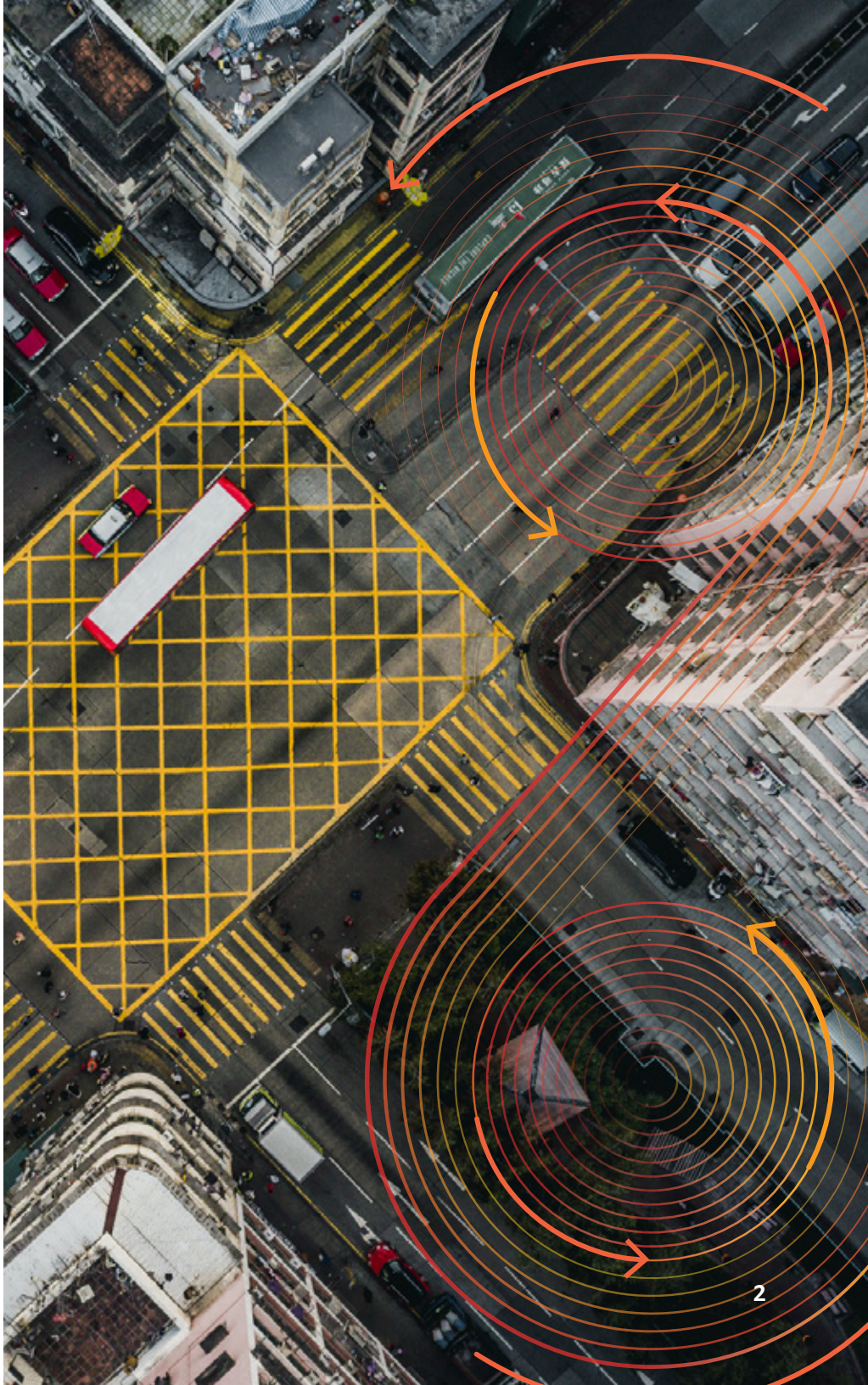

aws modern apps
AWS 기반의 현대적 애플리케이션 구축
효율적인 관리. 빠른 구축. 더 많은 혁신.

고객 가치 제공 방식을 바꾸는 현대적 애플리케이션

지난 40년간 개발된 앱을 모두 합친 것보다 더 많은 5억 개 이상의 새로운 앱이 향후 몇 년 안에 구축될 것입니다.[1] 많은 기업이 기술 관리와 새로운 기능 제공 사이의 균형을 찾기 위해 고군분투하면서 힘들게 이러한 앱을 구축하고 있습니다.

클라우드는 민첩성을 약속하지만 자동으로 이루어지지는 않습니다. 혁신을 가속화하고, 데이터를 최대한 활용하며, 새로운 고객 경험을 구축하려는 조직은 애플리케이션 구축 및 운영 방식을 현대화해야 합니다. 현대적 애플리케이션은 모듈식 아키텍처 패턴, 서버리스 운영 모델 및 애자일 개발 프로세스의 조합으로 구축됩니다.

이 eBook에서는 조직에서 현대적 애플리케이션 개발을 위한 토대를 마련하는 데 도움이 되는 세 가지 경로를 안내합니다. 또한 AWS를 사용하는 현대적 애플리케이션 개발이 조직의 혁신, 비용 절감, 출시 기간 단축, 안정성 향상에 어떻게 도움이 되는지 살펴보겠습니다.



[1] *IDC PlanScape: Connected Cloud Architectures to Enable the Future of Digital Infrastructure, 2020년 10월*

혁신이란 고객에게 귀를 기울이는 것

Forrester Research는 최신 비전 보고서, *Digital Rewrites the Rules of Business*에서 디지털 혁신 기업의 고객 중심 사고방식을 정의했습니다. 이러한 현대적 혁신 기업의 핵심 미션은 다음과 같습니다.

"...디지털 사고를 고객 경험, 운영, 에코시스템 및 혁신에 적용함으로써...디지털 자산과 에코시스템을 통해 고객 성과를 지속적으로 개선하는 동시에 운영 우수성을 개선합니다."

고객에게 집중한다는 것은 고객의 관점에서 시작하여 역방향으로 작업하여 비즈니스 의사 결정을 내리는 것을 의미합니다. 또한 고객 만족도 향상이라는 결과를 낼 수 있도록 끊임없이 제품과 서비스를 발전시켜야 한다는 뜻입니다. 그리고 고객이 진정으로 원하는 것이 무엇인지에 귀를 기울여 고객의 입장에서 창조와 혁신을 계속 반복해 나가야 한다는 것을 의미합니다. 이를 '혁신 플라이휠'이라고 합니다.

혁신 플라이휠의 기본 개념은 혁신은 고객 요구에서 시작하여 고객 피드백으로 개선되며 고객 요구가 바뀌어 전체 주기가 다시 시작될 때까지 끊임없이(수익을 내면서) 반복된다는 것입니다. 팀이 자체 혁신 플라이휠을 더 빨리 시작할수록 현대적 애플리케이션을 더 잘 구축할 수 있고 경쟁업체와의 격차도 더 벌릴 수 있습니다.

