E-Book

Red Hat

애자일 인테그레이션

엔터프라이즈 아키텍처를 위한 기반

www.redhat.com/ko

Red Hat

목차

"IT 현대화는 특히 디지털 트랜스포메이션의 필수 요소이면서도 오해를 불러 일으키는 경우가 많은데, 기업의 혁신과 성과 개선을 촉진하기 때문입니다. 디지털 문화 조성도 필수적인 트랜스포메이션 태스크 중 하나입니다."[1]

Michael Bender 및 Paul Wilmot
Digital McKinsey

혁신을 위한 통합의 역할

비즈니스 성공에 있어 변화에 대응하는 기업의 능력이 점점 더 중요해지고 있습니다. 새로운 혁신 기업이 시장에 진입하고 획기적인 기술의 등장으로 소비자의 기대치가 뒤바뀜에 따라, 조직은 그 어느 때보다 단축된 사이클 변화에 대응하기 위해 진화해야 합니다. 현대적인 소프트웨어 아키텍처와 프로세스를 통해 조직은 변화를 효과적으로 처리하고 시장의 선두주자로 부상할 수 있습니다.

업계 전체는 이미 기술을 통해 혁신을 이루고 있습니다. 오늘날 대부분의 기업은 전자상거래 서비스를 제공하며 소비자는 디지털 환경에서 기업들과 지속적으로 상호작용합니다. 이처럼 혁신적인 변화 트렌드로 인해 조직은 경쟁업체보다 더 빠르게 고객이 원하는 고품질의 새로운 디지털 서비스를 제공할 수 있도록 급격히 IT 환경을 혁신하고 있습니다.

애자일 인테그레이션은 엔터프라이즈 전체에서 애플리케이션과 서비스를 연결하는 새로운 방식으로서 핵심적인 솔루션입니다. 애자일 인테그레이션은 분산형 통합, 애플리케이션 프로그래밍 인터페이스(API) 및 컨테이너라는 3가지 강력한 아키텍처 기능을 결합하여, 민첩성을 향상하고 새로운 프로세스를 추진하여 경쟁 우위를 점할 수 있도록 합니다.

여행, 숙박 등의 산업이 새로운 비즈니스 방식을 통해 변화하였으며, 이제 새로운 서비스가 제공되어 소비자가 색다른 방법으로 서비스와 상호 작용하게 되었습니다. 이러한 혁신적인 변화의 동향은 신기술의 등장과 비즈니스-고객 간 상호 작용 방식에 대한 새로운 사고방식의 출현에 힘입어 금융 서비스 기업부터 공공기관에 이르기까지 주요 산업 전반으로 확산되고 있습니다. 또한 이러한 변화는 기존 조직에게 자사의 IT 기술을 근본적으로 혁신하여 새로운 서비스를 제공해야 한다는 압박감을 기존 조직에 안겨주고 있습니다.

디지털 비즈니스의 필수 요소, 통합

뛰어난 고객 경험 제공은 더 이상 차별화가 아닌 생존을 위해 필수적인 요소가 되었습니다. 개별 경험과 상호작용이 고객 충성도의 기본 구성 요소이긴 하지만 전체적인 고객 여정은 이러한 요소들을 결합합니다.

예를 들어, 호텔에서 숙박은 예약 과정, 투숙객의 Wi-Fi 문제 상담, 회원 우대 프로그램 등과 마찬가지로 중요합니다. 고객은 디지털 경제에서 더 많은 것을 요구하고 있으며 개별화된 컨텍스트별 상호작용에 대한 기대는 그 어느 때보다도 높아졌습니다. 결과적으로 고객 관계는 고객 여정의 어느 한 부분에 따라 성패가 좌우될 수 있습니다.

오늘날 고객 기대를 충족하기 위해 조직은 고객 여정 전 과정에서 고객이 상호작용하는 다양한 애플리케이션을 통합함으로써 원활한 애플리케이션 간 데이터 공유를 지원해야 합니다. 성공적인 통합 전략을 사용하면 다차원적인 고객 인사이트를 생성하고 고객의 요구 사항을 예측하며 고객 이탈을 최소화할 수 있습니다.

이에 가장 적합한 예로 Uber를 꼽을 수 있습니다. 많은 이들이 Uber를 디지털 변혁의 예로 제시하지만, 통합이 Uber의 성공에 미친 영향을 간과하고 있습니다. 택시 회사들은 20년 넘게 동일한 고객 데이터를 이용해왔으나, 기존에는 애플리케이션을 연결하고 데이터를 활용할 수 없었기 때문에 예측 가능성을 제시하거나 고객 기대에 영향을 주지 못했습니다. 애자일 인테그레이션은 이러한 격차를 해소하여 오늘날 디지털 환경의 혁신 기회를 거의 무제한으로 창출하고 기업의 경쟁 방식을 대대적으로 변화시키고 있습니다.

1 *Michael Bender 및 Paul Wilmott, Digital McKinsey, "디지털 혁신: 기술 혁신의 잠재력 활용 방법(Digital reinvention: Unlocking the 'how')". 2018년 1월,* https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Digital%20Reinvention%20Unlocking%20the%20how/Digital-Reinvention_Unlocking-the-how.ashx

"새롭게 실험하는 것을 주저하다가 시장에서 경쟁업체에 밀리고, 민첩성을 실현하지 못하게 되면 그 기업은 침몰하게 됩니다. 기능을 도입하는 건 도박과 같습니다. 운이 좋다면 10%는 원하던 이점을 얻게 되죠. 즉, 새로운 기능을 시장에 더 빨리 출시하여 테스트할 수 있으면 더 큰 성과를 올릴 수 있습니다. 부수적으로는 투자를 빠르게 회수하여 자본 활용도를 높일 수 있으므로 수익 창출 속도도 빨라집니다."[2]

Gene Kim
The Phoenix Project

오늘날 대부분의 혁신적인 시장 선도기업들은 모든 것을 연결하고 있습니다. 통합이 디지털 혁신은 물론, 더 나아가 기업의 성공에 핵심 요소라는 것을 인지하고 있기 때문입니다.

애자일 인테그레이션: 유연성과 속도 확보

디지털 환경에서 속도는 매우 중요합니다. 최신 동향에 발맞춰 나가기 위해서 조직은 계획을 수립하여 소프트웨어 시스템을 신속히 변경해야 합니다. 신속한 가격 변경 또는 신제품 출시가 가능한 기업은 신제품 출시까지 수동적인 검증 단계를 연속적으로 거쳐 약 3개월이라는 긴 시간을 소요하는 기업과 비교하면 의심할 바 없이 큰 시장 경쟁력을 가집니다.

디지털 경제에서 요구되는 속도에 맞춰 소프트웨어를 제공하려면 조직에 애자일 인프라 기반이 필요합니다. 이 경우 애자일이란 애자일 소프트웨어 개발을 의미하지 않으며 보다 전통적인 의미, 즉 유연성과 적응력 및 신속한 대응 능력을 말합니다.[3]

지금까지 애자일 방법론은 소프트웨어 개발에 중점을 두어 애플리케이션 구축 방식을 개선하고 간소화하는 데 주력했습니다. DevOps[4] 프랙티스를 통해 이러한 방법론을 애플리케이션 배포 방식에도 적용하고자 했습니다. 그러나 DevOps는 조직에서 개발한 신규 소프트웨어 애플리케이션을 해결하는 수준에 도달했을 뿐입니다.

여기서 더 나아가 인프라 민첩성은 레거시 소프트웨어를 포함한 모든 IT 시스템을 포괄하는 환경을 구축할 수 있도록 합니다. 애자일 인프라는 기존 시스템, 다양한 데이터 유형, 데이터스트림, 고객 기대치라는 복잡성을 해결하고 이를 통합합니다.

Red Hat은 이 프로세스를 "애자일 인테그레이션"이라고 부릅니다. 통합은 인프라의 일부가 아니라 데이터, 애플리케이션, 하드웨어, 플랫폼이 포함된 인프라라는 개념의 접근 방식입니다. 통합 기술을 애자일 및 DevOps 기술에 맞게 조정함으로써 조직은 개발 팀이 시장 수요에 따라 빠르게 변화하도록 지원하는 플랫폼을 구축할 수 있습니다.

계획의 무의미화: 조직과 민첩성

Red Hat CEO, Jim Whitehurst는 "계획은 더 이상 이전과 같은 의미가 아닙니다. (중략) 잘 알려지지 않은 환경에 대해 세운 계획은 오히려 비효율적입니다."라고 강조합니다.[5] 비즈니스 환경이 가속화되고 변화로 인한 충돌이 발생함에 따라 계획은 쉽게 중단되어 그 효과를 잃게 되며 작업 과정 중 어떤 한 단계에 갇히게 되면 매우 많은 비용이 발생하게 됩니다.

