터를 새롭고 효율적인 방식으로 사용할 수 있도록 하는 것입니다.

그럼에도 현재 IT 투자의 72%[2]와 IT 리소스가 집중되는 부분은 기존 시스템의 유지 관리입니다. 이로 인해 현재 수행해야 하는 작업과 미래를 대비하여 준비해야 하는 작업 사이에 불안감이 형성됩니다.

IDC[3]에 따르면 미들티어 애플리케이션은 데이터 통합, 메시징 및 API(Application Programming Interface) 관리를 제공하므로 디지털 혁신 전략의 가장 핵심이 됩니다. 또한, 미들티어 애플리케이션은 기존 엔터프라이즈 애플리케이션과 클라우드 네이티브 분산형 애플리케이션을 모두 처리할 수 있는 애플리케이션 개발 및 관리 플랫폼을 제공합니다.

특히 Java™ EE 기반 애플리케이션 플랫폼은 다음과 같은 방법으로 현재 기술과 클라우드 고유의 애플리케이션에 대한 지원을 제공할 수 있습니다.

- 기존 인력과 도메인 지식을 신기술에 사용
- 레거시 애플리케이션 및 중요한 데이터 보존
- 기존 환경과 병행하여 새로운 애플리케이션 개발
- 새로운 프로세스 및 아키텍처 구현

디지털 혁신은 조직마다 다양한 형태로 나타납니다. 각각의 조직들은 고유의 전략과 목표를 정의하므로, 해당 결정을 돕고 최대의 효과를 획득하기 위해 기존 IT 리소스를 활용할 수 있는 핵심 원칙들이 있습니다.

1 *Gens, Frank. IDC FutureScape: Worldwide IT Industry 2016 Predictions — Leading Digital Transformation to Scale (전 세계 IT 산업 2016년 예측 - 디지털 혁신 확장하기). IDC, 2015년 11월.*

2 *Zetlin, Minda. 'How to Balance Maintenance and IT Innovation (유지 관리와 IT 혁신의 균형을 유지하는 방법).' ComputerWorld, 2013년 10월 21일. 웹.*

3 *Fleming, Maureen. New Middle-Tier Competencies Enabling Digital Transformation (디지털 혁신을 가능하게 하는 새로운 미들티어 역량). Rep. IDC, 2016년 6월. 웹. Red Hat 후원.*

www.facebook.com/redhatkorea
080-708-0880
buy-kr@redhat.com

kr.redhat.com

미래에 대한 고려: 플랫폼, 프로세스, 아키텍처

현재의 소프트웨어 개발 상태에 대한 프레젠테이션에서 IDC 분석가인 Al Hilwa[4]는 우수한 소프트웨어는 소프트웨어 아키텍처, 개발자 프로세스 및 개발자의 기량이 제대로 작용해서 나타난다고 말합니다. IT 및 운영 부서는 이 세 가지 부분에서 중대한 변화를 겪고 있습니다. 클라우드 및 분산 컴퓨팅으로 인해 인프라에 대한 새로운 규모의 경제 원칙이 도입됨에 따라 아키텍처, 프로세스 및 플랫폼이라는 세 가지 기본 사항에서 클라우드 환경을 활용하는 쪽으로 변화하고 있습니다.

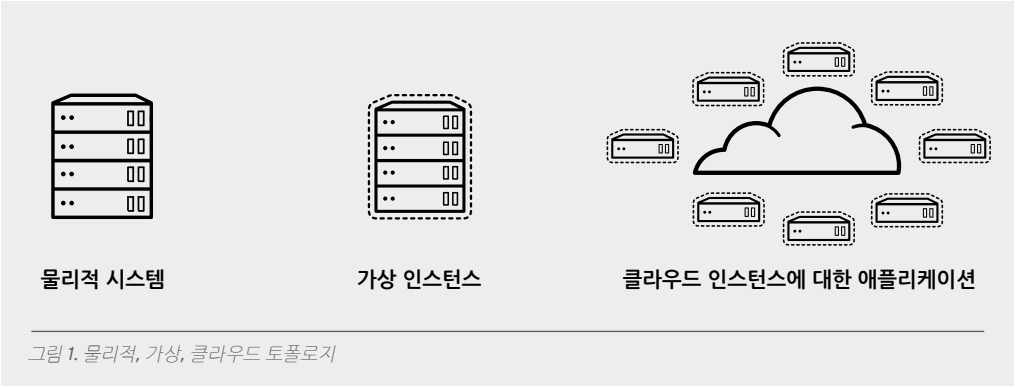

그림 1. 물리적, 가상, 클라우드 토폴로지

플랫폼: 클라우드

클라우드 컴퓨팅은 IT 혁신이라는 새로운 물결의 핵심 인프라가 될 것입니다. 분석가 그룹인 IDC는 2020년까지 총 인프라 지출의 최대 70%가 클라우드 서비스 관련 지출이 될 것으로 예상합니다.[5]

클라우드 기술이 효과를 충분히 발휘하려면 변화하는 환경에 대응해야 합니다. 가상화가 IT의 주요 동인이 되는 이유는 가상화를 통해 물리적 환경에서 운영 환경을 추출하기 때문입니다. 가상화는 운영 체제를 물리적 시스템으로부터 완전히 분리된 것으로 간주하므로, 동일한 하드웨어를 사용하여 복수의 운영 체제 인스턴스를 설치 및 실행할 수 있습니다. 클라우드는 기반이 되는 운영 체제 또는 물리적 환경에서 실행 애플리케이션을 추출하여 환경을 훨씬 더 다양하게 세분화합니다.

클라우드 기반 서비스는 물리적 시스템에 고정적으로 할당되지 않고 분산된 노드 그룹 전반에 걸쳐 존재합니다. 특정 서비스의 요구에 따라 새로운 인스턴스가 생성되거나 폐기되므로 이중화(노드가 실패하면 다른 노드가 이어서 작업 수행) 및 확장성을 제공합니다.