AWS 기반의 현대적 애플리케이션을 구축하면 출시 기간을 앞당길 수 있습니다. 개발자는 구축 및 릴리스 주기를 단축하고 운영 오버헤드를 줄임으로써 새로운 기능을 신속하게 구축할 수 있습니다. 전체 애플리케이션에 미치는 위험을 최소화하면서 개별 애플리케이션 구성 요소를 실험할 수 있도록 지원하는 모듈식 아키텍처를 통해 더 많은 혁신을 이뤄낼 수 있습니다. 테스트 절차를 자동화하고 개발 수명 주기의 모든 단계를 모니터링함으로써 안정성을 개선합니다. 또한 유휴 리소스에 대한 초과 프로비저닝 또는 지급으로 인한 비용을 줄이는 가치 기반 요금 모델을 통해 총 소유 비용(TCO)을 개선합니다.

현대적 애플리케이션을 구축하려면 구축 기반을 재고해야 할 수도 있습니다. 조직 차원에서는 이러한 아키텍처 변화가 매우 극적이지만, 그렇다고 한 번에 모든 것을 바꾸려고 할 필요는 없습니다. 많은 기업이 클라우드에서 새로운 현대적 앱을 구축할 때 획기적인 전환을 도모하지만, 팀별 및 워크로드별 여정을 통해 기회가 있을 때 한 번에 한 단계씩 이동하는 하이브리드 방식을 취하는 기업들도 많습니다.

50%

2023년까지 디지털 트랜스포메이션의 직접적인 대상이 될 것으로 예상되는 정보 통신 기술의 비율

67%

경쟁력을 유지하기 위해서는 속도를 높여야 한다고 믿는 경영진의 비율

90%

2025년까지 클라우드 네이티브로 구현될 것으로 예상되는 신규 애플리케이션의 비율

Amazon.com을 위한 애플리케이션을 구축하고 수백만 AWS 고객에게 서비스를 제공한 경험을 바탕으로 AWS는 조직이 애플리케이션 현대화에 대한 비전을 실현하고 그 과정에서 비즈니스 가치를 창출하기 위해 취할 수 있는 세 가지 경로를 찾아냈습니다.

1. 관리형 컨테이너 서비스로 리플랫포밍. 이미 온프레미스에서 컨테이너를 실행 중이거나 애플리케이션을 컨테이너로 이전하려는 조직은 해당 워크로드를 AWS 기반의 컨테이너 서비스로 리플랫포밍하여 운영을 단순화하고 오케스트레이션 및 인프라 프로비저닝과 같은 관리 오버헤드 비용을 줄일 수 있습니다.

2. 서버리스 아키텍처를 기반으로 새 앱 구축. 새 애플리케이션이나 기능을 구축하려는 조직은 서버리스 기술과 목적별 데이터베이스를 사용하여 민첩성을 극대화하고 고급 개발 도구를 사용하여 개발을 가속화하는 것이 좋습니다.

3. 현대적 Dev+Ops 모델로 변환. 현대적 애플리케이션을 대규모로 구축하는 문화적 변화를 만들기 위해 조직은 보안 및 거버넌스에 대한 높은 기준을 유지하면서 DevOps 서비스 및 도구를 활용할 수 있습니다.

각 경로에 대해 더 자세히 살펴보고 각각이 어떻게 민첩성을 높이고 비용을 낮추며 비즈니스 성공을 지원하는 더 나은 앱을 구축하는 데 도움이 되는지 알아보겠습니다. 애플리케이션 현대화를 언제 시작하든 상관없이 성과는 동일해야 합니다. 즉, 처음부터 안전하고 안정적이고 확장 가능하며 고객과 파트너가 더 빠르게 사용할 수 있는 애플리케이션을 구축하는 것입니다.

현대적 애플리케이션 개발로 가는 세 가지 경로

현대적 애플리케이션 개발은 클라우드에서 소프트웨어를 설계, 구축 및 관리하는 강력한 접근 방식입니다. 이러한 검증된 접근 방식을 사용하면 개발 팀의 민첩성과 애플리케이션의 안정성 및 보안이 향상되어 더 나은 제품을 더 빨리 개발하고 출시할 수 있습니다. 다양한 분야의 조직에서 애플리케이션을 구축하는 과정을 지원한 경험을 바탕으로 현대화를 향한 여정에 도움이 되는 현대적 애플리케이션 개발의 세 가지 원칙을 알아냈습니다.

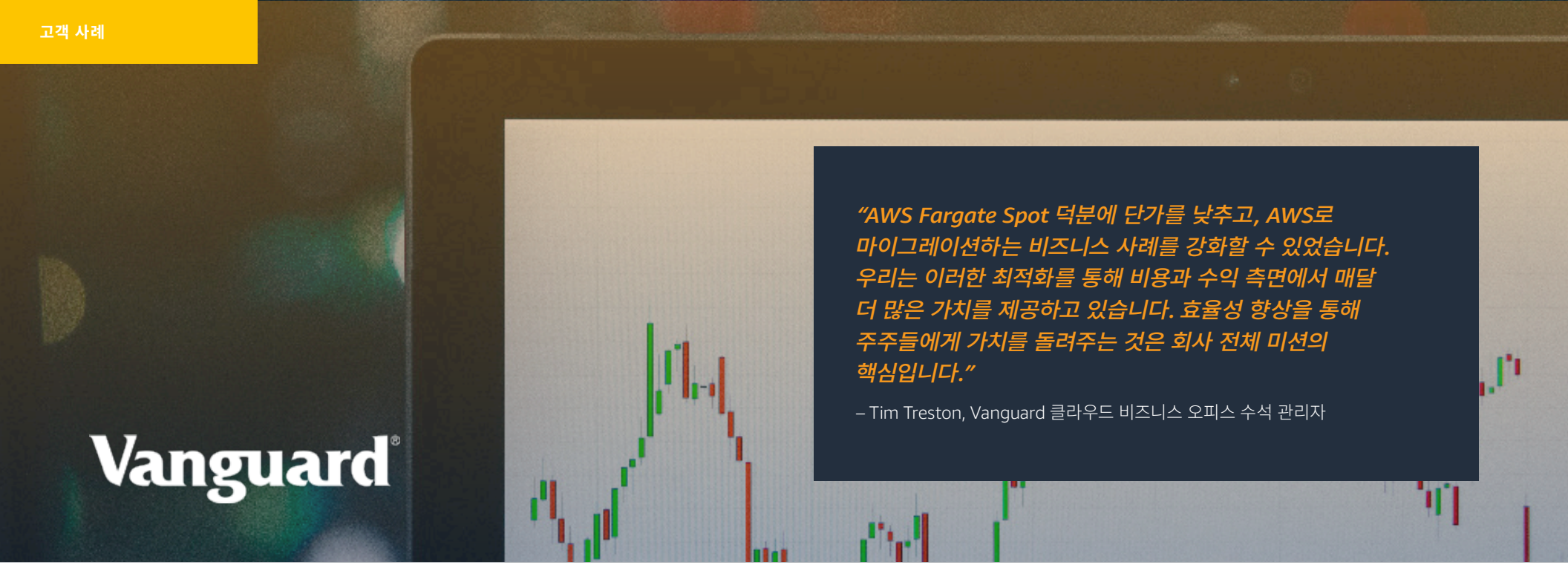
현대적 애플리케이션으로 가는 경로

1. 관리형 컨테이너 서비스로 리플랫포밍
2. 서버리스 아키텍처를 기반으로 새 보안 앱 구축
3. 현대적 Dev+Ops 모델로 변환

관리형 컨테이너 서비스로 리플랫포밍

애플리케이션을 실행하고 배포할 때, 컨테이너는 가볍고 이동성이 좋은 방법입니다. 기존 애플리케이션을 컨테이너화하는 것이 조직 현대화 여정의 첫 단계인 경우가 많습니다. 애플리케이션을 컨테이너로 이전하는 것을 고려하고 있다면 AWS Fargate를 통해 이러한 워크로드를 Amazon Elastic Kubernetes Service(Amazon EKS) 또는 Amazon Elastic Container Service(Amazon ECS)와 같은 AWS 관리형 서비스로 리플랫포밍하는 것이 좋습니다.