다시 말해, 기업이 보유한 정보가 적거나 환경이 덜 안정적일수록 계획의 가치가 저하됩니다.

변화 불능의 상태

일반적으로 인프라 계획은 수년에 걸친 장기적인 접근 방식을 취합니다. 그러나 다년간의 계획으로 인해 시장 변화에 맞게 적절히 대처하거나 변화에 대한 결정을 조속히 채택하지 못할 수 있습니다. 민첩성이란 보다 빨리 계획을 수립하고 그 계획을 실행할 수 있는 역량을 의미합니다. 이러한 환경에서는 계획의 전 과정이 단축되며 새로운 계획이 지속적으로 수립됩니다.

---

2 *Gene Kim, Kevin Behr, George Spafford, Phoenix 프로젝트: IT, DevOps, 비즈니스 성공에 관한 이야기. 오리건주 포틀랜드: IT Revolution Press, 2013.*

3 *옥스포드 영어 사전*

4 DevOps 이해, https://www.redhat.com/ko/topics/devops

5 Jim Whitehurst, Red Hat Summit 2017 기조연설. https://www.youtube.com/watch?v=8MCbJmZQM9c

팀이 6개월 또는 24개월의 개발 사이클에 익숙해져 있다면 이러한 급격한 변화를 쉽게 받아들이기 어려울 수 있습니다. 전통적인 구조를 유지하고 있는 조직이 새로운 방법으로 시장에 진출하고 있는 스타트업 기업과 경쟁해야 하는 경우 이 문제는 더 심각해집니다. Netflix와 Blockbuster 또는 Uber와 전통적인 택시 서비스 등에서 이러한 현상을 목격할 수 있으며 1980년대의 개인용 컴퓨터나 1993년의 Amazon에서 시작하여 정보화 시대 초기에도 스타트업 기업의 혁신적인 효과가 이미 나타난 적이 있습니다.

표 1. 산업 전반의 혁신적인 기업(Disruptor)

산업	전통적인 서비스	혁신적인 기업	영향
교통/운송	택시, 대중 교통	Uber, Lyft	현지의 소규모 회사에서는 따라하기 힘든 일관된 고객 경험 창출
자산 관리	투자 회사	펀드 자동화	자금 관리에 있어 개인의 역량이 아니라 알고리즘이 차별화 요소가 됨
소매	오프라인 쇼핑	Amazon	쇼핑 방식이 오프라인 쇼핑에서 온라인 구매로 변화
검색 엔진	Google, 브라우저 기반 검색	음성 검색	Google의 주요 시장 채널에 영향을 미치고 새로운 기업이 시장에 진입할 수 있게 됨

스타트업 기업과 혁신적인 기업이 가진 장점은 인프라, 팀, 애플리케이션, 아키텍처 및 배포 프로세스까지 새로운 방식으로 유연하게 구성할 수 있다는 점입니다. 단순히 혁신적인 아이디어를 생각해내는 것을 넘어서 아이디어를 실현할 수 있는 것은 Rachel Laycock이 "레거시 피플(legacy people)"[6]이라고 표현한 것처럼 레거시 인프라에 발이 묶여 있지 않기 때문이며, 따라서 이들 기업은 민첩하게 행동할 수 있습니다.

이들 조직은 새로운 것을 구축할 수 있는 능력이 있을 뿐 아니라 처음부터 변화에 바로 적응하도록 설계된 시스템도 구축할 수 있습니다. 소프트웨어 인프라는 이들이 다른 기업과 차별화되는 요소 중 하나이며, 변화하는 시장의 요구에 따라 시스템의 거의 모든 부분을 교체, 업데이트, 제거할 수 있습니다. 스타트업 기업 일부는 적응력이 아직 제대로 갖춰지지 않아 어려움을 겪기도 하지만 우수한 조직은 어떤 경우에도 기업 스스로 변화하는 능력을 유지합니다.

6 *Rachel Laycock, "지속적 제공(Continuous Delivery)." 오후 세션, Red Hat Summit – DevNation 2016. 2016년 7월 1일, 캘리포니아주 샌프란시스코. https://youtube.com/watch?v=y87SUSOfgTY*

민첩성을 위한 기반 구축

급변하는 환경에서 성공하려면 전체 IT 인프라가 민첩하게 작동해야 합니다.

다음과 같이 두 가지 측면에서 변화가 일어나야 합니다.

1. 아키텍처 설계에서 팀 커뮤니케이션에 이르기까지, 조직적이고 문화적인 차원에서 애자일 프로세스를 지원해야 합니다.
2. 기술 인프라에서 사용자가 기능을 신속하게 업그레이드, 추가, 제거할 수 있도록 지원해야 합니다.

기술 및 문화적 변화가 민첩성을 실현하는 것이 아니라 민첩성을 위한 기반이 됩니다.

eBay의 제품 매니저인 Marty Cagan은 스스로 세금이라 칭하는 개념을 모든 프로젝트에 적용합니다. 즉, 일상적인 모든 프로젝트에서 어느 정도의 시간과 리소스를 확보하여 신규 인프라 프로젝트에 투입하는 것입니다.[7] 이렇게 함으로써 신규 프로젝트와 혁신을 최우선순위로 삼을 수 있습니다.

인프라의 민첩성

여러 그룹이 개선을 위한 방법을 모색하기 위해 서로 다른 방향으로 움직이고 있기 때문에 신기술을 대량으로 도입하는 것이 애자일 인프라를 구축하는 데 도움이 되지 않는 경우가 많습니다. 최상위 수준의 목표를 일관성 있게 수립하지 못하면 조직의 전체 운영에 진정한 변화를 가져오기 위해 어떤 새로운 기능을 도입해야 할지 결정하기 힘들어집니다.

애자일 인테그레이션의 3가지 핵심 요소

세 가지 주요 기술이 애자일 인테그레이션 접근 방식을 뒷받침합니다.

1. **분산형 통합:** 수십 개의 고급 통합 패턴이 엔터프라이즈의 작업과 데이터 흐름을 반영합니다. 이러한 통합 패턴이 컨테이너 내에 배포되어 있으면 특정 애플리케이션 및 팀의 요구에 부합하는 위치와 규모에 맞게 통합 패턴을 배포할 수 있습니다. 이는 전통적인 중앙 집중식 통합 아키텍처가 아닌 분산형 통합 아키텍처이며, 이 아키텍처를 기반으로 개별 팀이 적절한 통합 패턴을 민첩하게 정의하여 배포할 수 있습니다.
2. **API:** 안정적이고 잘 관리된 API는 팀 간의 협업, 개발, 운영에 큰 영향을 미칩니다. API는 안정적이고 재사용 가능한 인터페이스에서 주요 자산을 래핑하여 인터페이스가 조직 전반에서, 파트너와의 관계에서 그리고 고객과의 관계에서 사용 및 재사용될 수 있는 빌딩 블록으로 작동하게 합니다. API는 컨테이너와 함께 다양한 환경에 배포할 수 있으며 이를 통해 여러 사용자가 서로 다른 API 세트와 상호 작용할 수 있습니다.
3. **컨테이너:** API와 분산형 통합 기술 모두에서 컨테이너는 기본적인 배포 플랫폼 역할을 합니다. 컨테이너를 사용하면 특정 환경 내에 적합한 서비스를 배포하여 일관되고 손쉬운 방법으로 개발, 테스트, 유지관리를 수행할 수 있습니다. 컨테이너는 DevOps 환경과 마이크로서비스에서 가장 많이 사용되는 플랫폼이므로 컨테이너를 통합 플랫폼으로 사용하면 개발 팀과 인프라 팀 간에 훨씬 더 투명하고 협력적인 관계를 구축할 수 있습니다.

---

7 *Cagan, Marty, '인스파이어드: 고객에게 사랑받는 제품을 만드는 방법(Inspired: How to Create Products Customers Love)'. Wiley Press, 2017.*

그림 1. 애자일 인테그레이션의 3가지 핵심 요소

애자일 인테그레이션의 3가지 핵심 요소는 각각 고도의 추상화 수준을 구현하여 서로 다른 팀들이 협업할 수 있게 하므로 IT 인프라의 민첩성을 높이는 토대가 됩니다. API 및 분산형 통합 기능과 함께 컨테이너 플랫폼을 사용하면 통합 자체에서 통합 구현을 추상화할 수 있습니다. API 및 분산형 통합 패턴은 기본 인프라를 파악하거나 변경할 필요 없이 광범위하게 이해가 가능한 수준에서 특정 자산을 패키지화하기 때문에 팀의 민첩성을 향상시킬 수 있습니다.