가벼운 인프라 덕분에 고도로 분산된 아키텍처 패턴(예: 마이크로서비스)을 지원할 수 있습니다. 또한, 클라우드 환경은 리소스 활용을 최대화함으로써 운영 효율성 및 비용 절감 효과를 가져옵니다.

4 *Hilwa, Al. 'The New Developer Landscape — Understanding the Modern Software Developer (새로운 개발자 전망 - 현대 소프트웨어 개발자 이해하기).' 2016년 3월. IDC 행사 프레젠테이션.*

5 *Ibid., Gens.*

프로세스: DevOps 및 민첩성

민첩성, CI/CD(지속적 통합/지속적 딜리버리) 및 DevOps는 서로 관련되어 있는 개념으로, IT 팀 구성과 기능을 수행하는데 필수적인 요소입니다.

민첩성은 프로젝트의 계획 및 실행과 관련된 개념입니다. Agile Manifesto(민첩성 선언)[6]에 정의된 민첩성 접근 방식에는 다음 네 가지 핵심 원칙이 있습니다.

- 인력/팀 우선 배치
- 외부 그룹과의 협업
- 변화하는 상황에 대응
- 이해하기 쉬운 소프트웨어 생성

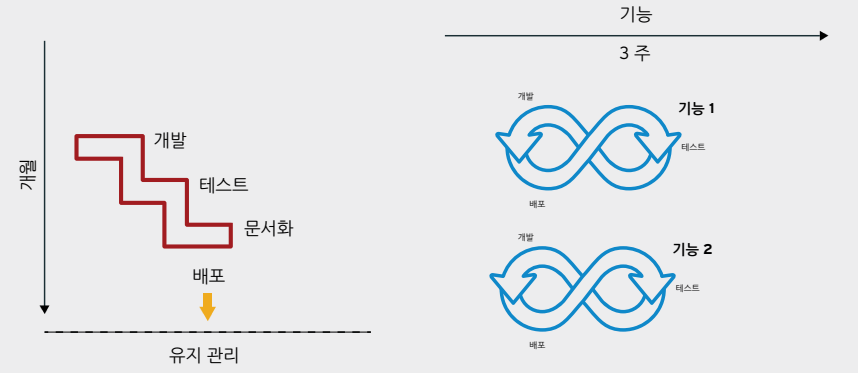

그림 2. 폭포수(Waterfall) 대 민첩한(Agile) 개발 주기

민첩한(Agile) 프로세스는 프로젝트의 기능을 관리 가능한 작업으로 나누고 그러한 작업을 빠르게 반복하여 기능 단위로 작동하는 소프트웨어를 제공하는 것을 목표로 합니다. 이 방법은 폭포수 방식 프로젝트의 제한된 순차적 접근 방식과 뚜렷한 차이를 보입니다. 최근의 Gartner 평가[7]에 따르면 민첩한 접근 방식이 인기를 얻고 있는 반면 폭포수 방식 프로젝트는 50% 미만으로 감소했습니다.

민첩한 접근 방식이 프로젝트 개발에 도입한 한 가지 변화는 개발이 완료된 후가 아니라, 전체 계획 및 개발 프로세스에 다른 그룹을 개입시킨다는 점입니다. 스프린트(Sprint) 기간이 끝날 무렵에 소프트웨어가 작동해야 하며 이는 개발 단계에서 소프트웨어 테스트를 마쳐야 한다는 의미입니다. 민첩한 팀은 개발 주기 안에 테스트를 포함시킵니다. 지속적 테스트 개념은 지속적 통합이 되었고, 빠른 출시는 지속적 배포가 되었습니다. 이제 두 개념은 정의되고 구분된 단계가 아니라 지속적인 프로세스가 된 것입니다.

6 *Beck, Kent, et al. 2001. 'Manifesto for Agile Software Development (민첩한 소프트웨어 개발 선언).' Agile Alliance.*

7 *Wilson, Nathan. Modernizing Application Development Primer for 2016 (2016년 애플리케이션 개발 현대화 지침서). Gartner, 2016년 1월 14일. 웹.*

애플리케이션 라이프사이클은 제품 출시후에도 종료되지 않습니다. 애플리케이션을 배포 및 유지 관리해야 합니다. 이러한 작업은 보통 개발보다는 운영 및 IT 부서로 할당됩니다. 팀이 너무 분리되어 있는 경우 개발자가 운영 환경을 제대로 이해하지 못하거나 운영에서 제품의 전략적 목적을 이해하지 못할 수 있습니다. 이러한 이해의 차이로 인해 소프트웨어 또는 인프라의 성능이 기대 수준 이하의 성과를 보일 수 있습니다. 개발 및 운영 팀이 통합된 방식이 바로 DevOps입니다.

아키텍처: 마이크로서비스

지금까지 기업용 소프트웨어는 규모가 매우 컸습니다. 하나의 애플리케이션이 전체 기업의 최대 작업량(Peak Loads)를 처리해야 했으며, 이 애플리케이션이 요구하는 기능은 해당 애플리케이션에서 수행되어야 했습니다. 서비스는 하나의 모놀리식 애플리케이션의 기능일 뿐이었습니다. 모놀리식 애플리케이션의 좋은 예는 데이터베이스이며, 여기에서 하나의 데이터베이스는 전체 조직을 지원할 수 있습니다.

엔터프라이즈 컴퓨팅 요구가 복잡해지면서 모놀리식 애플리케이션의 유지 관리 요구 사항이 증가했습니다. 모놀리식 인프라가 조금만 변경되어도 상황이 어려워질 수 있는데, 이는 일부가 변경되면 전체가 다 변경되어야 하기 때문입니다. 이로 인해 SOA(서비스 지향 아키텍처)라는 새로운 아키텍처가 탄생하게 되었습니다. 서비스 지향 아키텍처의 경우, 모든 요소를 처리하는 단일 애플리케이션을 사용하기 보다는, ESB(엔터프라이즈 서비스 버스)와 같은 통합 패턴을 사용하여 서로 다른 애플리케이션이 유연하게 연결된 상태로 일부 기능을 제공할 수 있습니다.