관리형 컨테이너 서비스는 확장성, 안정성, 보안성 및 가용성을 향상하면서 운영 부담을 줄이는 데 도움이 됩니다. AWS 기반의 관리형 컨테이너 서비스를 사용하면 더 이상 컨테이너 관리에 대해 걱정할 필요가 없습니다. 대신 서버리스 컴퓨팅으로 현대적 애플리케이션을 개발하는 데 필요한 기술 역량 교육에 리소스를 집중할 수 있습니다.



> *"AWS Fargate Spot 덕분에 단가를 낮추고, AWS로 마이그레이션하는 비즈니스 사례를 강화할 수 있었습니다. 우리는 이러한 최적화를 통해 비용과 수익 측면에서 매달 더 많은 가치를 제공하고 있습니다. 효율성 향상을 통해 주주들에게 가치를 돌려주는 것은 회사 전체 미션의 핵심입니다."*
>
> – Tim Treston, Vanguard 클라우드 비즈니스 오피스 수석 관리자

Vanguard 그룹은 2015년 AWS로 마이그레이션을 시작한 이후 상당한 이점을 얻었습니다. 예를 들어 Vanguard는 AWS 서비스를 사용했기 때문에 IT 부서에서 서버를 관리할 필요가 없었습니다. 결과적으로 개발자는 혁신적이고 새로운 마이크로서비스를 구축하고 현재 애플리케이션을 개선하는 데 더 많은 시간을 할애하여 Vanguard의 출시 속도를 3개월에서 24시간으로 단축했습니다.

그러나 출시 속도 단축 외에도 Vanguard는 다른 이점을 얻었습니다. Vanguard는 AWS Fargate와 함께 완전관리형 컨테이너 오케스트레이션 서비스인 Amazon Elastic Container Service(Amazon ECS)를 사용하기로 결정했습니다. AWS Fargate는 클라우드에서 안전하고 크기 조정 가능한 컴퓨팅 용량을 제공하는 웹 서비스인 Amazon Elastic Compute Cloud(Amazon EC2)를 프로비저닝하고 관리할 필요가 없는 서버리스 컴퓨팅 서비스입니다. 그리고 이 금융 서비스 회사는 AWS Fargate에 대한 새로운 구매 옵션을 활용하여 단가를 50% 절감했습니다.

전체 내용 보기 »

앱 개발 현대화는 개발자가 보다 민첩한 애플리케이션을 구축할 수 있도록 서비스, 사례 및 전략을 도입하는 것입니다. 또한 현대적 인프라의 속도와 안정성을 통해 개발자는 프로토타입에서 수백만 명의 사용자가 이용하는 보안 앱으로 자동 확장할 수 있으므로 혁신 및 변화에 더 빠르게 대응할 수 있습니다.

많은 현대적 애플리케이션이 서버리스 우선으로 구축되며, 이는 고객이 애플리케이션 스택 전체에서 민첩성을 높일 수 있도록 서버리스 서비스 도입을 우선시하는 전략입니다. 서버리스 기술을 사용하면 더 이상 물리적 서버를 관리할 필요가 없으며 자동 크기 조정, 기본 제공되는 고가용성 및 가치 기반 요금 모델의 이점을 누릴 수 있습니다. 서버 또는 런타임 관리 및 운영에 대해 걱정하는 대신 제품 혁신에 집중하는 동시에 출시 기간을 단축할 수 있습니다.

또한 AWS를 기반으로 프런트 엔드 웹 및 모바일 도구와 서비스를 구축할 수 있습니다. 이 인프라의 안정성은 브랜드를 전 세계로 자동 확장할 수 있는 안전한 고가용성 앱을 배포하는 데 도움이 됩니다.

확장 가능한 현대적 앱 구축에 관한 주요 고려 사항

아키텍처 패턴: 마이크로서비스

당장은 모놀리식 앱이 관리하기 쉬울 수 있지만, 규모가 커질수록 팀 간에 앱에 대한 권한과 책임을 분배하는 방법 등에서 문제가 점점 발생하기 시작합니다. 강력한 주인 의식 문화가 조성되더라도 애플리케이션 아키텍처 내부의 종속성 때문에 최종 제품에 대한 주인 의식을 부여하기 어렵다면 이러한 주인 의식 문화를 확장시키기가 어렵습니다. 따라서 빠르게 성장하고 변화하는 애플리케이션의 경우 마이크로서비스 아키텍처를 구축하는 것이 좋습니다. 마이크로서비스는 주인 의식 문화를 아키텍처로 표현한 것으로, 복잡한 애플리케이션을 단일 팀이 독립적으로 소유하고 실행할 수 있는 구성 요소로 깔끔하게 나눕니다.

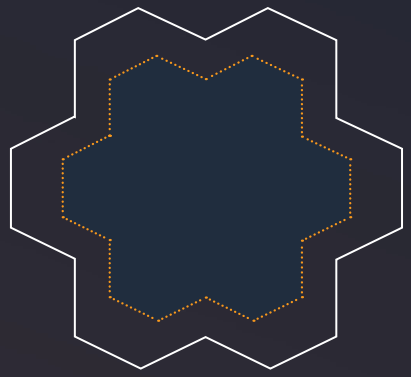
모놀리식의 경우 다수의 개발자가 공유 릴리스 파이프라인을 통해 모든 변경 사항을 푸시하므로 수명 주기의 여러 지점에서 마찰이 발생합니다. 개발 과정에서는 엔지니어가 다른 엔지니어의 코드에 영향을 미치지 않도록 변경 사항을 조율해야 합니다. 새로운 기능을 활용하기 위해 공유 라이브러리를 업그레이드하려면 모든 사용자가 동시에 업그레이드하도록 설득해야 하는데 이는 어려운 요청입니다. 중요한 기능 수정 사항을 신속하게 푸시하려는 경우에도 진행 중인 변경 사항과 병합해야 합니다.

개발 후 전달 파이프라인을 통해 변경 사항을 푸시할 때도 오버헤드 문제가 발생합니다. 작은 코드 한 줄을 변경하더라도 엔지니어들이 미리 변경 사항을 조율하고, 코드를 병합하며, 릴리스 시 충돌을 해결하고, 전체 애플리케이션을 다시 구축하며, 일련의 모든 테스트를 실행하고, 다시 배포해야 합니다.