이러한 기술은 각각 특정 통합 과제에 대해 뛰어난 민첩성을 제공하며 함께 사용하면 그 효과는 몇 배가 됩니다. 기술을 뒷받침하는 것은 문화입니다. DevOps 프랙티스, 특히 자동화와 배포 프로세스를 결합하면 기술의 이점이 더 강력해집니다.

요소 1: 분산형 통합

전통적인 IT 시스템이 직면한 최대 과제는 조직 전반을 비롯한 여러 환경에 애플리케이션을 연결해야 한다는 것입니다. 과거에는 더욱 복잡한 중앙 집중식 통합 허브를 통해 이를 해결했습니다. 이러한 허브는 ESB(Enterprise Service Bus)로 구현되어 매우 복잡한 병목 현상을 일으켰으며 신속한 변화에 대비하기 위한 융통성이 결여되어 있습니다.

ESB를 사용하려면 개발 및 운영 환경에서 사용 중인 툴 외에 전체 라이프사이클에서 ESB 툴을 사용해야 합니다. 이러한 제한으로 인해 운영에서 효율성이 떨어지고 여러 가지 문제와 오류가 발생하기 쉬워집니다.

분산형 통합을 사용하면 이전 세대의 ESB가 지향했던 기술적 목표를 조직 내의 여러 팀에게 더 익숙한 방식으로 달성할 수 있습니다. ESB와 마찬가지로 분산형 통합 기술은 트랜스포메이션, 라우팅, 파싱, 오류 처리, 경고 기능을 제공합니다.

"소프트웨어에서 문제가 발생했을 때는 사용 횟수를 줄이는 것이 아니라 늘리는 것이 문제를 완화하는 방법입니다."[8]

David Farley
지속적 제공: 빌드, 테스트, 배포 자동화를 통한 안정적인 소프트웨어 릴리스

차이점은 통합 아키텍처입니다. 분산형 통합 아키텍처는 각 통합 지점을 대규모 중앙 집중식 통합 애플리케이션의 일부가 아닌 별도의 고유한 배포로 처리합니다. 따라서 통합은 조직 전체에 배포된 다른 통합에 영향을 미치지 않고 특정 프로젝트나 팀을 위해 로컬로 컨테이너화되고 배포될 수 있습니다. 여기서 통합은 마이크로서비스로 다뤄지기 때문에[9] 개발 속도와 릴리스 사이클 속도가 향상됩니다.

이러한 분산형 접근 방식을 활용하면 유연성을 최대한 발휘할 수 있습니다. 또한 기본적인 컨테이너 플랫폼에서 애자일 또는 DevOps 팀과 동일한 툴체인을 사용함으로써 팀이 자체 툴과 일정을 통해 통합을 더 효과적으로 관리할 수 있도록 합니다.

개발자 툴과 프로세스에 맞게 조정하는 것이 중요합니다. 분산형 통합은 한 부서에서 여러 전문 사용자가 개발하고 관리하는 중앙 집중식 소프트웨어 인프라가 아니며 소프트웨어 개발 프로세스와는 별개로 배포됩니다. 공통 플랫폼 및 툴링으로 통합 아키텍처를 배포하면 프로젝트 수준에서 모든 개발자가 액세스할 수 있고 통합이 필요할 때마다 경량의 배포를 지원할 수 있습니다.

표 2. 소프트웨어 라이프사이클의 각 단계에 대한 통합 기술 비교

라이프사이클 단계	ESB, 대부분의 iPaaS(Integration Platform as a Service: 서비스로서의 통합 플랫폼)	분산형 통합 기술 지원
버전 관리	독점	GitHub 및 기타
빌드	독점	Maven 및 기타
배포	독점	컨테이너 및 기타 DevOps 툴
관리 및 확장	독점	컨테이너 및 기타 DevOps 툴

분산형 백플레인으로 스트리밍

애자일 인테그레이션은 애플리케이션 요구사항에 따라 동기식 또는 비동기식 통합을 자유롭게 사용할 수 있도록 지원합니다. 일반적인 분산형 통합 접근 방식은 앞서 논의한 바와 같이 동기식 방법을 사용하여 다양한 사용자 간에 데이터를 공유하도록 컨테이너를 배포합니다. 또 다른 분산형 통합 옵션으로는 비동기식 방법인 스트리밍이 있습니다. 이 방식을 통해 애자일 개발자는 필요한 경우 다른 소스에서 이벤트를 수신하고 다시 읽고 재생할 수 있습니다. 메시징은 중간 저장소에서 데이터를 복제하는 데 사용되므로 데이터는 여러 팀과 해당 애플리케이션 간에 공유될 수 있습니다. 중간 데이터 저장소가 있으면 마이크로서비스 팀이 지속적으로 다른 소스에서 동기식으로 데이터를 찾지 않아도 되므로 마이크로서비스 팀에 유용합니다.

8 *David Farley 및 Jez Humble, 지속적 제공: 빌드, 테스트, 배포 자동화를 통한 안정적인 소프트웨어 릴리스. Addison-Wesley Professional, 2010.*

9 *See Martin Fowler의 유용한 마이크로서비스 정의 참조:* https://martinfowler.com/articles/microservices.html

분산형 스트리밍 플랫폼은 실시간으로 기록 스트림을 게시, 구독, 저장 및 처리할 수 있습니다. 이는 여러 소스에서 데이터 스트림을 처리하고 여러 사용자에게 전달하도록 설계되었습니다. 스트리밍 플랫폼은 초당 최대 수백만 개의 데이터 포인트를 처리할 수 있으므로 IT 운영 및 전자상거래와 같이 실시간 데이터 가용성을 요구하는 활용 사례에 특히 도움이 될 수 있습니다. 아래 그림에서는 다양한 스트리밍 활용 사례를 나타내는 패턴을 보여줍니다. 운영 결과에 따라 모든 애플리케이션에는 변경 사항이 발생합니다. 이러한 애플리케이션은 하나 이상의 스트리밍 프로세서에 해당 변경 사항을 이벤트로 전파할 수 있습니다. 그러면 스트리밍 프로세서는 패턴 일치 또는 변환을 수행한 다음, 현재 상황을 대개 분석 시스템에서 제공되는 과거 인사이트와 일치시킵니다.

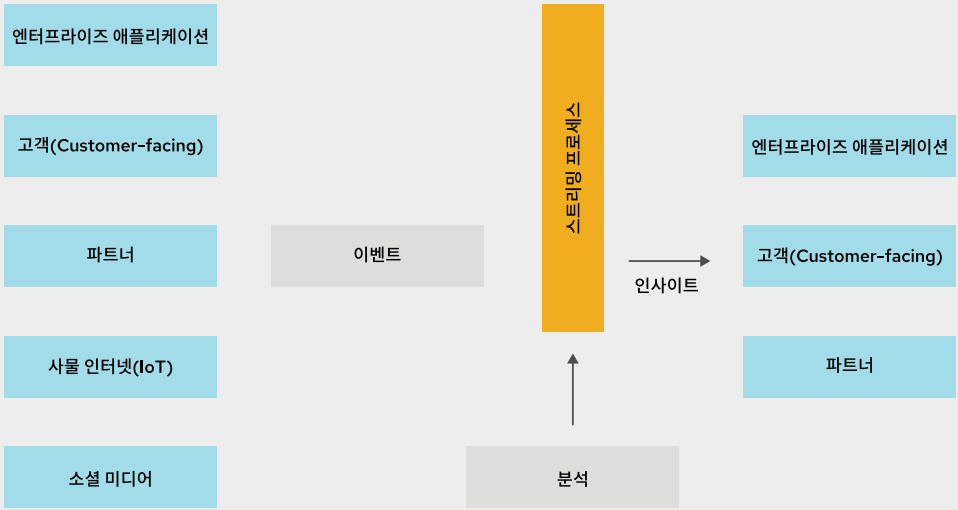

그림 2. 처리 플랫폼을 거쳐 인사이트를 제공하는 이벤트 흐름

비동기식 통합은 일부 활용 사례에서 동기식 통합보다 유용합니다. 동기식 통합의 경우 커뮤니케이션이 성공하려면 수신자가 있어야 하는 반면, 메시징 기반의 비동기식 통합에서는 수신자가 없더라도 메시지를 전송할 수 있습니다. 동기식 커뮤니케이션에서는 애플리케이션이 지속적으로 동작하기 위해 응답을 기다려야 하므로 시간 제한이 일반적입니다. 비동기식 메시징의 경우에는 이러한 지연이 발생하지 않아, 응답이 수신됐는지 여부와 상관없이 처리가 지속됩니다. 스트리밍과 같은 비동기식 통합은 이처럼 유연하므로 대용량 데이터 처리에 적합합니다. 비동기식 통합은 또한 시스템이 타임아웃될 가능성이 낮기 때문에 고가용성과 신뢰성을 제공합니다.