하지만 SOA는 전체 환경에 추가적인 시스템 복잡성을 가져왔습니다. 아키텍처의 특정 측면이 용이해지는 반면(새로운 구성 요소 도입 또는 업그레이드 수행 등), 구성 요소 상호 작용을 명확히 이해하지 못하면 SOA 또한 환경 전반에서 연속적인 변경을 수행해야 하는 위험에 처했습니다.

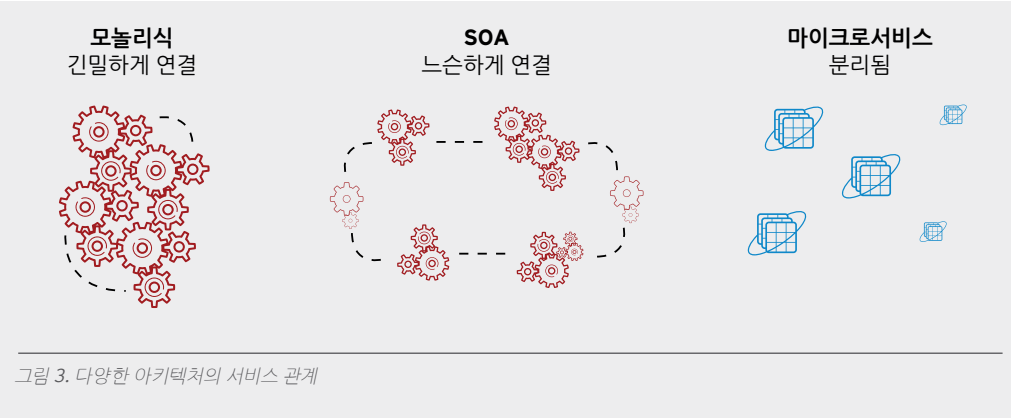

그림 3. 다양한 아키텍처의 서비스 관계

SOA가 올바른 방향으로 나아가고 있었지만, 그 사이 마이크로서비스 아키텍처라는 더 성숙한 모델이 발달했습니다. 마이크로서비스 아키텍처는 서비스가 전문화되고 유연하게 연결된다는 점은 SOA 패턴과 유사하지만 마이크로서비스가 훨씬 더 세분화되어 있습니다. 마이크로서비스 아키텍처는 다음과 같이 명확한 방식으로 서비스를 정의합니다.

- 하나의 명확한 목적
- 잘 정의된 매개 변수(parameter)
- 다중 언어 지원(Polyglot) 구현

1위의 위치 고수

"Java의 미래는 대단히 밝습니다. 언어와 플랫폼은 하룻밤 사이에 바뀌지 않습니다. 많은 우수한 기술들이 오픈소스 방식을 채택했지만, 지금까지 Java 기술을 대체할 만한 기술은 없었습니다. 마이크로서비스와 관련된 작업을 살펴보면 Java 기술을 기반으로 하는 작업이 많이 있습니다. 이러한 기술은 오픈 하이브리드 클라우드 컴퓨팅을 사용해 실제 엔터프라이즈 Java를 다시 정의하고 현대적 패러다임에 맞게 재구성하는 수준에 이르고 있습니다. 지금 진행되는 우수한 작업들은 모두 오픈소스 방식으로 이루어지고 있습니다."

RICH SHARPLES,
미들웨어 제품 관리 부문 선임 이사,
RED HAT

아키텍처 내 서비스는 REST(Representational State Transfer) API와 같은 공통된 메시징 프레임워크를 사용하여 어려운 데이터 변환 트랜잭션이나 추가 통합 계층 없이 서로 통신합니다.

이러한 메시징 프레임워크를 사용하면 새로운 기능 및 업데이트를 보다 빠르게 제공할 수 있을 뿐 아니라, 그러한 작업을 촉진할 수 있습니다. 각 서비스는 개별적으로 제공되므로 한 서비스를 대체, 개선 또는 중지하더라도 아키텍처 내의 다른 서비스에는 영향을 미치지 않습니다. 이러한 가벼운 아키텍처는 분산된 리소스나 클라우드 리소스의 최적화에 도움이 되고 개별 서비스의 역동적 확장성을 지원합니다.

콘웨이의 법칙과 미래

클라우드, DevOps 및 마이크로서비스는 분산된 복잡성이라는 공통적 특성을 공유합니다. 1967년 소프트웨어 개발자인 Melvin Conway[8]는 소프트웨어가 그 개발에 관여한 팀의 커뮤니케이션 구조와 유사하게 설계된다는 사실을 발견했습니다. 커뮤니케이션이 경직되고 불분명하며 불완전할수록 소프트웨어의 실행 성능은 떨어졌습니다.

조직이 디지털 트랜스포메이션 과정의 어디에 위치하든, 최종 결과물은 IT 팀의 문화를 반영합니다. 따라서 기술 논의에 앞서 다음과 같은 사항이 문화적[9]으로 고려되어야 합니다.

- 팀 전체에 걸쳐 이해력이 높은 커뮤니케이션 패턴 수립
- 그룹 간의 장벽 최소화
- 유연한 인프라 및 조직 체계의 수립 촉진

마이크로서비스는 분산된 복잡한 아키텍처와 마찬가지로 견고한 토대를 기반으로 합니다.

마이크로서비스는 기술적으로 클라우드 기반 인프라가 없으면 구현할 수 없습니다.[10] 서로 협조하고 커뮤니케이션이 활발한 여러 기능을 수행하는 팀을 기반으로 구축된 견고한 DevOps 또는 민첩한 수행 환경이 없다면, 조직에서 마이크로서비스 아키텍처는 실패할 것입니다.

JAVA EE의 과거 및 미래

미래에 클라우드 기반이 제공하는 생산성과 함께 복잡성이 동시에 존재한다면 현재의 IT 부서를 어떻게 그러한 미래로 나아가게 할 수 있습니까?

