이벤트 중심 아키텍처를 구현하는 방법에 관한 설계자 가이드

엔터프라이즈에서 이벤트 활용을 시작하기 위한 6단계

Sumeet Puri
최고 기술솔루션 책임자

solace.

“

비즈니스가 실시간을 추구하는 경향이 증가하고 고객 경험에 더욱 중점을 두면서 이러한 수요를 충족하기 위해 애플리케이션 아키텍처를 업그레이드해야 하며, 이벤트 중심 아키텍처가 대세로 떠올랐습니다."

Sumeet Puri
최고 기술솔루션 책임자

목차

서론

대다수 회사의 비즈니스 지원 애플리케이션은 온프레미스 데이터 센터, 현장 사무실, 제조 현장, 매장 등 다양한 환경에서 실행됩니다. 이와 같은 여러 환경간 호환성 및 연결성 부족으로 인해 비즈니스 애플리케이션은 긴밀하게 결합된 동기식 요청/응답 상호작용, 개별적으로 맞춤 설정된 배치 지향의 ETL 프로세스 또는 심지어 맞춤형 통합을 통해 상호작용할 수밖에 없습니다. 이는 다시 느린 응답 속도와 오래된 데이터로 이어집니다. 긴밀한 결합은 끊임없이 변화하는 비즈니스 요구와 실시간 수요에 대한 민첩한 대응을 어렵게 합니다.

실시간을 추구하는 엔터프라이즈는 이벤트를 애플리케이션과 이러한 이벤트를 처리하는 마이크로서비스 간 스트리밍하여 빠르게 인사이트를 확보하고 의사결정을 내려야 합니다. 응답성을 향상하고 클라우드, IoT, 마이크로서비스 등의 최첨단 기술을 활용하려면 엔터프라이즈 아키텍처가 실시간, 이벤트 중심 상호작용을 지원해야 합니다.

기본적으로 모든 비즈니스 프로세스는 일련의 이벤트입니다. '이벤트'는 광범위하게는 변화 통지로 볼 수 있습니다. 이러한 변화의 형태는 다양할 수 있지만, 공통적으로 개체에서 발생한 행동이라는 구조를 지니고 있습니다. 이벤트 중심 아키텍처는 느슨하게 결합된 애플리케이션과 마이크로서비스가 이러한 이벤트를 생성하고 소비하도록 하는 기업용 IT 시스템을 구축하는 방법일 뿐입니다.

이벤트 중심 아키텍처 구현은 다른 모든 여정과 마찬가지로 하나의 단계로 시작합니다. 이 여정을 떠나려면 물론 보유한 데이터에 대한 이해도 높아야 하지만, 더 중요한 것은 이벤트 중심의 사고방식을 갖추는 것입니다.

최고 기술 및 솔루션 책임자 Sumeet Puri가 엔터프라이즈에서 적절한 도구를 사용하여 이벤트 중심의 아키텍처를 구현하고 어려움을 해결할 수 있는 확실한 전략을 설명합니다. 실무에서 입증된 Sumeet의 6단계 프로세스를 실행하여 주요 이해관계자의 참여를 이끌어내고 조직 전체를 혁신하세요.

이벤트 중심의 사고방식

이벤트 중심 아키텍처는 새로운 개념이 아닙니다. GUI와 자본시장 거래 플랫폼은 이전부터 이벤트 중심으로 구축되어왔습니다. 다만 이제서야 주류로 떠올랐을 뿐이죠. 이는 실시간 수요를 충족하려면 서비스 지향 아키텍처(SOA), 추출, 혁신 및 로드(ETL) 방식과 배치 기반 방식이 변화해야 하기 때문입니다.

하지만 엔터프라이즈 아키텍처를 더욱 반응성이 뛰어나고 민첩한 실시간 아키텍처로 혁신하는 여정을 시작하기 전에 먼저 비즈니스에서 디지털 이벤트로 발생하는 모든 사항을 고려하고, 이러한 디지털 이벤트를 IT 인프라의 핵심 요소로 간주해야 합니다.

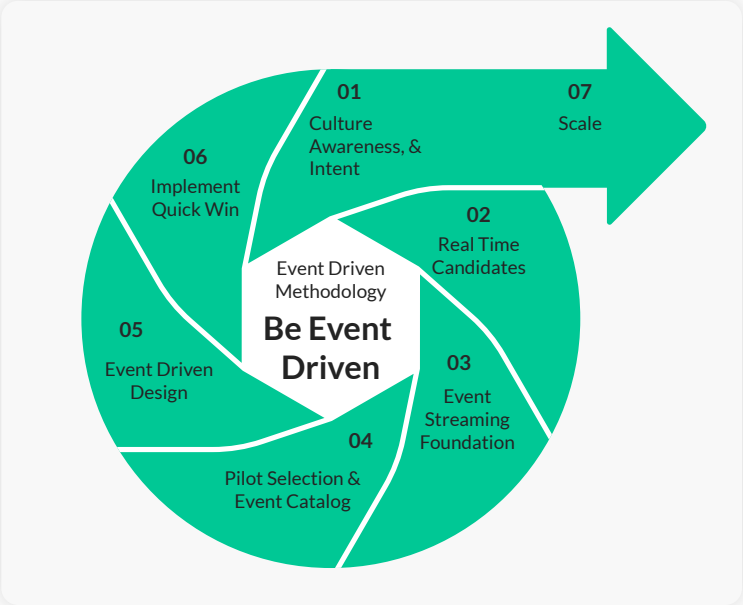
이벤트 중심 아키텍처에서 애플리케이션과 마이크로서비스는 이벤트 또는 이벤트 어댑터를 통해 소통합니다. 이러한 애플리케이션에서 이벤트는 다양한 주제에 대한 관심사를 나타내는 구독에 따라 게시/구독 방식으로 연결됩니다.
즉, 비즈니스 흐름은 배치 프로세스나 흐름을 조정하는 엔터프라이즈 서비스 버스(ESB)와는 달리 비즈니스 로직 구성 요소의 관심사와 역량에 기반하여 유통됩니다. 따라서 새로운 애플리케이션을 추가하기가 훨씬 더 쉬운데, 애플리케이션이 다른 시스템에 영향을 주지 않고 이벤트 스트림에 진입하여 기능을 수행하고 가치를 더할 수 있기 때문입니다.