마이크로서비스 아키텍처를 사용하면 하나의 애플리케이션은 각 애플리케이션 프로세스를 서비스로 실행하는 독립적인 요소로 구성됩니다. 서비스는 해당 비즈니스의 규모에 맞게 구축되며 각 서비스가 하나의 기능을 수행합니다. 서비스가 독립적으로 실행되고 단일 개발 팀에서 관리하므로, 애플리케이션의 특정 기능에 대한 요구에 맞춰 각 서비스를 업데이트, 배포 및 확장할 수 있습니다. 예를 들어, 세일 중에는 더 많은 사용자가 온라인 쇼핑 장바구니 기능을 사용합니다. 마이크로서비스는 가벼운 API, 이벤트 또는 스트림을 사용하여 잘 정의된 인터페이스를 통해 서로 데이터를 통신합니다. AWS 고객은 비용을 절감하면서 애플리케이션 확장성과 안정성을 개선하기 위해 데이터가 변경되면 동작이 트리거되는 이벤트 기반 아키텍처를 점점 더 많이 사용하고 있습니다.

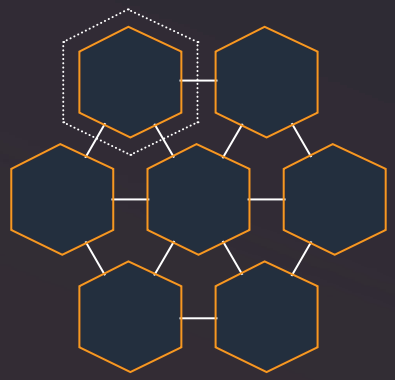
통합 관리 및 개별 관리 비교: 애플리케이션의 두 가지 유형

모놀리식 앱



- 모든 작업 수행
- 단일 앱
- 전체 앱 배포 필요
- 하나의 데이터베이스
- 기술 계층 중심의 구성
- 각 런타임 인스턴스에 상태 존재
- 전체 앱에 하나의 기술 스택 사용

마이크로서비스



- 하나의 작업 수행
- 최소 기능 서비스
- 개별 배포 및 상호 연동
- 각각 자체 데이터스토어 보유
- 비즈니스 기능 중심의 구성
- 상태의 외부화
- 마이크로서비스별로 기술 선택

Centrica는 영국의 다국적 에너지 및 서비스 회사로, 애플리케이션 아키텍처 방식을 변경하여 비용을 절감하고 민첩성을 높이고자 했습니다. 이 회사는 이러한 목표를 달성하기 위해 마이크로서비스 아키텍처로 리팩터링하고 서버리스 전략을 도입하기로 결정했습니다.

조직을 바꾸기 위해 주요 팀으로 서버리스 실무 그룹을 구성하고 함께 파일럿을 구축하기 시작했습니다. 이러한 성공을 바탕으로 조직 내의 다른 팀들도 구축 업무에 이 접근 방식을 도입했으며 이제는 서버리스가 조직 전체에 보편화되었습니다.

이제 서버리스를 통해 고객 문제를 실시간으로 파악하고 대응할 수 있게 되었습니다. 이전에는 불가능했던 부분입니다.

동영상 보기 »

서버리스 운영 모델

가능한 한 서버리스로 구현

아키텍처 패턴 및 소프트웨어 제공 프로세스가 변경되면서 비즈니스의 핵심 역량이 아닌 작업은 줄이는 운영 모델을 도입하게 될 것입니다. 빠른 혁신을 가능하게 하는 민첩성을 확보하기 위해서는 마이크로서비스 아키텍처를 구축하고, 애플리케이션의 모니터링, 프로비저닝, 비용 관리, 배포, 보안 및 거버넌스와 같은 작업을 자동화하여 소프트웨어를 운영 및 배포하는 것이 좋습니다. 가능한 경우 최대한 서버리스 기술을 적용하는 서버리스 우선 전략을 선택하면 AWS의 운영 이점을 극대화할 수 있습니다.

서버리스 운영 모델을 사용하면 서버를 프로비저닝 및 관리하지 않고도 애플리케이션과 서비스를 구축하고 실행할 수 있습니다. 이는 서버 관리를 없애고, 유연한 크기 조정 기능을 제공하며, 사용한 가치에 대한 비용만 지불할 수 있도록 하고, 고가용성을 자동화합니다. 이 모델을 사용하면 관련 세부 사항에 대한 걱정 없이 고객 가치를 제공하는 애플리케이션 부분에 초점을 두고 구축 및 관리할 수 있습니다.

완전히 새로운 애플리케이션을 구축하거나 레거시 애플리케이션을 마이그레이션할 때, 컴퓨팅, 데이터 및 통합을 위한 기본 요소를 서버리스로 구축하면 클라우드가 제공하는 탁월한 민첩성을 활용할 수 있습니다.

AWS에서는 서버리스를 어떻게 정의할까요?

서버리스라고 하면 서버 운영이라는 획일적 업무 부담을 제거하는 것을 의미합니다. 이는 애플리케이션을 지원하기 위한 인프라 관리 및 크기 조정보다는 애플리케이션을 구축하는 데 집중할 수 있도록 하기 때문에 중요한 차별화 요소입니다. 다음은 서버리스 운영 모델의 4가지 원칙입니다.

1. **서버 관리 없음** – 서버를 프로비저닝하거나 유지 관리할 필요가 없습니다. 설치, 유지 관리 또는 운영할 소프트웨어나 런타임이 없습니다.

2. **유연한 크기 조정** – 애플리케이션의 크기를 자동으로 조정하거나 개별 서버 단위가 아닌 소비 단위(예: 처리량, 메모리)로 용량을 조정하여 크기를 조정할 수 있습니다.

3. **사용한 가치에 따른 비용 지불** – 서버 단위가 아니라 일관된 처리량 또는 실행 시간과 같이 사용한 가치에 대해 비용을 지불합니다.

4. **자동화된 고가용성** – 서버리스는 가용성과 내결함성을 기본 속성으로 지니고 있습니다. 애플리케이션을 실행하는 서비스에서 기본적으로 이러한 기능을 제공하므로 이를 별도로 설계할 필요가 없습니다.

서버리스 운영 모델은 신속한 혁신을 원하는 고성장 기업에 적합합니다. 서버리스를 사용하면 팀이 더 빠르게 작업하고 비즈니스를 차별화하는 활동에 집중할 수 있으므로 혁신 플라이휠을 가속화할 수 있습니다.

AWS 기반 AWS Lambda 및 관리형 컨테이너 서비스 활용

컨테이너와 서버리스 컴퓨팅의 등장으로 인스턴스는 더 이상 유일한 클라우드 컴퓨팅 옵션이 아닙니다. 현대적 애플리케이션에 대한 최적의 컴퓨팅을 선택하는 것은 다음과 같은 몇 가지 질문으로부터 시작됩니다. 자체 관리 인프라로 비즈니스 결과를 개선할 수 있나요? 그것에 대한 전문 지식이 있나요? 추가 노력이 궁극적으로 가치를 창출하나요?

점점 더 많은 고객이 Amazon ECS 및 Amazon EKS와 같은 컨테이너 서비스 또는 AWS Lambda와 같은 이벤트 기반 서버리스 컴퓨팅 서비스를 도입하여 서버 관리 부담을 덜고 있습니다.