스트리밍과 같이 비동기식 이벤트 기반 통합을 사용하여 동기식 통합과 API 사용을 확대하면 애자일 통합이 한층 강화됩니다. 통합과 메시징을 결합하여 보다 효과적인 라우팅, 다양한 언어 및 프로토콜 지원, 높은 처리량, 뛰어난 데이터 관리를 통해 통합 아키텍처의 전반적인 성능을 향상시킬 수 있습니다.

이벤트 메쉬

새로운 비동기식 분산형 통합의 또 다른 옵션으로 이벤트 메쉬가 있습니다. 이는 분리된 애플리케이션, 기기 및 클라우드 간 이벤트를 분산하기 위한 설정 가능한 동적 인프라 레이어입니다.

이벤트 메쉬는 상호 연결된 이벤트 브로커 네트워크로 이루어져 있으며, 비동기식의 이벤트 기반 상호작용을 처리합니다. 이는 환경에 구애받지 않으므로 하나의 애플리케이션 이벤트는 온프레미스, 프라이빗 클라우드, 퍼블릭 클라우드, 하이브리드 클라우드, 서비스로서의 플랫폼(Platform-as-a-Service, PaaS) 또는 사물 인터넷(Internet of Things, IoT)에 이르기까지 배포 위치에 상관없이 이벤트 라우팅을 설정하지 않고도 다른 모든 애플리케이션에서 라우팅하고 수신할 수 있습니다.

뿐만 아니라 이벤트 메쉬는 마이크로서비스, 클라우드 네이티브 서비스, 레거시 애플리케이션, 기기, 데이터베이스 등 모든 것을 연결할 수 있으므로, 기본적으로 어떤 이벤트 생성자라도 이벤트 소비자에 연결됩니다. 이벤트 메쉬는 이벤트 생성자와 이벤트 소비자 간 최단 경로를 결정하여 실시간 상호작용을 지원하므로 매우 효율적입니다.

IT 환경이 점점 분산되면서 이벤트 메쉬는 현대적인 솔루션을 제공하여 유연하고 신뢰할 수 있으며 빠르고 보다 안전한 이벤트 커뮤니케이션을 지원하고 있습니다.

비동기식 상호작용과 이벤트 기반 아키텍처 패턴은 새로울 것이 없지만, 이벤트 메쉬는 통합 분야에서 새롭게 등장한 획기적인 접근 방식입니다. 이벤트 메쉬가 현재 널리 도입되어 있지는 않지만, 디지털 트랜스포메이션으로 인해 조직들이 이벤트 기반 통합의 유연성을 더 많이 활용하게 되면서 Red Hat은 이 기술이 앞으로 널리 사용될 것으로 전망하고 있습니다.

이벤트 메쉬는 서비스 메쉬와는 다릅니다. 이벤트 메쉬가 비동기식이라면 서비스 메쉬는 API 기반의 동기식 통합을 지원합니다.

분산형 통합은 그 구조상 통합을 마이크로서비스로 다루며 컨테이너화가 가능하고 로컬에 쉽게 배포할 수 있을 뿐 아니라 릴리스 사이클도 매우 짧습니다.

통합 기술은 경량의 마이크로서비스 기반 아키텍처를 지원할 수 있어야 합니다. Red Hat® Fuse를 사용하면 사용자가 통합을 코드로 처리할 수 있으며 컨테이너를 포함한 모든 곳에서 이를 실행할 수 있습니다.

또한 Fuse는 Red Hat AMQ와 번들링되어 강력한 메시징 인프라를 제공함으로써 시스템 간에 이벤트와 데이터가 효과적으로 라우팅되도록 지원합니다. 메시징은 비동기식 특성을 가지고 있어 종속성을 필요로 하지 않기 때문에 마이크로서비스와 연동 시 유용합니다.

Red Hat AMQ는 Fortune 100대 기업의 절반 가량이 신뢰하는 엔터프라이즈 쿠버네티스 플랫폼인 Red Hat OpenShift® Container Platform에서 AMQ 스트림을 통해 분산형 스트리밍 플랫폼인 Apache Kafka를 제공합니다.[10] AMQ 스트림은 Apache Kafka 프로젝트를 기반으로 대규모 확장이 가능한 분산형 고성능 데이터 스트리밍 기능입니다. 이러한 조합을 통해 높은 처리량과 짧은 지연 시간으로 마이크로서비스와 다른 애플리케이션 간 데이터를 공유합니다.

Red Hat은 또한 글로벌 P2P 이벤트 전송 시스템인 AMQ Interconnect를 통해 이벤트 메쉬를 제공합니다. AMQ Interconnect는 간편하고 안정적으로 사용자에게 도달할 수 있도록 병목 현상과 장애를 우회하여 라우팅하는 분산형 트래픽 관리자 역할을 합니다. 이러한 방식으로 노드 및 클라우드 장애에 복구 능력을 제공합니다.

요소 2: 애플리케이션 프로그래밍 인터페이스(Application Programming Interface, API)

대부분의 정보 인프라에는 수백 또는 수천 개의 시스템, 애플리케이션, 자산이 포함되어 있으며, 시스템에서 상호 작용이 이루어지기가 매우 어렵고 어떤 시스템이 사용 가능한지 IT 관리자가 파악하는 것도 불가능할 수 있습니다. API는 통합 기술을 사용하여 연결되는 모든 대상에 대한 인터페이스 역할을 함으로써 이 문제를 해결합니다.

조직이 중앙 집중식 통합 방식에서 분산된 접근 방식으로 전환하면서, 셀프 서비스가 주요 우선순위가 되었습니다. 애자일 팀에는 회사 내외부에서 개발된 서비스를 찾아내고, 테스트하고, 사용할 수 있는 권한과 자율성이 부여되어야 합니다. 강력한 API 기능은 이러한 권한과 자율성을 팀에 제공합니다. API를 사용하면 팀에서는 원하는 수준의 통합을 실현할 수 있으며 조직에서는 보안, 권한 부여, 사용 정책을 관리 및 시행할 수 있습니다.

API는 애플리케이션의 상호 커뮤니케이션 방식을 설정하는 정의 또는 룰 세트를 제공하여, 개발자에게 통합을 위한 공통 언어와 통합 설계 방식에 대한 참조를 제공합니다.

10 *Red Hat 보도 자료, "전 세계 1,000개 이상의 기업에서 Red Hat OpenShift Container Platform을 도입하여 비즈니스 애플리케이션 지원." 2019년 5월 8일,* https://www.redhat.com/ko/about/press-releases/more-1000-enterprises-across-globe-adopt-red-hat-openshift-container-platform-power-business-applications?extIdCarryOver=true&sc_cid=701f2000001OH74AAG

개발자들은 프로젝트 내에서 API를 구성 요소로 사용하지만, 기본적으로 API 공유도 가능하며 다양한 API 또는 서로 다른 API의 서브셋을 여러 대상에게 제공할 수 있습니다. 예를 들어, 한 공급업체의 요구 사항은 내부 개발 팀이나 커뮤니티 개발자의 요구 사항과는 다를 수 있으며, 적합한 API는 필요한 경우 이러한 그룹 각각에 공개될 수 있습니다.

API를 성공적으로 활용하려면 조직은 API 관리 기능을 갖춰야 합니다. API 관리에는 애플리케이션과 사용자 그룹을 위한 API를 설계하는 것뿐 아니라 API 라이프사이클을 관리하는 것도 포함됩니다. API는 점점 더 제품처럼 관리되고 있고 여러 팀이 각 API를 담당하고 있지만 모든 리소스 전반에서 일관성과 편의성을 보장해야 할 필요가 있습니다.

서비스 메쉬

서비스 메쉬는 마이크로서비스에 설정 가능한 인프라 레이어를 제공함으로써 API 통합에서 더 나아가 유연성, 신뢰성, 신속성, 관리 가능성을 갖춘 커뮤니케이션을 지원합니다.