그렇다면 이를 수행하는 방법은 무엇일까요? 어떤 전략을 실천해야 할까요?

광범위하게는 조직을 이벤트 중심 엔터프라이즈로 전환하고 싶다면 다음 3가지를 수행해야 합니다.

1. 기존 시스템에서 이벤트 활용
2. 플랫폼을 최신 상태로 업데이트하여 여러 환경에 걸친 스트리밍 지원
3. 내외부 이해관계자에게 관련 내용 통지 및 설명

다음으로는 이러한 단계를 어떻게 실천할지 고민해보아야 하는데요, 다음 6단계는 위의 전략을 활용하여 실제 사례에서 이벤트 중심 아키텍처로의 여정을 가속화하고 더욱 원활하게 하며 위험을 감소시킨다는 사실이 입증되었습니다.



solace.

1단계: 문화, 인지도와 의도

이벤트 중심 아키텍처를 구현할 준비가 되었고 이를 명확하게 깨달았으며 실제로 원하시나요?

이 글을 읽고 계신 분들이라면 분명히 그렇다고 대답하실 겁니다. 하지만 대다수의 IT 전문가들은 학교에서부터 점진적으로 사고하도록 배웠습니다. 저처럼 Fortran, C, Java나 Node에서 시작한 사람들은 대부분 동기식 함수 호출, RPC 호출, 모두 동기식인 웹 서비스와 API에 대한 IT 교육을 받고 이에 관한 경험을 쌓았을 뿐입니다. 단, 예외는 있습니다. 자본시장 프론트 오피스 시스템은 항상 실시간 대응이 필수적이었기 때문에 이전부터 이벤트 중심으로 설계되었죠. 반면 IT 부문은 이러한 측면에서 대대적인 변화를 필요로 합니다.

마이크로서비스는 서로를 호출하여 분산되고 고립된 요소들을 만들 필요가 없습니다. 설계자는 어떤 애플리케이션/마이크로서비스가 어떤 서비스를 소비하고 생산하는지 알기만 하면 ESB로 조정할 필요 없이 발행/구독 이벤트 브로커나 (이벤트 메시라고도 하는) 분산된 브로커 네트워크를 사용하여 유통할 수 있습니다.

EDA의 이점인 민첩성, 반응성, 더 나은 고객 경험에 대해 숙고해 보고, 관련 자료를 읽고, 이해관계자에게 이러한 이점을 알리는 것이 중요합니다. 지지를 구축하고 전략적으로 접근하고 혁신, 마이크로서비스, API 등 다음 프로젝트를 이벤트 중심으로 진행하세요.

대상이 될 만한 사용 사례를 고민해보고, 이벤트 중심의 관점에서 이러한 사례를 살펴보고, 이점을 인식시킬 수 있는 최적의 접근법을 명확하게
기술해 보세요.

실시간 중심으로 생각하세요. 이벤트 중심으로 생각하세요.

2단계: 실시간 후보 식별

모든 시스템이 변화를 필요로 하거나 실시간으로 전환 가능한 것은 아니지만, 대다수는 이벤트 중심 접근법의 이점을 누릴 수 있습니다. 아마 여러분도 실시간으로 전환하면 이득을 얻을 수 있는 여러 프로젝트, API와 후보를 떠올릴 수 있으실 겁니다.

여러분의 엔터프라이즈에서 실시간으로 전환할 만한 요소가 있나요? 번거로운 주문 관리 시스템이나 현재 구축 중인 차세대 결제 플랫폼을 실시간으로 전환하면 어떨까요? 가격 업데이트나 PLM 지침 변경과 같은 마스터 데이터를 폴링(polling)하기보다는 다운스트림 애플리케이션으로 푸시하면 어떨까요? 탑승권을 스캔하면 실시간 항공사 멤버십 포인트로 업그레이드를 받을 수 있게 된다면 어떨까요? 혹은 실시간 공항 지상 운영 최적화는 어떤가요?

각 프로젝트마다 우선순위와 과제는 다를 수 있으므로, 다음에 해당하는 프로젝트를 찾으세요.

- 불편(예: 불안정함, 성능 문제, 새로운 기능)을 제거함
- 비즈니스에 중간~높은 수준의 영향을 미침
- 신속한 성과 도출을 통해 비즈니스 가치를 선사하며 팀의 긍정적인 사기를 북돋아줌

우선, 소규모 프로젝트 하나로 시작하여 여러분이 소속된 조직의 고유한 특징을 찾으세요. 이러한 특징은 보다 큰 규모의 프로젝트에서도 반드시 나타나기 마련입니다. 모든 데이터 소스가 이벤트 중심 아키텍처의 적용 대상은 아니라는 사실에 유의하세요. 강력한 데이터 생성 기능을 지녔으며 메시지를 제출할 수 있도록 간편하게 변경할 수 있는 시스템을 찾으세요. 이러한 메시지는 이후 이벤트로 활용됩니다.