실제로 대부분의 고객은 이 두 가지를 조합하여 사용합니다. AWS 컨테이너 고객의 약 80%가 AWS Lambda도 도입했습니다.[2] 두 옵션을 모두 활용하면 AWS 인프라와 긴밀하게 통합되는 완전관리형 서비스, 광범위한 사용 사례 지원, 복잡성의 개념화, 광범위한 파트너 에코시스템 등의 이점을 얻을 수 있습니다.

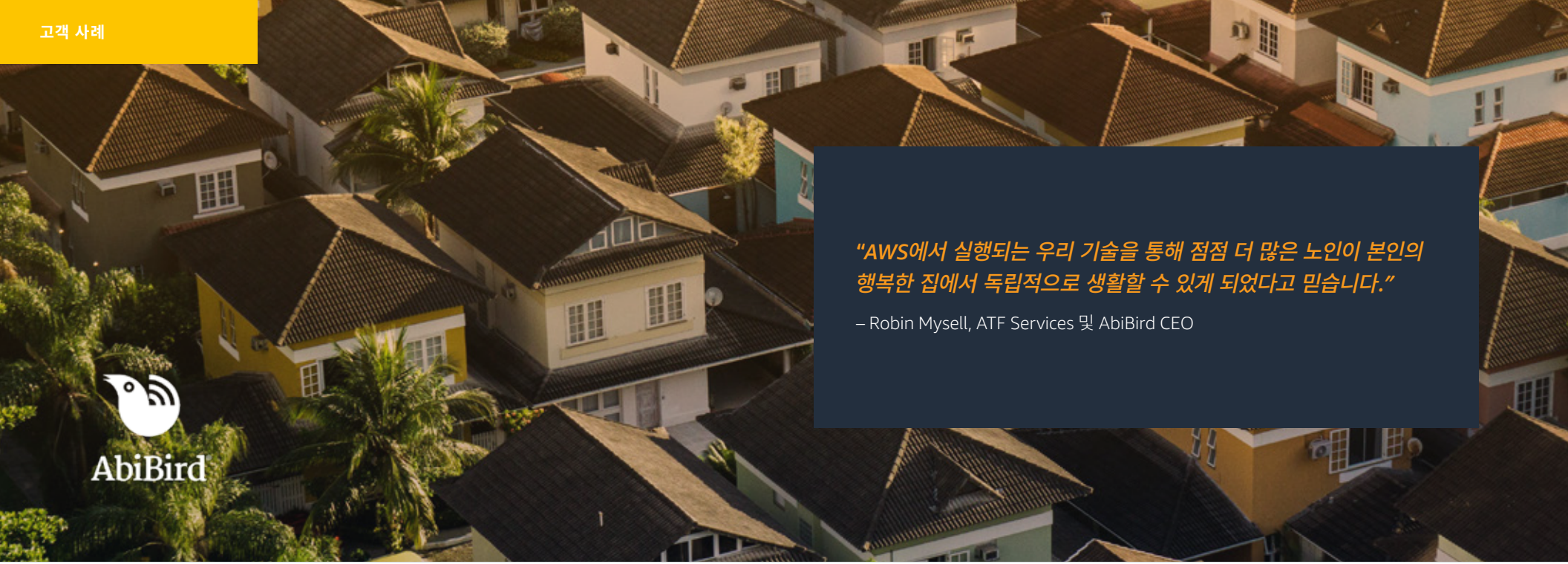
[2] *Datadog State of Serverless, 2020년*

그렇다면 어떻게 결정을 내리나요?

주로 코드 작성에 중점을 두고 기존 인스턴스 또는 컨테이너 플랫폼에 제한이 없는 경우 AWS Lambda를 선택합니다. AWS Lambda는 인프라에서 최대한의 개념화를 제공하여 고객이 가장 빠르게 릴리스할 수 있으므로 새로운 애플리케이션이 AWS Lambda에 매우 적합합니다.

기존에 컨테이너에 투자했거나 Kubernetes에 대한 오픈 소스 기본 설정 또는 인프라 관리나 구성에 대한 특정 요구 사항이 있는 경우는 컨테이너를 선택하는 경우가 많습니다. 컨테이너는 코드를 패키징하는 가장 보편적인 방법이며 레거시 애플리케이션을 현대화하기 위한 훌륭한 선택입니다.

	AWS의 관리 영역	고객의 관리 영역	
AWS Lambda 서버리스 함수	데이터 원본 통합 물리적 하드웨어, 소프트웨어, 네트워킹, 시설 프로비저닝	애플리케이션 코드	
AWS Fargate 서버리스 컨테이너	컨테이너 오케스트레이션, 프로비저닝 클러스터 크기 조정 물리적 하드웨어, 호스트 OS/커널, 네트워킹, 시설	애플리케이션 코드 데이터 원본 통합 보안 구성 및 업데이트, 네트워크 구성	관리 작업
Amazon ECS/ Amazon EKS 서비스로서의 컨테이너 관리	컨테이너 오케스트레이션 제어 영역 물리적 하드웨어, 네트워킹, 시설	애플리케이션 코드 데이터 원본 통합 보안 구성 및 업데이트, 네트워크 구성, 방화벽	작업 클러스터 관리 작업
Amazon EC2 서비스로서의 인프라	물리적 하드웨어, 네트워킹, 시설	애플리케이션 코드 데이터 원본 통합 확장 관리 작업	보안 구성 및 업데이트, 네트워크 구성 서버 프로비저닝, 서버 확장 및 패치 적용 관리



"AWS에서 실행되는 우리 기술을 통해 점점 더 많은 노인이 본인의 행복한 집에서 독립적으로 생활할 수 있게 되었다고 믿습니다."

– Robin Mysell, ATF Services 및 AbiBird CEO

AbiBird는 호주와 뉴질랜드에 60개의 지점과 350명의 직원 및 협력업체를 둔 호주 그룹인 ATF Services에서 전액 출자한 자회사입니다. AbiBird는 가정용 적외선 센서 및 스마트폰 기반 앱을 통해 노인 거주자의 활동을 모니터링할 수 있는 서비스를 제공합니다.

AbiBird는 Microsoft Azure에서 서비스를 실행하던 중, 서비스를 유지하기 위해 클라우드 공급업체에 너무 많은 지원 티켓을 제출했다는 사실을 깨닫게 되었습니다. 그리고 안정성과 확장성 개선이 필요했습니다. 이에 따라 AbiBird는 AWS로 이전했고 필요에 따라 컴퓨팅 서비스도 함께 사용하고 있습니다. 사용 편의성과 확장성은 물론 서비스의 관리형 특성 때문에 백엔드에서 AWS Lambda를 사용합니다.