조직이 처음 마이크로서비스를 사용하는 경우 빌트인 거버넌스가 운영 중단을 최소화하여 서비스 간 커뮤니케이션을 처리합니다. 그러나 새로운 애플리케이션, 서비스 및 기능이 마이크로서비스 형태로 지속적으로 추가되면서 어느 순간 아키텍처의 운영 복잡성으로 인해 문제가 발생할 수 있습니다. 모든 새로운 서비스는 새로운 잠재적인 장애 지점을 유발합니다. 서비스 요청이 과도하게 몰리는 경우도 있습니다. 수백 또는 수천 개의 마이크로서비스가 모두 서로 연결을 시도하면 커뮤니케이션 지연과 애플리케이션 다운타임이 발생할 수 있습니다. 게다가 복잡한 마이크로서비스 아키텍처에서 문제의 원인을 찾아내기란 매우 어렵습니다. 서비스 메쉬는 이러한 문제들을 해결하기 위해 도입되었습니다.

서비스 메쉬는 애플리케이션에 바로 구축된 전용 인프라 레이어입니다. 서비스 메쉬는 개별 서비스에서 서비스 투 서비스 커뮤니케이션을 제어하는 로직을 제거하고 이를 인프라 레이어로 추상화합니다. sidecar 프록시는 마이크로서비스와 나란히 위치하며 다른 프록시로 요청을 라우팅합니다. 이러한 sidecar들이 모여 메쉬 네트워크를 형성합니다. 이러한 방식으로 서비스 메쉬는 서비스에서 다음 서비스로 요청을 전송하여 모든 마이크로서비스의 작동 방식을 최적화합니다.

사용자는 마이크로서비스 구성 요소 자체를 변경하지 않고 서비스 메쉬를 사용해 마이크로서비스에 라우팅, 내결함성, 보안, 가시성, 모니터링, 테스트와 같은 추가 기능을 도입할 수 있습니다. 이러한 기능은 마이크로서비스에 정보와 기능을 삽입하여 sidecar 프록시에서 수행됩니다.

서비스 메쉬를 통해 커뮤니케이션 장애를 보다 손쉽게 해결할 수 있는 것은 문제의 원인이 마이크로서비스 내에 숨겨져 있지 않기 때문입니다. 오히려 문제는 서비스와 함께 가시적인 인프라 레이어에 있습니다.

뿐만 아니라 서비스 메쉬는 다운타임에 대한 애플리케이션의 취약성을 완화시키는데, 이는 서비스 메쉬가 장애가 발생한 서비스로부터 요청을 재라우팅할 수 있기 때문입니다.

서비스 메쉬는 또한 성능 메트릭스를 수집하여 서비스 장애 발생 시 더욱 효과적인 진단 기능을 제공함으로써 다운타임을 대폭 단축합니다. 개발자는 또한 이 성능 데이터를 사용하여 향후 릴리스에서 설계를 최적화할 수 있습니다.

서비스 메쉬가 없으면 각 마이크로서비스는 서비스 간 커뮤니케이션을 지원하는 로직으로 코딩해야 하기 때문에 개발자의 업무 부담이 늘어납니다. 서비스 메쉬는 이러한 추가 코딩의 필요성을 없애주어 개발 프로세스를 간소화합니다. 따라서 Red Hat은 향후 몇 년에 걸쳐 이 기술에 대한 수요가 급증할 것으로 전망합니다.

API의 강점은 내부 개발자와 외부 사용자가 모두 사용할 수 있다는 데 있습니다. Red Hat 3scale API Management는 모든 사용자를 위한 툴을 제공합니다. 개발자 포털을 제공하여 API 개발에 대한 협업을 촉진하고 관리 포털을 통해 이러한 API를 게시할 수 있도록 지원합니다. 3scale API Management는 인증을 제공하고 주요 클라우드 제공업체와 통합하며 컨테이너 내에서 실행하여 외부에서 API를 사용할 수 있게 합니다.

API 전략에는 퍼블릭 API를 위한 API 설계가 결합되어 있습니다. 3scale API Management, 특히 컨테이너 플랫폼상에 구축된 3scale은 이러한 전략을 실행하기 위한 수단을 제공합니다.

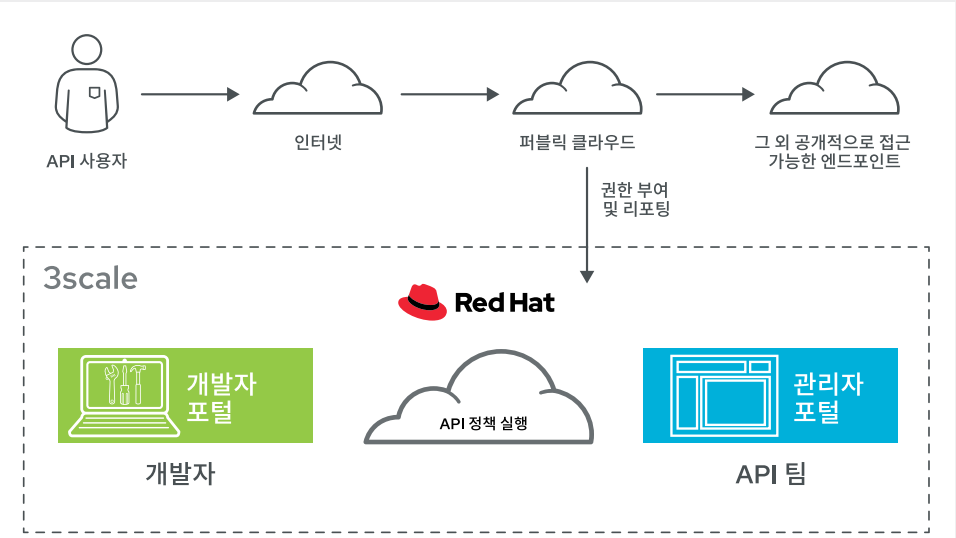

그림 3. API 관리, 엔드포인트, 퍼블릭 클라우드

Red Hat은 또한 Red Hat OpenShift Service Mesh를 제공합니다. 이는 널리 사용되는 Istio 서비스 메쉬를 Jaeger(추적 기능) 및 Kiali(시각화 기능)와 같은 다른 주요 프로젝트와 결합하여, 마이크로서비스 배포에 대해 개선된 관리 효율성 및 추적 기능을 제공합니다.

뿐만 아니라 3scale은 3scale Istio 어댑터를 사용하여 쿠버네티스의 Istio Service Mesh에 전체 API 관리 기능을 제공합니다.

"멀티클라우드를 제대로 사용하려면 프로세스, 직원의 기술 및 조직 구조를 면밀히 고려해야 하지만, 업계에서는 멀티클라우드를 실행 가능하게 해주는 기술 플랫폼은 컨테이너와 쿠버네티스라는 데 동의합니다."[11]

Suresh Vasudevan
Forbes

요소 3: 컨테이너

가상화, 클라우드 및 컨테이너 기술은 그 목표가 비슷합니다. 즉, 물리적 하드웨어와 별개로 소프트웨어 운영 환경을 추상화하여 하드웨어에 더 많은 인스턴스를 스택하고 가용성, 확장성을 제공하며 효과적인 배포를 지원하는 것입니다. 그러나 이러한 기술은 서로 다른 방식으로 과제를 해결합니다. 가상화는 운영 시스템 레이어를 추상화합니다. 클라우드는 영구적이며, 전용 서버 인스턴스라는 개념을 사용하지 않습니다. 컨테이너는 단일 애플리케이션을 실행하기 위한 라이브러리와 운영 체제의 적절한 버전을 정의합니다.

컨테이너 기술로 제공되는 경량의 규범적인 접근 방식을 통해 컨테이너는 현대적인 소프트웨어 환경에 이상적인 툴이 되었습니다. 각 인스턴스는 운영 체제에서 컨테이너에 포함된 각 라이브러리의 정확한 버전에 이르기까지 불변의 정의를 사용합니다. 그로 인해 해당 단위는 각 인스턴스에 대해 고도의 반복성과 일관성을 유지하므로 지속적 통합과 지속적 제공(CI/CD) 파이프라인에 적합합니다. 경량성과 반복성의 결합은 컨테이너가 애자일 통합에 적합한 기술 플랫폼이 되도록 합니다.

이뿐만 아니라 컨테이너 이미지는 단일 기능 단위에 필요한 요소만 정의하기 때문에, 컨테이너는 마이크로서비스의 잠재력을 실현하고 컨테이너 오케스트레이션은 대규모 마이크로서비스 인프라를 손쉽게 배포하고 관리하도록 지원합니다.