이벤트 중심 여정을 위해 실시간으로 전환할 대상 요소 후보를 선정하세요. 이벤트는 보통 비즈니스 중심이기 때문에 모든 이해관계자가 기술 관련 종사자는 아닙니다. 따라서 초기 단계부터 이해관계자를 참여시키는 것이 중요합니다. 혁신으로 인해 영향을 받을 팀은 어떤 팀인지, 프로젝트를 모든 관점에서 성공으로 이끌기 위해서는 어떤 사람들을 참여시키고 설득시켜야 할지를 고민해 보세요.

3단계: 이벤트 기반 구축

다음으로 고려할 사항은 도구입니다. 첫 프로젝트를 정했다면 그다음으로는 아키텍처와 도구에 대해 생각해 보아야 합니다. 이벤트 중심 아키텍처를 구현하려면 흐름을 마이크로서비스로 분해하고, 마이크로서비스가 일대다의 분산된 발행/구독 방식으로 상호 소통하도록 하는 런타임 패브릭을 마련해야 합니다.

또한 적시에 설계를 시작하고, 이벤트를 설명 및 분류하며 이벤트 간 관계를 시각화할 수 있는 도구도 구비해야 합니다.

비즈니스의 모든 사용 사례를 충족할 정도로 가변적이고, 비즈니스 데이터가 배포된 모든 장소(온프레미스, 비공개 클라우드, 공용 클라우드 등)에서 데이터를 수신할 수 있을 정도로 유연한 아키텍처를 구축해야 합니다. 이때 필요한 것이 바로 이벤팅 플랫폼과 이벤트 메시 런타임입니다.
이벤팅 플랫폼은 가장 기본적인 요소만 구축하더라도 첫 프로젝트에서 마이크로서비스로 활용할 수 있으며, 그러면 이벤트가 온라인에 발행되어 상호 통신이 가능해지므로 REST를 통해 발행되어 분산되고 서로 고립된 형태로 존재하는 일을 방지할 수 있습니다.

다음의 런타임 및 설계-시간 요소는 필수입니다.

- Event Broker
- Event Mesh
- Event Portal
- 이벤트 분류

Event Broker

이벤트 브로커는 발행/구독, 저지연 및 게재가 보장되는 이벤트 라우팅의 기본 런타임 구성 요소입니다. 애플리케이션과 마이크로서비스는 커플링이 해제되어 이벤트 브로커를 통해 통신합니다. 하나 이상의 애플리케이션이나 마이크로서비스 또는 분석 엔진이나 데이터

레이크에 의해 구독된 이벤트는 주제 서열(또는 분류)에 따라 주제를 통해 이벤트 브로커로 발행됩니다.

이상적인 이벤트 브로커는 고객이 특정 벤더에 종속되는 상황을 방지하기 위해 오픈 프로토콜과 API를 사용합니다. 개방형 표준 하에서는 적합한 이벤트 브로커 제공자를 천천히 선택할 수 있습니다. TCP/IP가 고객에게 네트워킹 장비를 자유롭게 선택할 수 있는 자유를 준 것에서 알 수 있듯이, 개방형 표준은 인터넷을 탄생시켰다고 해도 과언이 아닙니다. 오픈 소스 커뮤니티를 활용하면 즉각적인 변경 사항을 수행하기가 더욱 용이해지며, 폐쇄적인 문서를 참조하거나 긴 시간 동안 지원을 기다리지 않아도 됩니다.

마지막으로, 이상적인 이벤트 브로커는 단순합니다. 이상적인 이벤트 브로커는 간단한 배포, 이벤트 커버넌스와 확장성을 제공하여 가장 중요한 사항인 이벤트에 집중할 수 있게 해 줍니다.

Event Mesh

처음에는 한곳에서 하나의 이벤트 브로커로 시작하더라도, 대부분의 최첨단 애플리케이션은 분산되어 있습니다. 애플리케이션, 마이크로서비스와 인사이트 역량은 온프레미스, 여러 개의 클라우드, 공장, 지사와 판매점 등 여러 위치에 분산되어 있기 마련입니다.

판매점에서 발생한 이벤트는 지역 매장의 시스템과 중앙화된 ERP를 거쳐 클라우드 기반의 데이터 레이크로 전송되었다가 다시 외부 파트너에게로 전송되어야 할 것 입니다. 따라서 이벤트 분산은 생산자와 소비자에게 투명해야 합니다. 마치 사람들이 호스팅 서버와 관계없이 모든 웹사이트에 액세스하기 위해 가정의 Wi-Fi에 연결하는 것처럼, 이벤트는 로컬 이벤트 브로커에 연결되어야 합니다.
이벤트 중심 아키텍처는 지속적으로 진화하는 시스템이므로, 오늘 온프레미스에서 파악한 이벤트 소스의 이벤트를 내일이면 클라우드에서 찾을 수 있습니다.

이벤트 메시는 온프레미스, 클라우드, IoT 엣지 등의 이벤트 등 배포된 곳이 어디든지 애플리케이션 간 동적으로 이벤트를 라우팅하는 이벤트 브로커 네트워크입니다. 라우팅 테이블을 수렴하여 인터넷을 구현하는 라우터처럼, 이벤트 메시는 이벤트 스트리밍을 위해 다양한 최적화로 주제 구독을 수렴합니다.

이벤트 메시는 다음과 같이 다양한 방법으로 애플리케이션 아키텍처를 지원합니다.