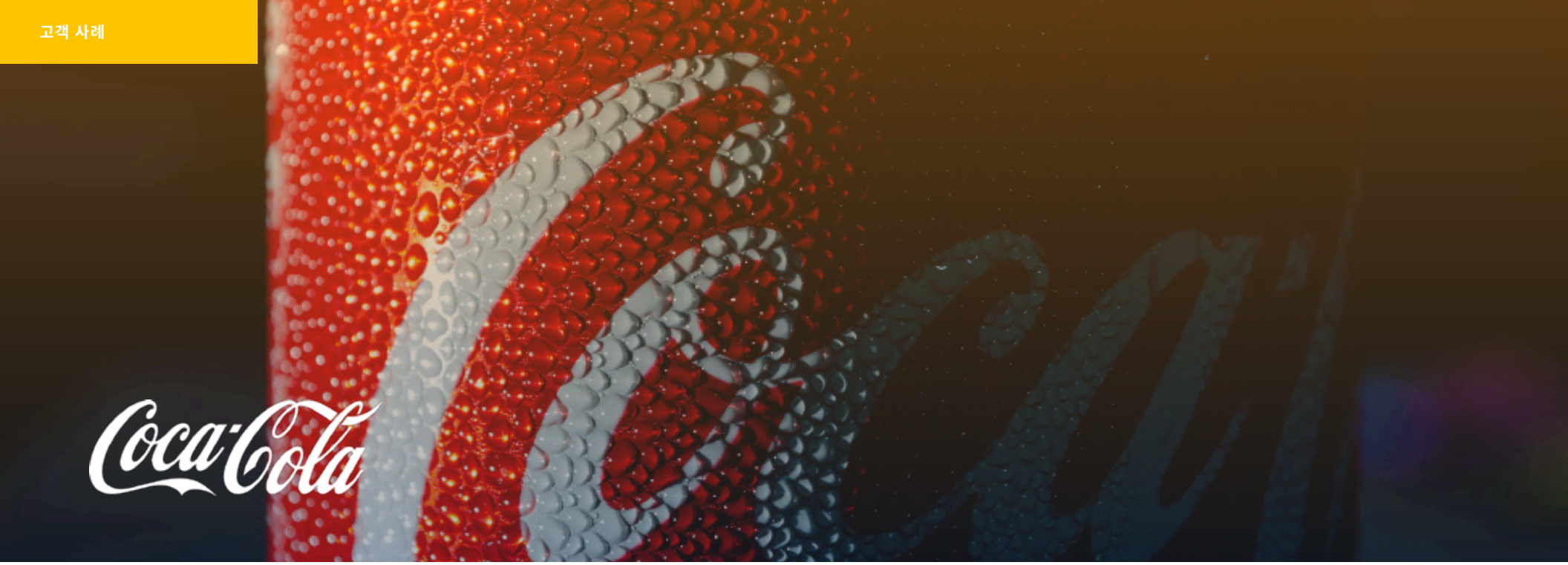
또한 AWS 기반 컨테이너 서비스를 사용하고, Amazon ECS에서 퍼블릭 API를 호스팅하며, AWS Fargate를 활용하여 가상 머신 플릿을 관리하는 번거로움 없이 컨테이너를 실행합니다.

2019년에 AWS에서 이 시스템을 시작한 후 단 한 건의 지원 티켓도 제출하지 않았습니다. 따라서 최소한의 오버헤드 지원으로 효율적으로 시스템을 실행할 수 있게 되었습니다.

전체 내용 보기 »

미국의 가장 상징적인 식품 브랜드 중 하나인 Taco Bell은 미국 전역에 7,000개 이상의 레스토랑을 보유하고 있습니다. 코로나19 팬데믹 상황에서 Taco Bell은 소비자의 배달 수요를 충족하기 위해 빠르게 전환해야 했습니다. 엔지니어링 및 분석 담당 부사장인 Vadim Parizher에 따르면 Taco Bell은 Amazon Web Services(AWS)에서 거의 모든 인프라를 실행하고 AWS의 서버리스를 사용하여 서버 관리보다는 비즈니스 로직 및 데이터 변환을 구축하여 실제 메뉴 및 레스토랑 정보를 해당 배달 업체에 제공하는 일에 더 집중합니다. "매우 복잡하고 여러 디지털 채널에 공유해야 하는 메뉴가 있습니다. 서버리스는 그 모델에 정말 잘 맞습니다."라고 Parizher는 말합니다. 서버리스 서비스를 사용함으로써 Taco Bell은 초기 비용을 절감하고, 작게 시작하여 필요한 만큼만 비용을 지불하고, 더 많은 서비스를 사용함에 따라 자동으로 확장할 수 있게 되었습니다.

동영상 보기 »



코로나19가 발생했을 때 소비 습관은 문자 그대로 하룻밤 사이에 바뀌었습니다. Coca-Cola는 혁신적인 Freestyle 음료 자판기에 터치리스 기능을 추가함으로써 신속하게 대응했습니다. Coca-Cola는 AWS Lambda를 활용하여 구축하기로 결정했으며 그 결과 팀은 보안, 지연 시간 또는 확장성보다 애플리케이션에 집중할 수 있었습니다. AWS Lambda에는 이 모든 것이 내장되어 있기 때문입니다. 새 애플리케이션은 단 100일 만에 출시되었으며 현재 52,000대 이상의 기계에 터치리스 기능이 탑재되었습니다.

전체 내용 보기 »

"짧은 지연 시간은 사용자 경험에 필수적이며, 이것이 우리가 AWS의 서버리스 솔루션에 전념하는 이유입니다."

– Michael Connor, Coca-Cola Freestyle Equipment Innovation Center 최고 아키텍트

현대적 Dev+Ops는 조직이 소프트웨어를 빠르고 안전하게 개발하고 프로덕션으로 릴리스하고 목표 가용성과 성능을 유지할 수 있도록 하는 문화적 철학, 관행 및 도구의 조합입니다.

AWS는 고성능 DevOps 조직을 도입하고 구축하기 위한 메커니즘을 제공하는 일반적이고 널리 받아들여지는 일련의 사례를 확인했습니다. 이 접근 방식은 지속적인 개선이라는 간단한 아이디어를 계획 및 코드 작성에서 배포 및 모니터링에 이르는 DevOps 수명 주기의 모든 과정에 적용합니다.

우리는 이 접근 방식을 현대적 Dev+Ops라고 부르며 규정 준수, 관찰 가능성, 탄력성 및 인프라와 같은 운영 작업을 개발 프로세스 초기에 공유하고 인공 지능 및 기계 학습(AI/ML)으로 이를 개선함으로써 개발자와 운영을 더 가깝게 만드는 데 중점을 둡니다.

개발자 민첩성: 개념화, 자동화 및 표준화

마이크로서비스 아키텍처를 사용하면 팀이 좀 더 민첩하고 빠르게 작업할 수 있으므로 더 많은 기능을 구축하고 출시할 수 있습니다. 그러나 구축 및 릴리스 프로세스가 팀의 작업 속도를 따라가지 못하면 고객에게 새로운 기능을 빠르게 제공할 수 없습니다. 기존 개발 프로세스 및 릴리스 파이프라인은 주로 수동 프로세스와 사용자 지정 코드 때문에 속도가 느렸습니다. 사용자 지정 코드는 오류가 발생할 수 있고 오랫동안 유지 관리를 해야 하므로 결국 장기적으로 부담이 됩니다. 코드 변경 및 구축 요청부터 테스트 및 배포에 이르는 수동 단계는 릴리스 속도에 가장 큰 걸림돌입니다. 이를 해결하기 위해서는 추상화, 자동화 및 표준화가 필요합니다.