Java는 1995년에 처음 소개된 이래로 세계적으로 가장 인기 있는 프로그래밍 언어로 성장했습니다.[11] Oracle은 전 세계 Java 개발자의 총 인구가 900만 명이 넘는 것으로 추정합니다.[12] 이 수치는 전 세계 전문 개발자의 수가 1100만 명 정도라고 할 때 거의 82%에 해당합니다.[13]

8 *Conway, Melvin E. 'How do Committees Invent (조직은 어떻게 발명하는가)?' Datamation, 14 (5): 28-31. 1968년 4월.*

9 *DevNation 오후 일반 세션. Rachel Laycock. California, San Francisco. 2016년 6월 27일. Performance. https://youtu.be/EC2rk9Jh5Ps*

10 *Gens, Frank. IDC FutureScape: Worldwide IT Industry 2016 Predictions — Leading Digital Transformation to Scale (전 세계 IT 산업 2016년 예측 - 디지털 혁신 확장하기). IDC, 2015년 11월.*

11 *'TIOBE Index for August 2016 (2016년 8월 TIOBE지수).' 2016년 8월. 웹.*

12 *Beneke, Timothy 및 Tori Wieldt. 'JavaOne 2013 Review: Java Takes on the Internet of Things (Java가 사물 인터넷을 이끌다).'*

13 *Hilwa, Al. 'The New Developer Landscape — Understanding the Modern Software Developer (새로운 개발자 전망 — 최신 소프트웨어 개발자 이해하기).' 2016년 3월. IDC 행사 프레젠테이션.*

Sun(나중에 Oracle로 통합)은 애플리케이션의 일반적인 작업을 위한 표준 세트, API 세트 및 런타임 환경을 정의하는 Java 프로그래밍 언어를 기반으로 구축되었습니다. 이 모두를 합하여 Java EE(Java Enterprise Edition)라고 합니다. Java EE 사양을 구현하는 서버를 Java 애플리케이션 플랫폼이라 하고, 이러한 애플리케이션 플랫폼은 여러 IT 개발 환경의 핵심이 되었습니다. Java의 서버-클라이언트 프로그래밍 모델은 초기 인터넷 프레임워크 및 이후 전사적 애플리케이션에 잘 적응할 수 있었습니다.

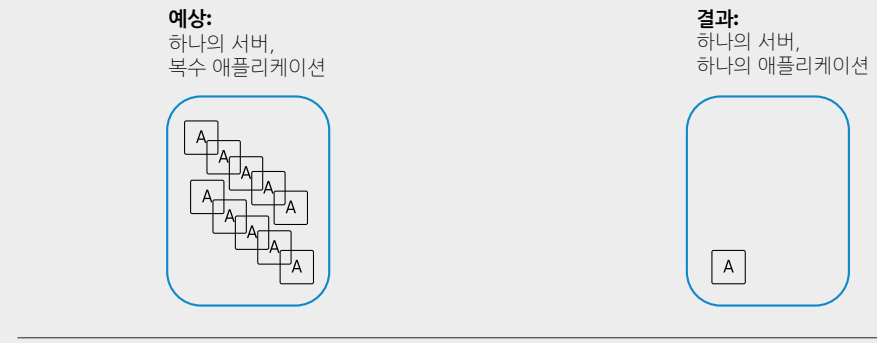

그림 4. 단순화된 애플리케이션/서버 토폴로지

2000년대 초, 애플리케이션 플랫폼을 개발할 때는 대부분의 IT 아키텍처가 모놀리식 애플리케이션을 기반으로 하였습니다. Java EE 애플리케이션 플랫폼은 여러 개의 Java 애플리케이션을 중앙의 단일 위치에서 호스팅해야 했습니다. 실제 애플리케이션에서는 트래픽 분포, 네트워크 대역폭과 대기 시간, 이중화 및 조직의 구분으로 인해 애플리케이션 플랫폼은 종종 한 애플리케이션만을 호스팅했습니다. Gartner는 이러한 Java 애플리케이션 플랫폼을 '슈퍼 플랫폼'이라고 했는데[14] 여기에서는 IT 리소스가 낭비되는 경향이 있었습니다.

Java가 언어 및 플랫폼으로서 복원력이 우수한 이유 중의 하나는 그 적응성에 있습니다. Java 및 Java EE는 Java Community Process라는 커뮤니티를 통해 정의됩니다. 이 Java 커뮤니티가 Java EE 개발 및 관련 커뮤니티 프로젝트(MicroProfile, Wildfly Swarm, Node.js 등)에 활발히 참여합니다.

이 커뮤니티는 상호 연결된 가벼운 애플리케이션의 클라우드 네이티브 개발 및 배포를 허용하는 방향으로 Java 및 Java 플랫폼을 이끌고 있습니다. Java EE 6에서는 프로필 개념을 도입했으며, 여기에는 기존 전체 규모 웹 서버와 대등한 전체 프로필과 보다 경량화된 애플리케이션을 위한 웹 프로필이 포함되어 있습니다. Java EE 7의 경우 프로필 및 모듈 방식이 확장되었으며 몇몇 커뮤니티 프로젝트가 더욱 깊이 있게 추진되었습니다.

- MicroProfile은 웹 프로필에 비해 규모가 작고 마이크로서비스에 특화되어, 트랜잭션 및 메시징 같은 기능에 초점을 맞춘 새로운 Java 프로필의 사양을 개발 중입니다.
- Wildfly Swarm은 요구되는 모든 라이브러리(요구되는 라이브러리에 한함) 및 디펜던시(Dependencies)를 단일 팻 저장소(Single Fat Jar)인 독립형 Java 아카이브로 묶는 컨테이너 이미지로 처리합니다.

대부분의 기술과 마찬가지로 Java는 클라우드 인프라에 적응하고 있습니다. 중요한 장점은 핵심 언어가 이미 이해도가 높고 수백만 명의 개발자가 사용하는 기술이라는 것입니다. 애플리케이션의 아키텍처 및 구현 방식은 새롭지만 그것을 만드는 데 필요한 기술은 이미 확립되어 있습니다.

14 Wilson, Nathan. Modernizing Application Development Primer for 2016 (2016년 애플리케이션 개발 현대화 지침서). Gartner, 2016년 1월 14일. 웹.