전통적인 통합 접근 방식에서는 ESB가 인프라의 주요 지점에 배치된 고도로 중앙 집중화된 구조가 사용되었습니다. 분산형 통합 및 API 관리에서는 분산된 아키텍처를 통해 특정 위치 또는 팀에 필요한 기능을 배포할 수 있습니다. 컨테이너는 변경이 불가능한 특성 때문에 전체 환경에서 이미지와 배포가 일관되게 유지되어 두 접근 방식에 적합한 기본 플랫폼으로 동작하므로 불투명한 종속성이나 충돌 없이 빠르게 배포하거나 교체할 수 있습니다.

통합을 사용하든 API를 사용하든 분산형 아키텍처의 핵심은 복잡한 승인 프로세스 없이 새로운 서비스를 설계하고 배포할 방법이 있어야 한다는 것입니다.

컨테이너를 사용하면 분산형 통합과 API를 마이크로서비스로 다룰 수 있습니다. 개발 팀과 운영 팀 모두를 위한 공통된 툴링을 제공하고 관리형 릴리스 프로세스와 함께 신속한 개발 프로세스를 사용할 수 있습니다.

마이크로서비스가 별도의 단일 기능을 나타내는 것처럼 각 컨테이너는 단일 서비스 또는 애플리케이션을 나타냅니다. 마이크로서비스 아키텍처에는 수십 개에서 수백 개에 이르는 개별 서비스가 있을 수 있으며 이러한 서비스는 개발, 테스트, 프로덕션 환경 전반에 걸쳐 복제됩니다. 이렇게 많은 수의 인스턴스가 있기 때문에 인스턴스를 오케스트레이션하고 고도화된 관리 태스크를 수행하는 능력은 효과적인 컨테이너 환경을 위해 반드시 필요합니다.

---

11 *Vasudevan, Suresh, "제2의 클라우드 도입 사이클을 촉진하는 컨테이너와 쿠버네티스(Containers And Kubernetes Are Powering The Second Cloud Adoption Cycle)," Forbes. 2019년 7월 10일,* https://www.forbes.com/sites/forbestechcouncil/2019/07/10/containers-and-kubernetes-are-powering-the-second-cloud-adoption-cycle/#171ce5006929

Red Hat OpenShift는 Linux® 컨테이너를 Google의 쿠버네티스 오케스트레이션 프로젝트와 결합하며 인스턴스 관리, 모니터링, 로깅, 트래픽 관리, 자동화 등의 중앙 집중식 관리를 포함합니다. 이는 컨테이너만 있는 환경에서는 거의 제공하기 어려운 기능입니다.

또한 Red Hat OpenShift는 셀프 서비스 카탈로그, 인스턴스 클러스터링, 애플리케이션 지속성, 프로젝트 수준 격리와 같이 개발자에게 친숙한 툴을 제공합니다.

이러한 결합을 통해 특히 안정성과 테스트라는 운영 요구 사항 그리고 간편한 사용과 신속한 제공이라는 개발자 요구 사항 간에 균형을 이룰 수 있습니다.

애자일 인테그레이션 구현

팀 프랙티스

애자일 인테그레이션의 3가지 핵심 요소는 재사용 가능한 기능으로 팀에 배포 및 제공될 때 가장 큰 효과를 발휘합니다. 여기서 '기능'이란 승인된 그룹이 셀프 서비스 방식으로 기술을 사용하고, 조직의 지침을 쉽게 따르며, 모범 사례 정보에 액세스할 권한을 얻는 것을 말합니다.

정보 아키텍트나 IT 관리자는 개별 팀을 위해 다음과 같이 명확한 프로세스를 정의해야 합니다.

- 광범위하게 사용 가능한 사용 지침 제공
- 적절한 경우 사용 방식과 모범 사례 규칙을 적용하되 해당 규칙을 넘어 실험을 할 수 있는 자율성 부여
- 프로토타입에서 테스트, 실행, 업데이트, 사용 만료 단계까지 이동할 수 있도록 명확히 정의된 프로세스 확립
- 신규 배포 및 개발을 위한 정보 공유 허용
- 인프라 팀이 모든 프로세스에 참여하지 않으면서도, 셀프 서비스 기능을 구축하고 제공할 수 있도록 지원

예를 들어, 소프트웨어 팀이 전체 셀프 서비스를 통해 새 API를 실행할 수 있도록 개발, 테스트, 준비하는 것이 가능하며 다른 그룹과 문서를 업데이트하기 위한 프로세스를 갖출 수 있습니다.

게시 또는 프로덕션 단계로 넘어가기 전에 다른 팀과 함께 진행해야 할 프로세스나 크로스 체크가 필요할 수 있으나, 인프라를 통해 이러한 프로세스를 최대한 자동화할 수 있습니다.

인프라 아키텍처

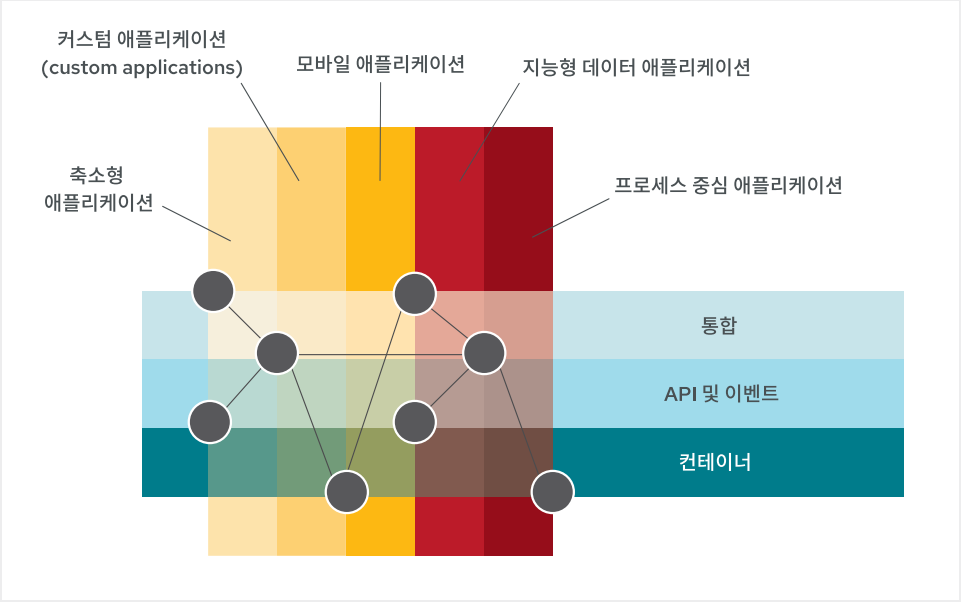

그림 4. 애자일 인테그레이션 기술이 애플리케이션 스택에 미치는 영향

조직의 내부 소프트웨어 에코시스템과 많은 경우에 외부 통합을 위한 액세스 포인트는 동기식 레이어와 비동기식 레이어의 두 가지로 구성됩니다. 컨테이너, API 및 통합은 동기식 레이어에서 연동됩니다. 이벤트, 컨테이너 및 통합은 비동기식 레이어에서 연동됩니다. 이벤트 처리에는 이벤트 메쉬 및 분산형 스트리밍 플랫폼이 포함됩니다.

다양한 시스템에서 재사용 가능한 여러 엔드포인트를 제공합니다. 이러한 엔드포인트 각각은 재사용 가능한 API로 표시되며 컨테이너 내에서 실행되어 확장성과 배포 편의성을 구현합니다. 통합 프로세스에서는 개별 서비스 그룹을 통합하거나 조직의 여러 부분에서 얻은 결과를 수집하여 시스템 전반에서 필요한 곳에 트랜스포메이션, 컴포지션 또는 인라인 비즈니스 로직을 제공합니다.

통합된 애플리케이션은 최종 사용자 애플리케이션을 제공하기 전에 더 긴밀하게 통합될 수 있습니다.

모든 시스템이 점점 더 작은 부분으로 분해되고 여러 레이어의 API 추상화를 통과한다는 가정은 없습니다. 이러한 운영은 효율성을 낮추고 지연 시간을 늘리며 불필요한 복잡성을 가중시킬 수 있습니다. 일부 영역에서는 기존의 레거시 ESB 기능을 유지하면서 특정 애플리케이션 간의 연결을 그대로 두는 것이 적절한 선택일 수 있습니다. 분산형 시스템 간의 종속성 또한 적절한 툴을 사용하여 추적하고 관리해야 합니다.