- 구독/발행, 주제 필터링과 분산된 네트워크 전체로의 전달 보장을 통해 마이크로서비스를 연결하고 유통합니다.
- 온프레미스의 이벤트를 클라우드 서비스와 애플리케이션으로 푸시합니다.
- IoT의 디지털 혁신을 가능하게 합니다.
- 여러 사업부에 걸쳐 DaaS(서비스로서의 데이터)를 제공하여 인사이트, 분석, ML 등을 가능하게 합니다.
- 반응성이 훨씬 더 뛰어난 이벤트 집계 방식을 지원합니다.

물론 이벤트 메시 배포 방식에 따라 적절한 이벤트 브로커의 종류도 달라집니다.

Event Portal

이벤트 포털은 API 포털과 마찬가지로 이벤트 메시로의 디자인이자 런타임 뷰입니다. 이벤트 포털은 설계자가 거버넌스 하에 이벤트를 정의하고 설계할 수 있는 간편한 GUI 기반 도구를 제공하며, 퍼블리셔와 구독자도 이러한 도구를 사용할 수 있도록 합니다. 이벤트를 정의한 후에는 이벤트 포털에서 마이크로서비스나 애플리케이션을 설계하고, 이벤트 카탈로그를 살펴보고 검색하여 이러한 마이크로서비스나 애플리케이션이 소비할 이벤트를 선택할 수 있습니다.

정의된 이벤트는 발견될 수 있도록 이벤트 카탈로그에 등록됩니다. 이벤트는 이벤트 카탈로그를 통해 확인할 수 있습니다. 이 단계는 비록 문서화 단계에 가깝지만, 처리할 수 있는 이벤트를 시각화하고 설명하는 데 도움이 됩니다. 카탈로그는 더 많은 이벤트 소스를 확보하고 더욱 다양한 방법으로 이러한 소스를 소비할 수 있도록 시스템을 구축할 때 이미 구축된 요소를 확인하여 중복된 노력을 방지할 수 있는 좋은 참조로 기능합니다.

이벤트 분류

주제 라우팅은 이벤트 중심 아키텍처의 핵심입니다. 주제는 이벤트 메타데이터로서, HTTP URL 또는 파일 경로와 마찬가지로 이벤트를 설명하는 a/b/c 형태의 태그입니다. 이벤트 브로커는 주제를 이해하고 이벤트를 구독한 사람에 기반하여 이러한 이벤트를 라우팅할 수 있습니다. 여기에는 와일드카드 구독이 포함됩니다.

이벤트 중심 아키텍처를 향한 여정을 시작할 때는 주제 분류에 유의해야 합니다. 초기부터 주제 명명 규칙을 정하고 관리하세요. 적절한 분류는 여러분이 투자할 수 있는 가장 중요한 설계 요소이므로 서두르지 말고 신중하게 설정하세요. 효과적인 분류는 이벤트 라우팅에 크게 도움이 되며, 애플리케이션 개발자가 명명 규칙을 명확하게 이해할 수 있게 됩니다. 주제 서열에 관한 활용 팁은 bit.ly/topic-hierarchy를 참조하세요.

Udders Ice Cream 예시: 이벤트 분류

Udders Ice Cream의 주문 예시를 살펴보겠습니다. 싱가포르의 전자상거래 웹사이트 Lazada를 통해 럼 앤 레이즌 맛 아이스크림 주문이 들어왔습니다.

Event	주제
신규 주문	order/init/1.1/icecream/udders/rumraisin/sg/lazada
확인된 주문	order/valid/1.1/icecream/udders/rumraisin/sg/lazada
주문 배송	order/shipped/1.1/icecream/udders/rumraisin/sg/lazada

퍼블리셔(마이크로서비스, 애플리케이션, 레거시-투-이벤트 어댑터)의 이벤트에는 주제가 메타데이터로 존재하므로, 소비자는 이를 사용하여 이벤트를 구독할 수 있습니다.

Event	퍼블리셔	구독자
신규 주문	Lazada 전자상거래 스토어 사이트	주문 확인 마이크로서비스
확인된 주문	주문 확인 마이크로서비스	싱가포르의 아이스크림 주문 처리자
주문 배송	모든 업스트림 퍼블리셔	데이터 레이크 또는 AI/ML 인제스터 (ingestor)

주제 구독은 다음과 같습니다.

- 1.1 버전의 모든 주문을 소비 및 확인하고 주문 유효 메시지 발행: **order/init/1.1/>**
- 싱가포르에서 접수된 모든 유효한 주문 소비 및 처리: **order/valid/*/icecream/*/*/sg/***
- 모든 단계, 카테고리, 위치의 주문: **order/>**

조직 정리

소규모로 시작한 이벤트 중심 아키텍처가 점점 성장할수록 조직 또한 이벤트 중심의 사고로 전환해야 합니다. 이를 위해서는 선구안을 지닌 리더들이 조직 전체의 동의를 받아야 합니다. ESB 팀은 조정보다는 유통을 고려하기 시작해야 하며, API 팀은 단순한 요청/응답이 아닌 이벤트 중심 API를 고려해야 합니다.