현대화를 위해 어떤 경로를 선택할 것인가?

- 클라우드로 전환
- 클라우드 또는 데이터 센터 환경에서 모두 동일하게 작동하는 애플리케이션 플랫폼 사용
- IT 전략을 추진하는 비즈니스 전략 사용
- 현대화해야 하는 인프라의 측면 및 현대화 방법 결정
- 팀 그리고 문화 구축
- 개발자에게 적절한 툴 제공
- 통합, 확장성 및 상호 운용성을 지원하는 IT 에코시스템 설계

리소스에 대한 변경

IDC는 클라우드를 3차 플랫폼 IT 환경의 핵심으로 칭합니다.[15] IDC는 컴퓨팅 기술의 발전에 있어 다음 세 가지로 정의했습니다.[16]

1. 메인프레임 및 개인용 컴퓨터
2. 인터넷 기반 트랜잭션 및 클라이언트/서버 아키텍처
3. 클라우드로 호스팅되는 앱 중심 기술(모바일, 소셜, 사물 인터넷, 빅데이터 등)

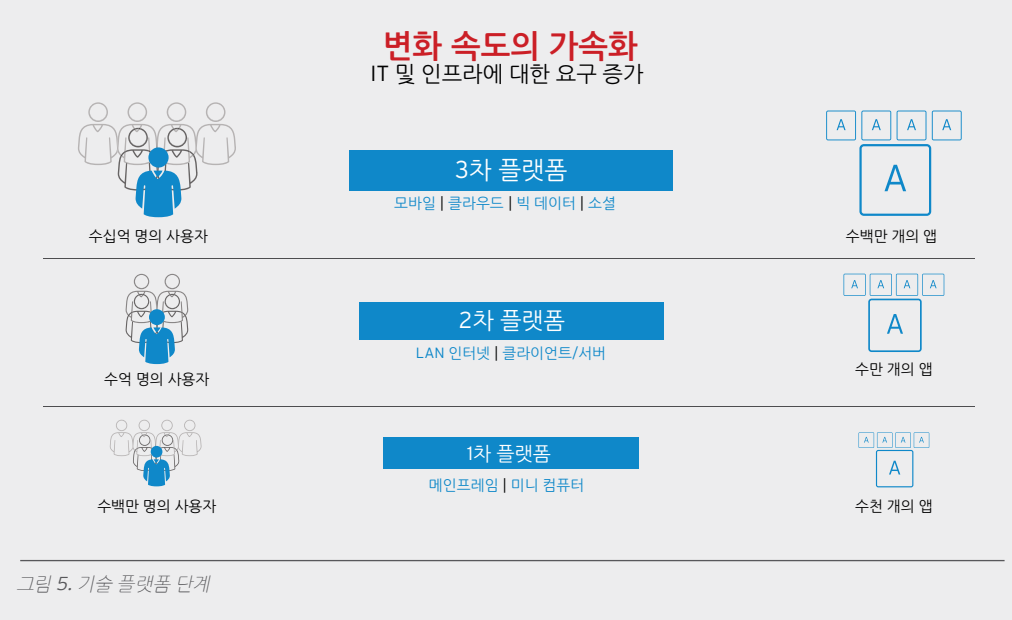

그림 5. 기술 플랫폼 단계

3차 플랫폼은 다른 두 기술에 크게 의존하지만 단순히 동일한 작업을 보다 효율적으로 수행하는 것이 문제가 아니라, 그러한 계층을 기반으로 새로운 작업을 수행하는 것과 관련이 있습니다.

1. 클라우드로 전환(Java EE 사용)

클라우드 컴퓨팅은 최신 IT 환경의 핵심 기술입니다. 이는 클라우드 컴퓨팅의 확장성, 특히 기존의 물리적 또는 가상 시스템에서는 가능하지 않은 방식으로 요구에 따라 역동적으로 노드를 추가하고 연계시키는 능력 때문입니다.

Java EE와 같은 기술 플랫폼에서는 플랫폼이 다음과 같은 여러 환경 유형 범위로 확장 가능해야 합니다.

- 온 프레미스
- 퍼블릭 클라우드(Amazon Web Services, Microsoft Azure, Google 등)
- 프라이빗 클라우드(OpenStack® 프라이빗 클라우드 등)

15 *Gens, Frank. IDC FutureScape: Worldwide IT Industry 2016 Predictions — Leading Digital Transformation to Scale (전 세계 IT 산업 2016년 예측 — 디지털 혁신 확장하기). IDC, 2015년 11월.*

16 *IDC. IDC는 모든 산업에서 광범위한 디지털 혁신과 3차 플랫폼의 거대한 확장이 이루어지는 중요한 시기에 'DX 경제'가 대두할 것으로 예측하고 있습니다. 2015년 11월 4일. 웹.*

- 컨테이너
- 호스팅 서비스

애플리케이션은 모든 환경에서 동일한 방식으로 실행 가능해야 합니다. 일부 애플리케이션 플랫폼은 모든 필요한 환경에서 작동하지 않거나 그러한 환경 간에서 동일한 기능을 실행하지 못할 수 있습니다. 올바른 Java EE 플랫폼을 선택하면 혼합된 환경에서 중요한 상호 운용성이 보장됩니다.