개발 프로세스를 가속화하려면 프로덕션용 앱을 개발하고 제공하는 데 필요한 코드, 특히 비즈니스 로직이 아닌 코드를 최대한 추상화해야 합니다. 이를 위한 한 가지 방법은 리소스 프로비저닝 및 구성의 복잡성을 줄이는 프레임워크와 도구를 적용하는 것입니다. 그러면 개발자가 개발 프로세스 전반에서 보안, 프라이버시, 안정성, 성능, 관찰성 및 확장성을 위한 모범 사례를 적용하면서 동시에 신속하게 작업할 수 있습니다. 개발 프레임워크를 사용하면 아키텍처가 장기적인 비즈니스 성장을 지원할 것이라는 확신을 가질 수 있습니다.

모범 사례 템플릿을 사용하여 소프트웨어 전달 프로세스를 정의하면 클라우드 환경에서 모든 인프라 리소스의 모델링 및 프로비저닝을 위한 표준을 제공할 수 있습니다. 이러한 '코드형 인프라' 템플릿은 수동 프로세스를 사용하는 대신 코드를 통해 애플리케이션의 전체 기술 스택을 프로비저닝하므로 팀이 올바르게 시작할 수 있도록 도움을 줍니다.

자동화를 통해 반복 가능한 동작을 만들어 소프트웨어 제공 수명 주기를 가속화할 수 있습니다. 지속적 통합 및 지속적 전달(CI/CD)을 통해 릴리스 파이프라인을 자동화하면 팀이 고품질 코드를 더 빠르게 더 자주 릴리스할 수 있습니다. CI/CD를 활용하는 팀은 많은 코드를 더 빠르게 제공할 수 있고, 문제 발생 시 더 신속하게 대응할 수 있습니다. Puppet 2020 State of DevOps Report에 따르면 이러한 방식을 구현하는 팀은 실패율이 5배 더 낮고, 배포 커밋 속도가 440배 더 빠르며, 배포 빈도가 46배 더 높습니다.[3] 특히 CI/CD를 구현하는 팀은 프로세스 및 도구 관리보다 새로운 기능 및 코드 생성에 44% 더 많은 시간을 사용합니다.

CI/CD 파이프라인은 현대적 애플리케이션을 구축하는 새로운 기반이 되었습니다. Amazon에서는 릴리스 속도를 높이기 위해 CI/CD를 사용하기 시작했고 그 결과는 놀라웠습니다. 연간 수백만 건의 배포를 달성했으며 매년 더 빠르게 성장하고 있습니다. AWS에서는 AWS의 경험을 고객이 활용할 수 있도록 내부적으로 사용하는 도구를 기반으로 개발자 도구 제품군을 구축했습니다. 따라서 고객은 이를 활용해 코드를 더 빠르게 제공할 수 있습니다.

추가 정보

지속적 통합(CI)은 개발자가 코드 변경 사항을 정기적으로 중앙 리포지토리에 병합하면, 나중에 자동 구축 및 테스트가 실행되는 소프트웨어 개발 방식입니다. 지속적 통합은 대부분 소프트웨어 릴리스 프로세스의 구축 또는 통합 단계를 말하며, 자동화 요소(예: CI 또는 구축 서비스) 및 문화적 요소(예: 잦은 통합)를 모두 포함합니다.

지속적 전달(CD)은 프로덕션에 릴리스할 수 있도록 코드 변경이 자동으로 준비되는 소프트웨어 개발 방식입니다. 지속적 전달은 구축 단계 후에 모든 코드 변경 사항을 테스트 환경 및/또는 프로덕션 환경에 배포하여 지속적 통합을 확장합니다.

Amazon에서 안전한 핸드오프 배포를 자동화하는 방법 알아보기 »

[3] *Puppet 2020 State of Devops Report, 2020년*

Lululemon Athletica는 AWS를 통해 며칠이 아니라 몇 분 만에 개발 환경을 가동하고, 환경을 자동화하고, 지속적 통합 및 배포를 지원할 수 있게 되었습니다. 이 캐나다 회사는 요가에서 영감을 받아 제작한 의류 및 기타 다른 의류를 전 세계 350개 이상의 지역에서 판매합니다. Lululemon은 AWS 클라우드에서 개발 및 테스트 환경과 곧 출시될 모바일 앱을 실행합니다.

Lululemon은 AWS CloudFormation 템플릿 및 AWS CodePipeline을 사용하여 새로운 프로덕션 계정을 구축하는 시간을 2일에서 몇 분으로 단축했습니다. 이러한 민첩성 향상 덕분에, 이제 Lululemon 개발 팀은 현재 리소스가 할당된 솔루션에 안주하는 대신 실험을 통해 최상의 솔루션을 확보할 수 있게 되었습니다.

전체 내용 보기 »

"모든 지속적 통합 및 배포 파이프라인은 자동화되고, 관리가 쉽고, 탐색 가능해야 합니다. 이것이 바로 우리가 AWS를 사용하면서 얻는 이점입니다. 이전의 온프레미스 환경에서는 도저히 이룰 수 없었던 수준의 간편성과 투명성을 얻었습니다."

– Sam Keen, Lululemon 제품 아키텍처 담당 이사

HyperTrack은 앱을 통해 실시간으로 위치를 추적하는 셀프 서비스 클라우드 플랫폼입니다. 2015년 말에 출시되었을 때 HyperTrack은 개발자가 새로운 기능을 구축하는 데 필요한 시간을 확보하면서, 앞으로 예상되는 성장에 맞춰 자동으로 확장되는 플랫폼을 구축해야 했습니다.

HyperTrack은 엔지니어의 개입 없이 자동으로 확장하고 축소할 수 있도록 모바일 개발 프레임워크 및 서버리스 아키텍처에 AWS Amplify를 사용하기로 결정했습니다.

그 결과, 서버리스로 전환하기 전에 사용하던 아키텍처 대비 30%의 비용 절감 효과를 달성했습니다. 서버 관리에 집중하는 운영 리소스가 필요 없어짐에 따라 상당한 비용이 절감되었습니다. HyperTrack은 수백만 건의 이벤트를 관리하는 동시에 매주 40시간의 작업 시간을 절약하게 되었습니다.

전체 내용 보기 »

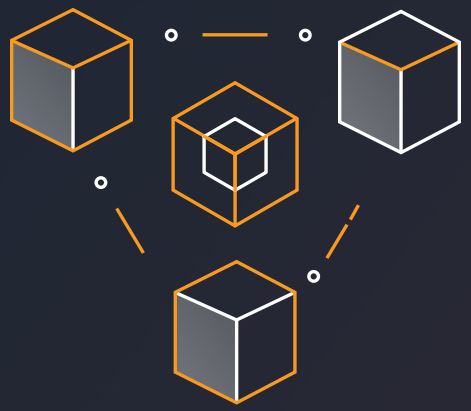
주인 의식 문화 조성: 현대적 Dev+Ops로 효율적인 관리 및 더 많은 혁신

혁신은 궁극적으로 사람으로부터 비롯됩니다. 따라서 직원들이 더 나은 고객 성과를 내도록 지원하는 것이야말로 현대적 애플리케이션 개발의 시작입니다. AWS에서는 '프로젝트가 아닌 제품'의 개념을 사용하여 주인 의식이 팀 구조에 어떤 영향을 미치는지 설명합니다. 간단히 말해서 제품을 만드는 팀에 제품을 실행하고 유지 관리하는 책임도 있다는 것입니다. 따라서 제품 팀은 제품의 일부가 아니라 전체를 개발하는 책임을 지게 됩니다.