4단계: 시범 애플리케이션 선택

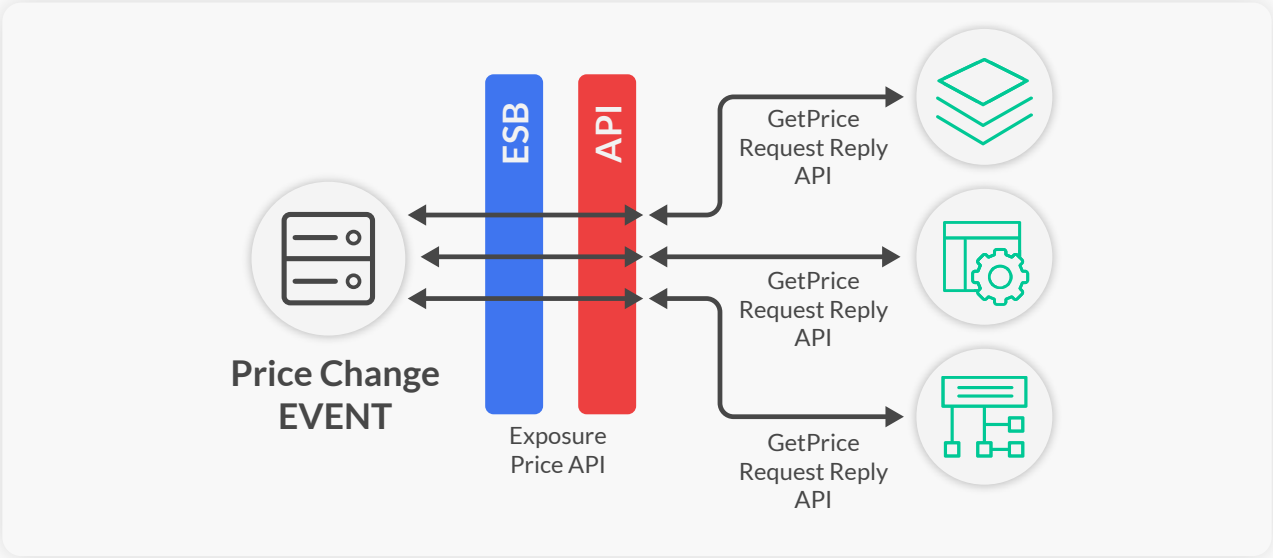
이벤트 중심 아키텍처 구현을 위한 다음 단계는 어떤 이벤트 흐름을 가장 먼저 시작할지 정하는 것입니다. 기본적으로 이벤트 흐름은 비즈니스 프로세스를 기술 프로세스로 해석한 것입니다. 이벤트 흐름은 이벤트가 생성되어 브로커를 통해 전송되고 궁극적으로 소비되는 방식을 의미합니다.

이 단계에서는 먼저 시범 프로젝트로 시작해 실험하면서 인사이트를 도출한 다음 규모를 확대하는 것이 좋습니다.

시범 프로젝트에서 이벤트와 이벤트 흐름을 파악하면 이벤트 카탈로그가 자동으로 구성되기 시작합니다. 단순한 스프레드시트 형태로 관리하든 아니면 이벤트 포털에서 관리하든, 최초의 이벤트 카탈로그는 재사용 가능한 이벤트의 시작점으로도 기능합니다. 이러한 재사용 가능한 이벤트는 향후 애플리케이션에 의해 소비될 수 있습니다.

현재의 현대화 상태 또는 불편 감소에 가장 적합한 흐름을 선택하는 것이 좋습니다. 기내 프로젝트나 예정된 전환은 성능, 강건성 또는 클라우드 도입을 통한 기술적 부채 감소나 혁신에 적합한 후보입니다. 이러한 사례를 향후 구현의 참조로 삼아야 합니다. 빠르게 성과를 도출할 수 있는 흐름을 선택하세요.

다음의 가격 변화 흐름을 예시로 참조하세요.



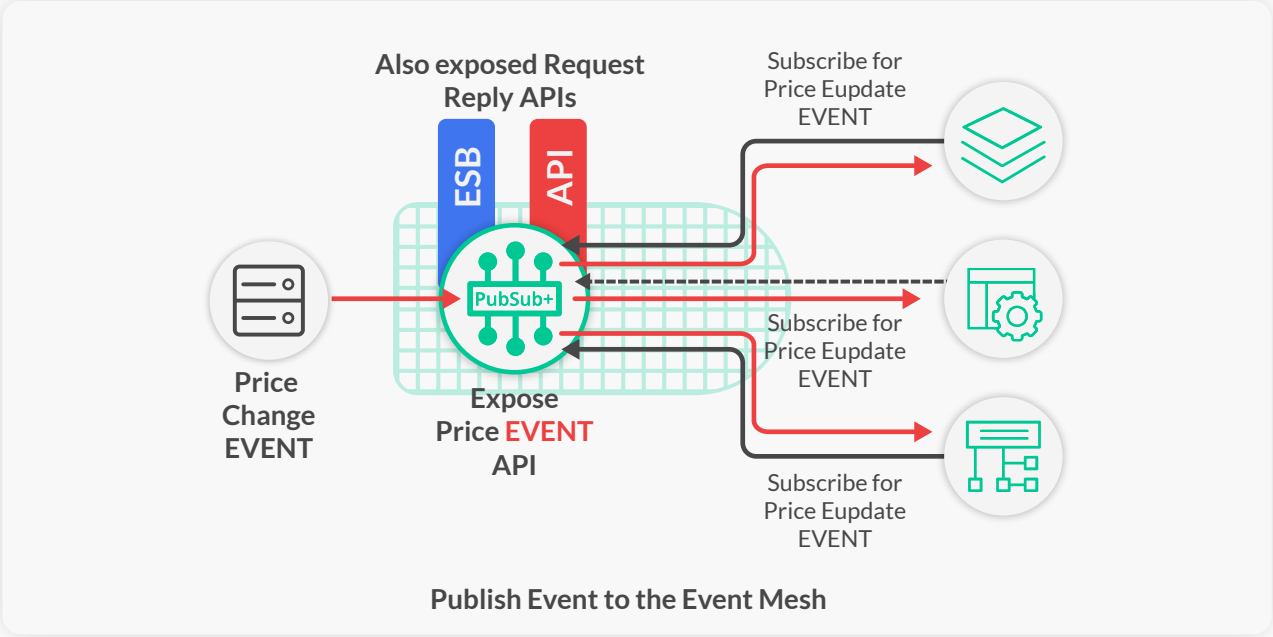
가격 변경 자체가 이벤트로 간주됩니다. 여기서 '가격'은 명사이고, '변경'은 동사입니다.