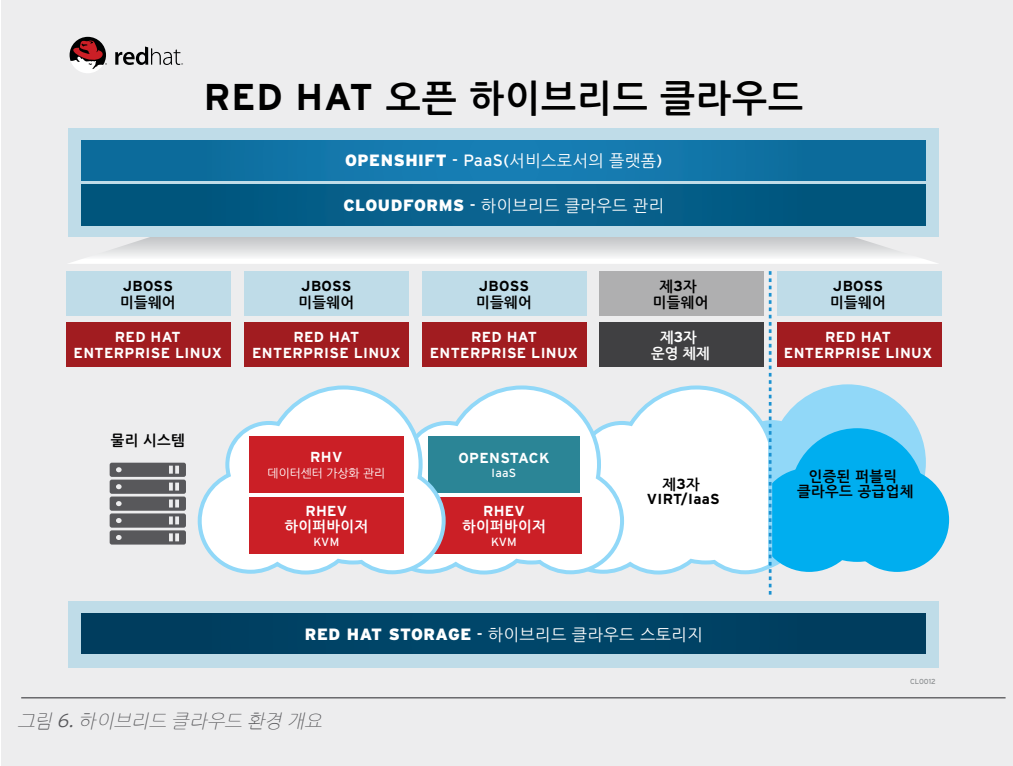

그림 6. 하이브리드 클라우드 환경 개요

하이브리드 환경은 운영하기에 복잡하다는 점 외에도 여러 환경을 운영하기 위한 경제적인 비용이 발생합니다. Gartner에서는 소프트웨어 라이선스를 IT 부서에서 줄일 수 있는 비용임을 파악했으며,[17] 이를 비용 절감과 관련하여 가장 간과하기 쉬운 부분이라고 했습니다. 소프트웨어 가격은 온프레미스, 가상 또는 클라우드에 대한 별도의 라이센스가 있고 다양한 지원 유형에 대해 서브스크립션 서비스가 추가되는 복잡한 구조로 되어 있습니다.

17 McGittigan, Jim 및 Sanil Solanki. The Gartner Top 10 Recommended IT Cost Optimization Ideas (Gartner의 IT 비용 최적화 권장 아이디어 탑 10), 2016년. Tech. no. G00301094. Gartner, 2016년 2월 29일. 웹.

팁: 최종 지향 목표를 정의하십시오.

모든 IT 인프라가 디지털 혁신의 목표를 달성하기 위한 스타트업 스타일의 마이크로서비스 아키텍처로 변화할 필요는 없습니다. 컨테이너 기반 배포를 사용한 클라우드로의 이동 또는 새로운 사물 인터넷 또는 모바일 이니셔티브 도입이 전략적인 계획에 맞는 올바른 단계가 될 수 있습니다.

2. 실제적인 질문의 필요성

귀사의 환경에 적합한 최적의 옵션을 선택하는 방법은 전략 목표와 변화로 인한 위험을 비교한 후 균형을 맞추는 것입니다. 조직마다 요구 사항은 서로 다릅니다. 그러므로 디지털 전략에 비즈니스 전략이 반영되어야 합니다.

- 조직의 목표 또는 전략적 방향은 무엇입니까?
- 이러한 전략적 방향을 실행하는 데 이용할 수 있는 기술이나 리소스를 보유하고 있습니까?
- 해당 애플리케이션을 계속 실행할 수 있는 환경이 있습니까?

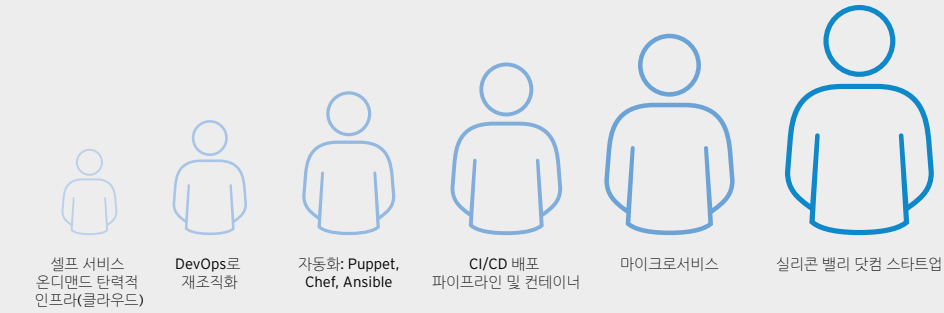

그림 7. 환경 진화의 단계

3. 현대화해야 하는 대상 및 방법 결정

IDC의 Peter Marston은 애플리케이션 현대화에 대한 접근 방식이야말로 디지털 혁신의 핵심 질문이라고 합니다. IDC의 연구에 따르면 조직의 약 40%는 애플리케이션 현대화를 IT 부분 최우선 순위로 이미 설정했거나 향후 3년 이내에 그렇게 할 것이라고 합니다.[18]

다음과 같은 몇 가지 방법으로 현대화를 시도할 수 있습니다.

- 기존 애플리케이션을 현대적인 환경에 맞게 재조정 또는 재설계
- 하나의 환경에서 다른 환경으로 애플리케이션 마이그레이션
- 기존 애플리케이션을 새로운 애플리케이션으로 대체
- 병행(parallel) 환경 생성

조직이 구현할 마이크로서비스 단계에 따라 그에 맞는 방법을 선택해야 합니다.