하지만 시스템 전체를 고려했을 때 컨테이너, API, 통합 측면에서 아키텍처를 재구성하는 것은 서비스, 통합 지점, 고객 상호 작용에 대해 적절한 선택을 할 수 있다는 의미가 됩니다. 예를 들어, 대량의 인바운드 요청이 들어오면 보안을 점검한 후 올바른 백엔드 서비스에 바로 라우팅할 수 있으며 이때 ESB 병목 현상은 발생하지 않습니다.

하이브리드, 분산형 클라우드 환경에서, 문제의 백엔드 시스템은 다양한 물리적 위치에 다수 존재할 수 있습니다. 로컬 요구를 충족하기 위해 인접한 로컬 시스템을 통합하면 주요 비즈니스 로직이 포함된 단일 중앙 통합 시스템을 통해 모든 것을 라우팅하는 것보다 높은 효율성과 보안을 실현할 수 있습니다.

애자일 조직과 문화

인프라 라이프사이클은 소프트웨어 개발이나 운영의 라이프사이클과는 매우 다릅니다. 소프트웨어 개발 사이클은 하나의 프로젝트를 완료한 후 다음 프로젝트로 이동하며, 제품을 얼마나 빨리 출시하느냐 또는 주어진 시간에 기능을 얼마나 많이 생성하느냐가 효율성의 기준이 됩니다. 유지관리와 안정성이 중시되는 운영 사이클에서도 더욱 신속하고 효율적인 보안 패치와 업데이트 적용, 신규 서비스 배포, 또는 변경 사항 롤백은 여전히 큰 이점을 가져옵니다.

하지만 인프라에는 상당히 다른 접근 방식이 있습니다. 인프라는 고도로 전문화된 여러 그룹이 많은 시간을 들여 구축하는 것이 대부분이며 다기능팀(cross-functional team)이 특정 소프트웨어 엔지니어링 프로젝트를 진행하는 것과는 매우 다릅니다.

인프라 프로젝트는 일반적으로 소프트웨어 프로젝트보다 규모가 훨씬 크며 릴리스 사이클이 짧을 경우 많은 성과를 내는 것이 불가능해질 수 있고 프로젝트가 불완전한 상태로 남겨질 수도 있습니다. 엔터프라이즈 IT 전문가인 Andrew Froehlich는 InformationWeek에서 인프라가 특히 하드웨어와 데이터센터와 관련하여 귀환 불능 지점이라는 제약을 가진다고 밝힌 바 있습니다. 퍼블릭 클라우드의 경우에도 더 이상 프로젝트를 폐기하고 다시 시작할 수 없는 지점이 존재합니다.[12] 인프라는 영구적입니다. 하지만 인프라 성능에 관해 방법론을 조정할 수 있습니다.

애자일 및 DevOps와 같은 응답성 높고 반복적인 프로세스를 통해 개발 팀과 운영 팀은 큰 이점을 얻을 수 있는 반면, 인프라 팀은 동일한 이점을 획득하기 어렵습니다. 하지만, 최대 효율을 위해서는 인프라 팀과 개발 및 운영 팀을 연계하는 것이 중요합니다. 글로벌 컨설팅 회사인 McKinsey는 "애자일 트랜스포메이션을 통해 IT 인프라 조직을 현대화하기란 쉽지 않지만 그럴 만한 가치가 있습니다. 경험에 비춰볼 때, IT 인프라 그룹은 애자일 접근 방식을 통해 조직 규모에 따라 6개월에서 18개월 내에 생산성을 25-30% 향상했습니다"라고 밝혔습니다.[13]

애자일 통합 기술은 보다 민첩한 인프라를 뒷받침하며 API, 컨테이너 이미지, 분산형 통합은 소프트웨어 인프라 언급에서 새로운 주제로 부상했습니다.

---

12 *Froehlich, Andrew, "애자일 IT 구현에 관한 찬반 양론(Should IT go agile? The pros and cons)." 2015년 10월 6일,* http://www.informationweek.com/infrastructure/pc-and-servers/should-it-go-agile-the-pros-and-cons/d/d-id/1322448

13 *Cormella-Dorda, Santiago 외. "애자일을 통한 IT 인프라 조직 혁신(Transforming IT infrastructure organizations using agile)." McKinsey Digital, 2018년 10월,* https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/transforming-it-infrastructure-organizations-using-agile

애자일 선언문(Agile Manifesto)에는 소프트웨어 개발의 네 가지 핵심 원칙이 정의되어 있습니다.[14] 해당 원칙들은 애자일한 통합 기반 인프라에서 통합 전략에 적용할 수 있습니다.

1	**프로세스와 툴보다 개인과 개인 간의 상호작용이 더 중요하다** 인프라가 갖춰진 상태에서는 팀 간의 상호 작용에 초점이 맞춰집니다. 상호 작용에는 API, 메시징, 트래픽 패턴 등에 의해 제어되는 직접적인 커뮤니케이션, 시스템 차원의 상호 의존성, 테스트 및 릴리스 프로세스(예: CI/CD 파이프라인)가 포함됩니다.
2	**포괄적인 문서보다 작동하는 소프트웨어가 더 중요하다** 인프라는 그 특성상 상시 작동되어야 하며 급격하게 바뀌는 것이 아니라 서서히 조정되어야 합니다. 따라서 기본적으로 중단 없이 작동하는 인프라가 요구됩니다. 인프라 전략 측면에서 '작동한다'는 말은 인프라 구성 요소를 통해 적정 성능 범위 내에서 최종 사용자 동작이 적절하게 이루어진다는 의미입니다.
3	**계약 협상보다 고객과의 협업이 더 중요하다** 인프라 시스템이 갖춰지면 인프라팀이 시스템 의존성을 관리하는 방법(예: 보안 정책, 서비스 수준 계약, 게시된 API)이 계약서에 명시되며 이 시스템의 내부 및 외부 사용자가 모두 고객이 됩니다. 민첩성이 구현되면 시스템에 관련된 정책과 인터페이스가 변경될 때 사용자가 이를 알 수 있을 뿐 아니라, 이러한 변경을 더 신속하게 적용할 수 있습니다. 분산형 통합을 사용하면 여러 팀이 통합을 개발하고 배포할 수 있게 되어 협업의 범위가 확장됩니다.
4	**계획을 따르기보다 변화에 대응하는 것이 더 중요하다** 이 원칙에 기반을 두고 기술을 통해 프로세스를 지원할 수 있습니다. 인프라를 위해 시스템의 안정성은 계속 유지하되, 컨테이너와 같은 신기술을 사용하여 탄력적인 플랫폼을 제공해야 합니다. 요구에 따라 인스턴스를 동적으로 추가하거나 삭제하고, 배포 및 업데이트를 자동화하며, 여러 인스턴스에서 변경을 오케스트레이션할 수 있습니다. 게시된 API 정의를 통해 재사용 가능한 툴을 사용하면 개발에서 일관성이 유지되며 이 접근 방식을 기반으로 변화에 적응할 수 있는 안정적인 플랫폼을 구축할 수 있습니다.

그림 5. 애자일 선언문(Agile Manifesto)에 명시된 소프트웨어 개발의 핵심 원칙

애자일 인테그레이션 아키텍처는 기술을 통해 인프라 팀 내의 문화가 바뀌도록 지원하여 인프라 전략을 위한 기반을 구축합니다. 이로 인해 인프라 기술과 팀이 개발 및 비즈니스 전략에 더 긴밀히 연계되도록 할 수 있습니다.

애자일 방법론을 사용하면 개인, 빌드, 종속성과 같은 소프트웨어 프로젝트의 핵심 요소를 식별한 후 이들 요소 간의 관계를 정의할 수 있습니다. 애자일 프로젝트로서 통합 인프라에 접근하는 경우 팀, 컨테이너 이미지, API, 통합 지점 등 애자일로 정의되는 유사 요소와 관계가 존재한다는 점을 발견할 수 있습니다. 표 3에서는 이러한 유사 요소 중 일부를 설명합니다.