이벤트가 사용 해제된 기존 아키텍처에서 가격 변경 이벤트의 특징은 다음과 같습니다.

- 푸시되지 않음 - 요청/응답 API 또는 배치 프로세스가 필요할 수 있습니다.
- 실시간이 아님 - 다운스트림 애플리케이션은 묻기 전까지는 가격 변경 사항을 알지 못합니다.
- 비용 효과적으로 확장할 수 없음
- 버스트 발생 - 이벤트 애플리케이션이 API를 지속적으로 호출하여 부하/버스트가 발생할 수 있음

다운스트림 앱은 가격 변경을 이벤트로 사용 설정하고 이벤트 메시로 이벤트 중심 아키텍처를 구현하여 가격 변경 이벤트를 구독하고 실시간으로 이벤트 알림을 받습니다. 이벤트 메시는 다운스트림 애플리케이션이 구독한 이벤트만을 (주제 분류에 따라) 필터링하고 푸시합니다.

이벤트는 보다 간편한 규모 확대를 위해 대기열에 추가되고 조절되며, 손실 없고 보장된 방식으로 전달됩니다.



5단계: 이벤트 흐름을 비동기식 이벤트 중심 마이크로서비스로 분해

시범 흐름 파악 및 초기 이벤트 카탈로그 시작 후의 단계는 비즈니스 흐름을 이벤트 중심 마이크로서비스로 분해한 다음, 프로세스에 존재하는 이벤트를 식별하여 이벤트 중심 설계를 시작하는 것입니다.

이벤트 흐름을 마이크로서비스로 분해하면 각 마이크로서비스가 전체 이벤트 흐름 중 하나의 측면만 처리하므로 이벤트 소스를 가져오기 위한 노력을 줄일 수 있습니다. 어댑터를 사용하여 기존 애플리케이션(SAP, 메인프레임, 맞춤 앱)에 이벤트를 사용 설정하는 동시에 마이크로서비스를 사용하여 새로운 비즈니스 로직을 구축할 수 있습니다.

마이크로서비스 조정 vs 유통

조정과 유통의 두 가지 방법을 사용하여 이벤트 흐름 내에 있는 마이크로서비스를 관리할 수 있습니다. 조정의 경우, 마이크로서비스가 호출 및 응답(요청/응답)의 방식으로 작동하며, 긴밀하게 결합되어 있어 매우 상호 의존적이며 밀접하게 연결되어 있습니다. 이벤트 라우팅 유통의 경우, 마이크로서비스가 반응적(발생하는 이벤트에 반응)이며 느슨하게 결합되어 있습니다. 따라서 하나의 애플리케이션에 오류가 발생하는 경우에도 문제를 해결하는 동안 해당 애플리케이션에 의존하지 않는 비즈니스 서비스는 계속해서 정상 작동됩니다.

이 단계에서는 동기식으로 수행해야 하는 단계와 비동기식으로 수행해도 되는 단계를 식별해야 합니다. 동기식 단계는 흐름을 발생시키는 애플리케이션이나 API가 응답이나 블로킹을 기다리는 경우 실행되어야 하는 단계입니다.

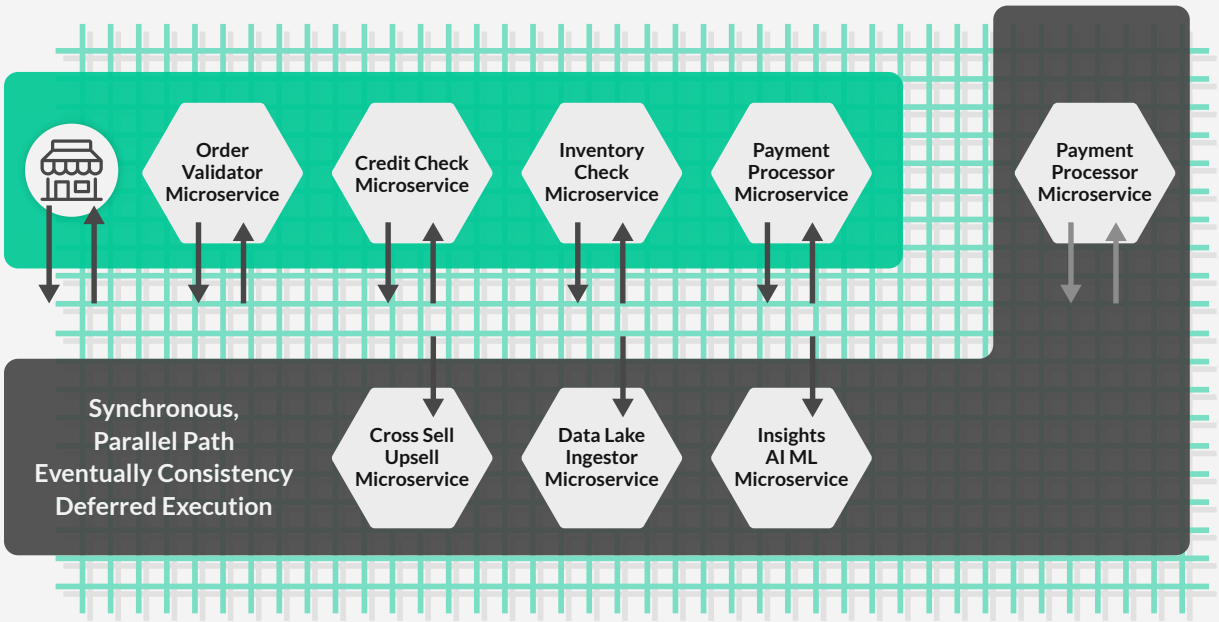
비동기식 이벤트는 실행 이후에 (대부분의 경우 병행하여) 발생할 수 있는 로깅, 감사, 데이터 레이트 작성과 같은 이벤트입니다. 즉, 궁극적인 일관성이 허용되는 애플리케이션의 경우 비동기식으로 처리해도 됩니다. 이 경우 이벤트는 이리저리 이동하며, 이러한 이벤트의 처리 방식은 마이크로서비스 유통에 의해 결정됩니다. 이러한 중요한 구분 때문에 흐름의 동기식 부분을 비동기식 부분과 분리해야 합니다.