한 가지 목표는 IT 리소스의 유지 관리 분야에 대한 집중을 감소시키는 것입니다. 새로운 프로젝트와 기존 프로젝트 간에 고르게 IT 비용 및 노력을 나누는 것이 가장 이상적인 방법입니다. 그럼에도 불구하고, IT 투자의 3/4(72%)이 유지 관리 및 운영 작업에 집중되어 있습니다. 다수의 CTO(63%)들은 이 비율이 너무 높다고 생각합니다.[19]

18 Marston, Peter. Ten Criteria to Use for Application Modernization Service Provider Selection (애플리케이션 현대화 서비스 제공업체 선택 시 사용할 10가지 기준). Rep. no. IDC #US41012716. IDC, 2016년 2월. 웹.

19 Zetlin, Minda. '유지 관리와 IT 혁신의 균형을 유지하는 방법.' ComputerWorld, 2013년 10월 21일 웹.

클라우드 기반 Java EE 토대는 유지 관리 비용을 전환하는 데 도움이 됩니다. 모놀리식 및 마이크로서비스 애플리케이션이 혼합된 경우에도 기존 애플리케이션과 새로운 애플리케이션이 모두 동일한 토대 위에서 실행됩니다. 보다 중요한 점은 궁극적으로 여러 환경 간에 Java 애플리케이션을 마이그레이션할 수 있다는 점입니다.

이러한 이식성 덕분에 애플리케이션 현대화를 수행하는 동안 트랜잭션 비용 및 마이그레이션 위험이 감소합니다. 따라서 조직은 인프라를 점진적으로 변경할 수 있습니다. 먼저 클라우드의 비용상 이점을 확보하고, 다음으로 컨테이너의 운영 효율성을 확보할 수 있습니다. 이때 마이크로서비스 또는 고도로 분산된 아키텍처가 현재의 전략에 적합하지 않다면 이를 적용하지 않아도 됩니다.

4. 팀에 집중

CIO가 직면하는 주요 해결 과제 중 하나는 새로운 3차 플랫폼에서 작업을 수행할 수 있는 IT 인재가 부족하다는 인식입니다. IT 인재 위기에 관한 하버드 비즈니스 리뷰 분석에서 CIO 중 59%가 기량 부족으로 인해 IT 또는 전략 과제를 이행하지 못하는 것으로 생각하고 있습니다.[20] IDC의 3차 플랫폼의 중점이 되는 관심 분야는 다음과 같습니다.

- 빅데이터 및 분석(36%)
- 아키텍처 - 엔터프라이즈(27%) 또는 기술(24%)
- 개발(27%)
- 모바일 개발(24%)
- IT 전략(22%)

Forrester Research의 Nigel Fenwick은 IT 조직이 종종 맞춤형 솔루션을 사용해 모든 IT 목표를 완수하려 하지만, 그것이 반드시 최적의 전략적 접근 방식은 아닐 수 있다고 말합니다.
"일반적인 기능을 지원하는 맞춤형 소프트웨어에 수백만 달러를 투입했습니다. 하지만, IT는 더 복잡해졌고 인터페이스는 어려워졌으며 IT 민첩성은 감소하고 비용은 추가되었습니다."[21]

목표를 모두 완수하려고 하기 보다는 다음 절차에 따라 IT 전략을 단순화하고 거기에 집중해야 합니다.

- 조직이 전략적 목표로 제공해야 하는 핵심 기능 2~3개를 파악합니다.
- 다른 모든 요구 사항에 대해 맞춤형 솔루션보다는 표준 기반의 솔루션을 사용합니다.
- 유지 관리가 쉬운 솔루션을 선택합니다.

숙련도에 있어서 차이가 가장 크다고 CIO가 파악한 부분은 기술이 아닌 프로세스 관련 부분이었습니다. Java와 같은 일반적인 기술처럼, 오픈 표준을 사용하면 기술을 배우는 시간을 줄이는 데 도움이 됩니다. Java EE 애플리케이션 플랫폼을 개발 플랫폼으로 구현하면 IT 부서에서 기존 지식과 경험을 활용하여 새로운 분야에 집중하는 프로젝트를 개발할 수 있습니다. Java 경험은 전문 개발 커뮤니티에서 기본적인 능력으로 여겨지므로 프로젝트에 기여할 수 있는 인재의 풀이 넓어집니다.

---

20 I.T. 인재 위기: CIO 및 HR 리더의 검증된 조언. Tech. Harvard Business Review Analytic Services, 2016년 7월. 웹. 후원: Red Hat

21 Zetlin, Minda. 'How to Balance Maintenance and IT Innovation (유지 관리와 IT 혁신의 균형을 유지하는 방법).' ComputerWorld, 2013년 10월 21일 웹.

5. 개발자에게 적합한 툴 제공

기술 플랫폼에 통합된 개발자 중심 툴이 있으면 특별히 테스트 및 자동화 기능이 통합되어 전체 개발 주기가 원활하게 진행되는 데 도움이 됩니다. 개발자에게 직접적인 혜택이 나타나는 핵심 분야는 다음과 같습니다.

- CI/CD용 통합 테스트 모듈
- 배포 자동화
- 개발자 툴킷
- 맞춤형 클래스 로드
- 성능

성능과 관련된 작은 기능도 결과에 큰 영향을 미칠 수 있습니다. 예를 들어 애플리케이션을 배포할 때 시작 속도가 빨라지면 개발자 생산성에 있어 주당 몇 시간을 줄일 수 있는데, 이는 단순히 개발자가 정규 개발 과정에서 하루에도 여러 번 애플리케이션을 다시 시작할 수 있기 때문입니다.

통합 테스트와 같은 다른 기능은 DevOps 또는 CI/CD에 있어 중요합니다. 오스트레일리아의 한 정보 기술 회사는 단순히 테스트 모듈을 배포 프로세스의 일부로 로드함으로써 개발자 생산성을 15% 향상했습니다.[22] 개발, 테스트 및 운영 간 통합은 코드 품질을 개선하고 개발 라이프사이클을 단축하는 데 도움이 됩니다.