AWS는 고도로 확장 가능한 웹 애플리케이션인 Amazon.com을 10년 이상 구축 및 실행하면서 팀에 자율성을 부여하는 것이 얼마나 중요한지 몸소 체득했습니다. 고객 의견 수렴, 로드맵 계획, 애플리케이션 개발 및 운영을 비롯하여 전체 애플리케이션 수명 주기에 대한 전면적인 권한을 팀에 부여했을 때, 팀에 주인 의식이 생겼고 새로운 고객 성과를 개발하고 제공할 권한이 있다는 것을 스스로 느끼게 되었습니다. 자율성은 동기를 부여하고, 창의성의 문을 열며, 신뢰가 바탕이 된 환경에서 위험을 감수할 수 있는 문화를 조성합니다.

주인 의식 문화를 수용하는 것이 본질적으로 기술적인 것은 아니지만, 현대적 애플리케이션 개발에서 가장 어려운 부분 중 하나입니다. 팀이 제품에 대한 주인 의식을 가질 수 있도록 권한을 부여하려면 조직의 사고방식, 팀의 구조 및 담당 업무를 바꾸어야 합니다.

대부분의 조직에서 IT는 두 가지 진영 중 하나에 속합니다. IT를 전략적 경쟁 무기로 간주하거나 보다 일반적으로 비즈니스 성장을 지원하는 데 필요한 비용 센터로 간주합니다.



혁신 문화 구축

1. **고객으로부터 시작 –** 모든 혁신은 고객의 요구로부터 시작하며 궁극적으로 고객을 만족시켜야 합니다. 고객의 요구에 집중하기 위해 끊임없이 우선순위를 정합니다.

2. **구축 담당 직원 채용 –** 고객을 위한 제품과 기능의 구축 및 릴리스 속도를 저해하는 모든 장애물을 제거합니다. 반복 속도가 빠를수록 플라이휠 회전 속도도 빨라집니다.

3. **신뢰성 높은 시스템으로 구축 담당 직원 지원 –** 말뿐인 혁신이 아니라, 경영진부터 영업, 지원에 이르기까지 비즈니스의 모든 분야에서 혁신에 전념합니다.

효율적인 관리, 더 많은 혁신

현대적 애플리케이션은 빠른 혁신을 가능하게 함으로써 경쟁력의 차별화를 만듭니다. 속도와 민첩성을 강조하는 서비스, 사례 및 전략을 도입함으로써 리소스를 일반적인 비즈니스 업무에서 깊은 고객 가치를 지닌 차별화된 활동으로 전환할 수 있습니다. 더 많은 실험을 수행하고 더 빠르게 아이디어를 제품으로 출시할 수 있습니다. 구축에 더 많은 시간을 들이고 관리에는 더 적은 시간을 할애할 수 있는 환경을 조성할 수 있습니다. 현대적 애플리케이션은 Amazon을 비롯한 많은 기업에서 빠르고 민첩하게 혁신하는 방법입니다.

AWS 기반의 현대적 애플리케이션을 구축해야 하는 이유

출시 기간 단축	개발자는 구축 및 릴리스 주기를 단축하고 운영 오버헤드를 줄임으로써 새로운 기능을 신속하게 구축할 수 있습니다. 자동화된 테스트 및 릴리스 프로세스를 통해 오류 발생률이 감소하므로 제품 출시 준비 시간이 단축됩니다.	**실제 사례 보기:** **Urbanbase, AWS로 20배 더 빠른 서비스 출시**
혁신 증대	모듈식 아키텍처에서는 개별 애플리케이션 구성 요소를 신속하게 변경할 수 있으며 전체 애플리케이션에 큰 위험 부담을 주지 않으므로 개발 팀은 새로운 아이디어를 더 자주 실험해 볼 수 있습니다.	**실제 사례 보기:** **iRobot, AWS Lambda 및 AWS IoT 플랫폼을 사용하여 Roomba 로봇 진공 청소기 관리**
안정성 개선	테스트 절차를 자동화하고 개발 수명 주기의 모든 단계를 모니터링함으로써 현대적 애플리케이션을 안정적으로 배포할 수 있습니다. 모든 문제를 실시간으로 평가하고 처리할 수 있습니다.	**실제 사례 보기:** **Siemens, 고객 제어 시스템 경보를 90% 줄이고 인프라 비용을 85% 절감**
TCO 개선	현대적 애플리케이션은 가치 기반 요금 모델을 통해 오버프로비저닝으로 인한 비용과 유휴 리소스에 지불하는 비용을 줄입니다. 인프라 관리를 오프로드함으로써 유지 관리 비용 또한 절감됩니다.	**실제 사례 보기:** **AWS Lambda로 앱 유지 관리 비용 최대 80% 절감**

애플리케이션 현대화 여정 시작

관리형 컨테이너 서비스로 리플랫포밍

클라우드에서 실행되는 모든 컨테이너화된 애플리케이션의 80%가 AWS에서 실행.*
클라우드에서 실행되는 모든 Kubernetes 워크로드의 84%가 AWS에서 실행*

리소스
Amazon ECS 워크숍Amazon EKS 워크숍AWS AppRunner 워크숍

추천 교육 과정(강의식 교육)
Running Containers on Amazon ElasticKubernetes Service(Amazon EKS)

추천 교육 과정(온라인)
Amazon Elastic Container Service(ECS) Primer

서버리스 기술 및 도구로 새로운 현대적 애플리케이션 구축

현대적 애플리케이션 구축을 위한 서버리스 우선 전략 도입 시 유지 관리 시간의 최대 80%, 개발 시간의 약 70% 절감**

리소스
Innovator Island - 서버리스 웹 애플리케이션 개발 워크숍. 서버리스 웹 앱 동영상 튜토리얼을 구축합니다.

추천 교육 과정(강의식 교육)
Advanced Developing on AWS

추천 교육 과정(온라인)
Architecting Serverless Solutions

현대적 Dev+Ops 모델로 변환

개선된 DevOps 방식을 채택한 팀의 60%가 1일 이내에 보안 취약점 완전 수정#

리소스
Amazon Builder's LibraryAWS DevOps 서비스

추천 교육 과정(강의식 교육)
DevOps Engineering on AWS

추천 교육 과정(온라인)
Getting started with DevOps on AWS

AWS 기반 현대적 애플리케이션 구축에 대해 자세히 알아보기 →

전문가에게 현대적 애플리케이션 개발의 모범 사례 구현 방법 들어보기 →

AWS 파트너와 함께 현대화 프로젝트 가속화하기 →

aws modern apps

* 출처: Nucleus Research
** 출처: Deloitte
# 출처: 2020 DORA state of DevOps Report