이러한 이벤트 중심 프로세스 모델링은 우선 수동으로 실시하거나, 마이크로서비스를 시각화하고 유통할 수 있는 이벤트 포털에서 실시할 수 있습니다.

Udders Ice Cream 예시: 궁극적 일관성
궁극적 일관성

RESTful 동기식 API 기반 시스템의 한 가지 문제는 비즈니스 흐름(각 마이크로서비스)의 모든 단계가 직렬식이라는 것입니다. Udders Ice Cream의 POS 시스템은 주문을 제출할 때 모든 주문 단계가 완료될 때까지 기다려야 할까요, 아니면 일부 필수 단계만 직렬식이어야 할까요?

생각해보면 POS 시스템이 주문 확인을 받기 위해 주문 모든 '인사이트' 프로세스가 발생해야 할 필요는 없습니다. 이 경우 전반적인 반응 속도만 느려질 뿐입니다. 실제로 주문 받기 및 확인만 직렬식이면 되며, 나머지는 비동기식으로 발생해도 됩니다.



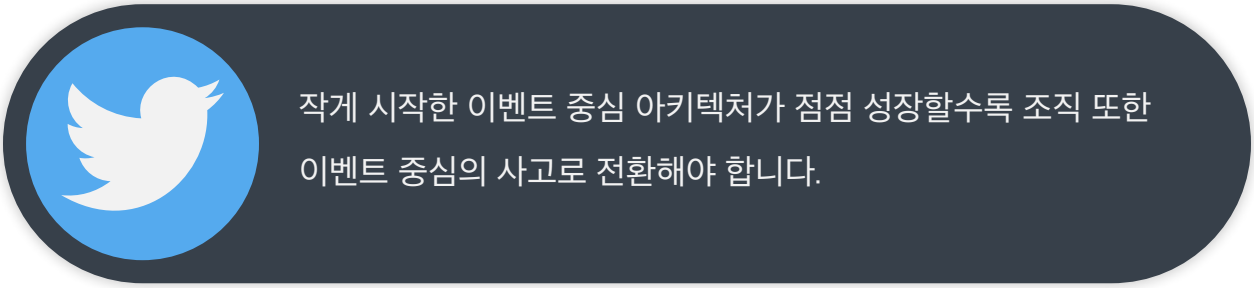
이벤트 메시는 확실한 전달을 보장하므로, 비직렬식 마이크로서비스는 이후에 조정된 방식으로 데이터를 확실히 받아볼 수 있습니다. 또한 이벤트 스트림이 동시다발적으로 시스템으로 유입되므로 성능과 지연이 개선됩니다.

기생적 경청자

이벤트는 이동하고 분류되는 와중에도 분석, 감사, 규정 준수와 데이터 레이크에 의해 소비될 수 있습니다. 승인된 애플리케이션은 와일드카드 기반의 필터링된 분산 이벤트 스트림을 수신하게 됩니다. 이때 기존 마이크로서비스의 변화나 ESB 또는 ETL 변화는 없습니다.

이벤트 소스와 싱크, 기존 애플리케이션에 대응하기

몇 가지 이벤트 소스와 싱크는 숙지하시는 것이 좋습니다. 일부 소스와 싱크는 이미 이벤트 중심이며, 이러한 이벤트를 소비하기 위한 처리도 거의 없습니다. 이외의 소스와 싱크를 'API에 바로 활용 가능'한 것으로 규정해보겠습니다. 'API에 바로 활용 가능'하다는 것은 이벤트 브로커가 (대부분의 경우 다른 마이크로서비스나 애플리케이션을 통해) 가져올 수 있는 메시지를 생성하는 API가 있음을 의미합니다. 일부 개발은 필요하지만, 이러한 소스는 비교적 쉽게 이벤트 브로커와 함께 작동하게 할 수 있습니다. 이외에도 이벤트 송수신이 불가능한 레거시 애플리케이션이 있는데, 이러한 애플리케이션은 이벤트 중심 애플리케이션과 프로세스에 통합하기가 어렵습니다.



다음은 세 가지 이벤트 소스와 싱크를 정리한 내용입니다.