6. 에코시스템 구축

IDC 분석가인 Maureen Fleming은 디지털 혁신의 강점은 통합이라고 분석했습니다.[23] 3차 플랫폼은 다음을 포함하여 상호 연동 가능한 여러 측면을 보유하고 있습니다.

- 가상화, 퍼블릭 클라우드 및 프라이빗 클라우드
- 컨테이너 및 오케스트레이션 기능
- 데이터 가상화
- 메모리 캐시 및 스토리지
- 다중 메시징 프로토콜
- 다양한 데이터 소스의 여러 가지 데이터 형식
- 관리 및 배포 툴
- 테스트 자동화
- 비즈니스 프로세스 자동화

디지털 혁신 계획에 적합한 플랫폼을 설계할 때 실제 플랫폼은 해당 계획의 일부에 지나지 않습니다. 해당 플랫폼, 그 위에 구축된 애플리케이션이 작동하게 되는 훨씬 더 큰 에코시스템이 있어야 합니다.

IDC의 Peter Martson은 애플리케이션 현대화 전략의 일부로서 애플리케이션 플랫폼 제공업체의 광범위한 에코시스템을 살펴보기를 권장합니다.[24] 기술 및 숙련도 전반에서 경험과 안내를 제공할 수 있는 솔루션 제공업체만이 아키텍처 계획부터 클라우드 프로비저닝에 이르는 다양한 구현 단계에서 도움이 될 수 있습니다.

---

22 Fleming, Maureen 및 Matthew Marden. Red Hat JBoss Enterprise Application Platform의 비즈니스 가치. Tech. no. #257256. IDC, 2015년 7월. 웹. Red Hat 후원

23 Fleming, Maureen. 통합은 디지털 혁신의 핵심 역량이다. Tech. no. IDC #US41293916. IDC, 2016년 5월. 웹. Red Hat 후원

24 Marston, Peter. 애플리케이션 현대화 서비스 제공업체 선택 시 사용할 10가지 기준. Rep. no. IDC #US41012716. IDC, 2016년 2월. 웹.

결론

디지털 혁신은 IT 인프라 및 데이터를 고객에게 제공 가능한 상품으로 취급하는 전략적 IT 접근 방식입니다. 디지털 혁신을 수행하기 위해서는 내부 비즈니스 기능을 지원하는 내부에 집중된 애플리케이션으로부터 애플리케이션을 제공 가능한 상품으로 다루는 방향으로 관점을 전환하고, 데이터 및 데이터 소스를 이용하는 새로운 방법을 찾아보며, 고객과 협력하는 새로운 방법을 찾아내야 합니다.

디지털 혁신을 통해 앞으로 나아가기 위해서는 다음 세 가지가 필요합니다.

- 조직의 명확한 목표 및 전략
- 기능(functional) 팀 전반에 걸쳐 확실하고 협력적이며 커뮤니케이션이 활발하게 이루어지는 프로세스
- 체계화가 잘 되어 있고 이해하기 쉬운 분산형 아키텍처

Java 애플리케이션은 20년 동안 많은 기업에서 핵심 기술로 사용되었습니다. 이러한 풍부한 데이터, 기능 및 지식은 기업에 있어 매우 중요합니다. Java 기반 애플리케이션을 사용하면 클라우드 네이티브 아키텍처를 사용하는 동시에 기존 애플리케이션을 지원할 수 있는 플랫폼을 기반으로 최신의 개발이 진행할 수 있습니다. 기존의 전사적 모놀리식 배포와 클라우드 기반 배포가 결합을 통해 조직은 기존 지식과 리소스를 활용하는 동시에 새로운 애플리케이션 적극적으로 나아갈 수 있습니다.

클라우드 중심의 애플리케이션 플랫폼을 사용하는 것은 조직에 다음과 같은 몇 가지 이점을 제공합니다.

- IT 부서 내부의 숙련도 또는 기술 차이로 인한 영향을 완화합니다.
- 잠재적 리소스 풀을 확대합니다.
- 기존 워크로드에 대한 브리지 마이그레이션 전략을 제공하는 한편 새로운 환경에서 개발이 가능합니다.
- 통합, 데이터 관리 또는 기타 전략 이니셔티브를 위해 도입되는 다른 기술에 맞는 환경을 제공합니다.

적절한 Java 애플리케이션 플랫폼을 선택하면 기존 엔터프라이즈 애플리케이션을 관리하는 것 이상을 수행할 수 있습니다. Java 애플리케이션 플랫폼은 임원진(Executives)이 계획하는 디지털 트랜스포메이션을 달성하는 동시에, 기존 IT 리소스를 극대화하고 중요한 유지 관리 프로젝트를 이어나갈 수 있는 방법입니다.

한국레드햇 홈페이지 https://www.redhat.com/ko/global/south-korea

Red Hat 소개

Red Hat은 세계적인 오픈소스 솔루션 공급업체로서 커뮤니티 기반의 접근 방식을 통해 신뢰도 높은 고성능 클라우드, Linux, 미들웨어, 스토리지, 가상화 기술을 제공합니다. 또한, 전세계 고객에게 높은 수준의 지원과 교육 및 컨설팅 서비스를 제공하여 권위있는 어워드를 다수 수상한 바 있습니다. Red Hat은 기업, 파트너, 오픈소스 커뮤니티로 구성된 글로벌 네트워크의 허브 역할을 하며 고객들이 IT의 미래를 준비하고 개발할 수 있도록 리소스를 공개하여 혁신적인 기술 발전에 기여하고 있습니다.

아시아 태평양
+65 6490 4200

인도네시아
001 803 440224

뉴질랜드
0800 450 503

베트남
800 862 6691

호주
1 800 733 428

일본
03 5798 8510

필리핀
800 1441 0229

중국
800 810 2100

브루나이 및 캄보디아
800 862 6691

한국
080 708 0880

싱가포르
800 448 1430

홍콩
852 3002 1362

인도
+91 22 3987 8888

말레이시아
1 800 812 678

태국
001 800 441 6039

대만
0800 666 052

www.facebook.com/redhatkorea
080-708-0880
buy-kr@redhat.com

kr.redhat.com
INC0441724_1016