14 *애자일 선언문(Agile Manifesto),* http://agilemanifesto.org/

표 3. 소프트웨어 애자일과 인프라 애자일 요소 비교

프로젝트	조직	세부 정보
개인	팀	팀은 인프라의 특정 부분을 담당합니다. 팀에서 관리하는 시스템과 API, 팀 리더, 팀 목표 등 팀이 담당하는 업무와 관련된 정보를 식별합니다.
모듈	API	인터페이스(API)가 명확히 정의되어 있고 장기간 안정적이며 자체 로드맵을 보유하여 특정 팀에 의해 실행됩니다. 또한 조직 내에서 중요한 특정 기능을 구축합니다.
빌드	컨테이너 이미지	릴리스는 테스트를 거쳐 태그가 지정된 배포 가능 유닛을 기반으로 하며 액세스 권한을 가진 팀이 안정적으로 배포할 수 있습니다. 이는 모놀리식 버전 코드를 대체합니다.
컴파일 종속성	통합	이 요소는 이러한 분산형 시스템 내부의 다양한 구성 요소 간 통합과 매핑을 식별합니다. 그런 다음 시스템의 다른 부분처럼 통합 지점에 대해 관리, 커미셔닝, 디커미셔닝, 버전 관리, 테스트를 수행합니다.
빌드 테스트	인프라 자동화	소프트웨어 빌드, 성능 및 사용자 요구 사항을 테스트하는 기능부터 여러 시스템을 운영하고 모니터링하는 작업에 이르는 전체 라이프사이클 관리입니다.

애자일 원칙을 인프라 계획에 적용

대부분의 변경 관리 접근 방식에서는 모든 하위 시스템을 종합적으로 문서화해야 합니다. 이 문서는 모니터링 방법에서 성능 매개 변수 그리고 담당 팀에 이르기까지 시스템의 모든 측면을 상세하게 다루어야 합니다. 애자일 원칙에서는 협력과 적응성이 요구되며 이는 많은 문서화 작업을 필요로 하는 변경 관리와 충돌합니다.

잠재적인 이해 관계자, 변경 사항, 시스템 구성 요소 전체를 규범적으로 정의하려 할 것이 아니라 변경 요청과 계획을 평가하는 데 사용할 수 있는 일련의 지침과 표준을 정의해야 합니다. 다음 질문에 대한 답을 생각해 보십시오.

- 사용자를 위해 설계된 엔드 투 엔드 환경은 무엇인가?
- 각 팀, API, 시스템 등이 시간이 지남에 따라 이러한 환경을 개선하기 위해 어떻게 기여하고 있는가?
- 모니터링과 경고는 어떤 매개변수로 어떻게 정의되며 서비스 수준을 어떻게 유지하는가?
- 작동이 제대로 되는지 확인하기 위해 어떤 유형의 자동화된 테스트가 필요한가?
- 사용자 환경을 방해하지 않으면서 팀이 자체 하위 시스템의 새 버전을 테스트하고 배포하기 위한 릴리스 파이프라인은 무엇인가?
- 구성 요소 서비스의 장애가 전체 시스템의 서비스 수준에 어떤 영향을 미치는가?

애자일 인프라 내에서의 변경 관리는 계약보다는 지속적인 협력을 통해 이루어져야 합니다.

목표를 달성할 확률

귀사의 IT 프로젝트가 성공할 가능성은 어느 정도일까요? 먼저 사양을 충족하는 것, 고객 채택률을 높이는 것, 출시를 완료하는 것 등 성공의 기준을 파악하는 것이 중요합니다. Project Management Institute가 실시한 설문조사에 따르면 지난 5년간 계획한 목표를 달성한 프로젝트의 수가 증가했음이 밝혀졌습니다. 이렇게 수치가 증가한 것은 IT 팀과 비즈니스 팀이 밀접하게 연계되어 자사 전략과 고객 요구 사항에 대해 더 나은 정보를 활용할 수 있기 때문이라고 합니다.[15]

이렇게 전략적 연계를 구축하는 이유 중 하나는 애자일 팀을 구현하는 것입니다. 애자일 방식은 협력과 피드백, 시스템과 문제에 대한 전체적인 시각 및 창의적인 접근 방식을 권장합니다.

공유된 기술 스택이 있으면 개별 코드에서 벗어나 시스템과 상호의존성을 논의할 수 있게 됩니다. 즉, 시스템 수준의 사고를 하면서 내부에서 개발한 소프트웨어, 벤더 시스템 그리고 이들 간의 연결을 포함한 전체 소프트웨어 인프라 컬렉션을 단일 시스템으로 다루는 것입니다. API와 메시징 시스템은 전체 인프라를 포괄하여 소프트웨어 시스템을 통합할 수 있습니다.

API와 분산형 통합은 개발팀 또는 운영팀 내에서 개발되고 파악될 수 있으므로 통합과 관련하여 팀의 책임을 더 명확히 알 수 있습니다. 개발과 배포를 처리하는 팀이 시스템과 애플리케이션 간의 상호의존성을 인식할 수 있기 때문에 통합 자체를 더 면밀히 파악할 수 있게 됩니다.

인프라를 위한 기반으로 통합을 사용하고 이 통합과 관련된 책임을 여러 팀에 분산하면 애자일 접근 방식을 적용하기 알맞은 인프라 환경이 구축됩니다.

15 *Florentine, Sharon, "IT 프로젝트의 성공률, 마침내 증가하다(IT project success rates finally improving)." 2017년 2월 27일.* https://www.cio.com/article/3174516/project-management/it-project-success-rates-finally-improving.html

결론: 애자일 인테그레이션 실현

민첩성은 프로젝트가 아니라 프로세스입니다.

시장 변화에 대응하는 능력이 오늘날처럼 중요한 때도 없었으며 이를 위해서는 IT 시스템이 변화를 인식하여 새로운 서비스를 출시하거나 기존 서비스를 빠르게 업데이트할 수 있어야 합니다. 디지털 서비스의 기반이 되는 IT 인프라를 재고하는 것도 매우 중요합니다.

지금까지 인프라팀은 위험 완화와 안정성 유지라는 요구 때문에 시간이 많이 소요되는 모듈레이티드(Modulated) 프로세스에 얽매여 있었습니다. 하지만 인프라를 보는 사고방식을 하드웨어 또는 플랫폼 기반에서 통합 기반으로 전환할 수 있습니다. 통합은 인프라의 일부가 아니라 데이터, 애플리케이션, 하드웨어, 플랫폼이 포함된 인프라라는 개념의 접근 방식입니다.

Red Hat은 애자일 인테그레이션의 3가지 핵심 요소인 분산형 통합, API 및 컨테이너를 사용하여 민첩성과 적응성이 더 높은 인프라를 구축하는 접근 방식을 애자일 통합이라고 정의합니다. 기술은 문화의 변화를 지원하는 데 사용해야 합니다. 이는 소프트웨어뿐 만 아니라 인프라팀이 더 민첩해지도록 노력하는 것을 의미합니다. 인프라팀은 애자일 원칙을 적용하면서 기술을 서서히 도입하여 이러한 변화를 지원할 수 있습니다. 하나의 프로젝트에서 전체 조직을 근본적으로 재구성하여 민첩하게 만들 수는 없습니다. 하나의 애자일 인테그레이션 기술을 구현하거나 비즈니스의 한 영역을 변경한 후 변화를 점진적으로 확대해 나가는 것이 효과적입니다.

변화에 대한 IT 인프라의 대응력을 높이는 것은 장기적이고 전략적인 목표입니다. 이러한 목표를 달성하기 위해 조직 전체를 아우르는 전면적인 변화가 필요한 것은 아니며 격리된 상태에서 변화를 적용한 후 롤아웃하는 것조차 필요하지 않을 수 있습니다.

애자일 인테그레이션은 IT 인프라를 재구성하는데 필요한 기술적이면서도 조직적인 프레임워크를 제공합니다.

한국레드햇 홈페이지 https://www.redhat.com/ko

RED HAT 정보

Red Hat은 세계적인 엔터프라이즈 오픈소스 솔루션 공급업체로서 커뮤니티 기반 접근 방식을 통해 신뢰도 높은 고성능 Linux, 하이브리드 클라우드, 컨테이너, 쿠버네티스 기술을 제공합니다. 또한 고객으로 하여금 신규 및 기존 IT 애플리케이션을 통합하고, 클라우드 네이티브 애플리케이션을 개발하며, 업계를 선도하는 Red Hat의 운영 체제를 기반으로 표준화하는 동시에 복잡한 환경의 자동화, 보안 및 관리를 실현할 수 있도록 지원합니다. Red Hat은 전세계 고객에게 높은 수준의 지원과 교육 및 컨설팅 서비스를 제공하여 권위있는 어워드를 다수 수상한 바 있으며, Fortune 선정 500대 기업의 신뢰를 받는 어드바이저로 인정받고 있습니다. 또한 기업, 파트너, 오픈소스 커뮤니티의 전략적인 파트너로서 고객들이 디지털 미래에 대비할 수 있도록 지원하고 있습니다.

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