	이벤트 중심	API에 바로 활용 가능	레거시
API 상태?	진정한 이벤트 중심	REST/SOAP API 및 스키마 보유	표준 기반 API 없음(단, 다양한 방법으로 연결 가능)
푸시/풀 (Push/Pull) 기능	이벤트 푸시 가능, 이벤트 소비 가능	요청/응답, 주제 또는 푸시 개념 없음	데이터 가져오기(pull) 또는 폴링(poll) 필요하며, 데이터가 트리거될 수 있음
스트리밍용 어댑터	필요 없음	마이크로서비스나 스트리밍 파이프라인이 요청-응답 소스를 스트리밍 도착지로 전환해야 하며, 지능형 주제를 발행해야 함	이벤트를 전환하고 pub/sub할 수 있도록 통합 어댑터(JDBC, MQ, JCA, ASAPIO, Striim, 레거시 ESB)를 이벤트 도착지의 소스 근처에 설치해야 함

이벤트 중심 애플리케이션

이벤트 중심 애플리케이션은 이벤트를 발행하고 구독할 수 있으며, 주제와 분류를 인식합니다. 이러한 애플리케이션은 실시간 반응형 애플리케이션으로, 이벤트 메시를 활용하여 이벤트 라우팅, 궁극적 일관성과 지연된 실행을 실시합니다.

API와 함께 즉시 사용 가능한 애플리케이션

API와 함께 즉시 사용 가능한 애플리케이션은 관련 주제로 이벤트를 발행할 수는 없지만 API를 통해 소비 가능한 이벤트를 발행할 수는 있습니다. 이 경우, 어댑터 마이크로서비스를 사용하여 'raw' API(이벤트 메시는 REST API를 주제로 변환할 수 있습니다)를 구독하고, 페이로드를 점검하고, 페이로드 내용에서 파생된 적절한 주제로 메시지를 발행할 수 있습니다. 이러한 접근법은 콘텐츠 기반 라우팅보다 더 효과적입니다. 콘텐츠 기반 라우팅은 의미론(semantics) 단위의 매우 엄격한 페이로드 거버넌스를 요구하므로 어떤 경우에는 실용적이지 못하기 때문입니다.

기존 애플리케이션

아마 API에 바로 활용할 수조차 없는 레거시 시스템이 있으실 겁니다. 대다수의 비즈니스는 이러한 레거시 시스템의 소비 데이터나 기여 데이터를 처리하므로, 이러한 시스템을 이벤트 메시와 통합할 방법을 찾아야 합니다. 시스템을 폴링해야 할까요? 데이터 요청이나 업데이트 요청이 발생하면 어댑터를 통해 실행해야 할까요? 아니면 이벤트를 오프램프하고 외부에서 캐시해야 할까요? 어느 경우든 API조차 없는 레거시 시스템에서 이벤트를 사용할 수 있는 방법을 모색해야 합니다.

레거시 시스템을 적응시키는 일의 핵심은 초기에 식별하여 최대한 빠르게 적응을 시작하는 것입니다.

자세한 정보와 예시는 bit.ly/sources-and-sinks를 참고하세요.

최대한 클라우드 네이티브로 전환하기

클라우드의 접근성과 확장성은 이벤트 메시의 접근성과 확장성을 반영합니다. 즉, 클라우드와 이벤트 메시는 함께 배포하기에 적합합니다. 다만 모든 이벤트를 클라우드에서 생성하거나 처리할 수는 없습니다. 온프레미스 시스템을 무시해서는 안 되기 때문입니다. 이벤트 메시를 구현했다면 데이터의 위치에 구애받지 않고 더욱 복합적인 접근법을 활용할 수 있습니다.

예를 들어 데이터 레이크는 Azure에 있는 반면, AI 및 ML 기능은 GCP에 있을 수 있습니다. 또한 제조 및 물류 시설은 중국이나 한국에 위치한 반면, 시장은 인도나 미국일 수 있습니다. 5G가 차세대 글로벌 연결의 시대를 열게 될 지금은 이벤트 중심의 근간이 하이브리드/멀티 클라우드와 인텔리전트 엣지라는 판이한 두 가지 패턴 모두와 함께 작동해야 합니다.

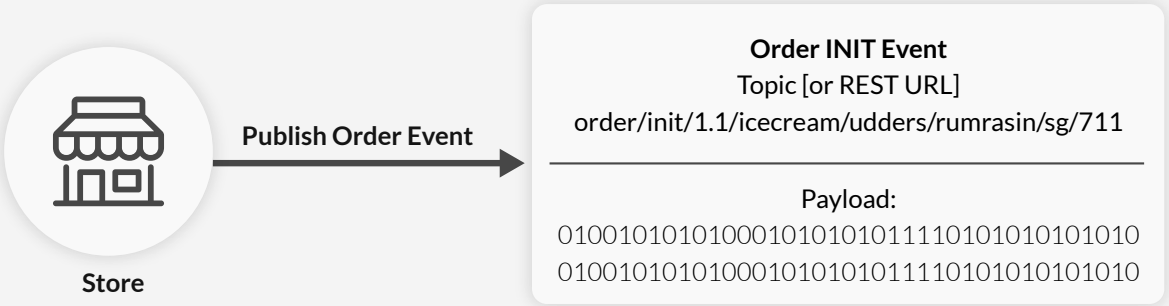
표준 프로토콜, 분류와 이벤트 메시를 사용하면 어디서든 비즈니스 로직을 실행할 수 있습니다.

Udders Ice Cream 예시: 주문 관리

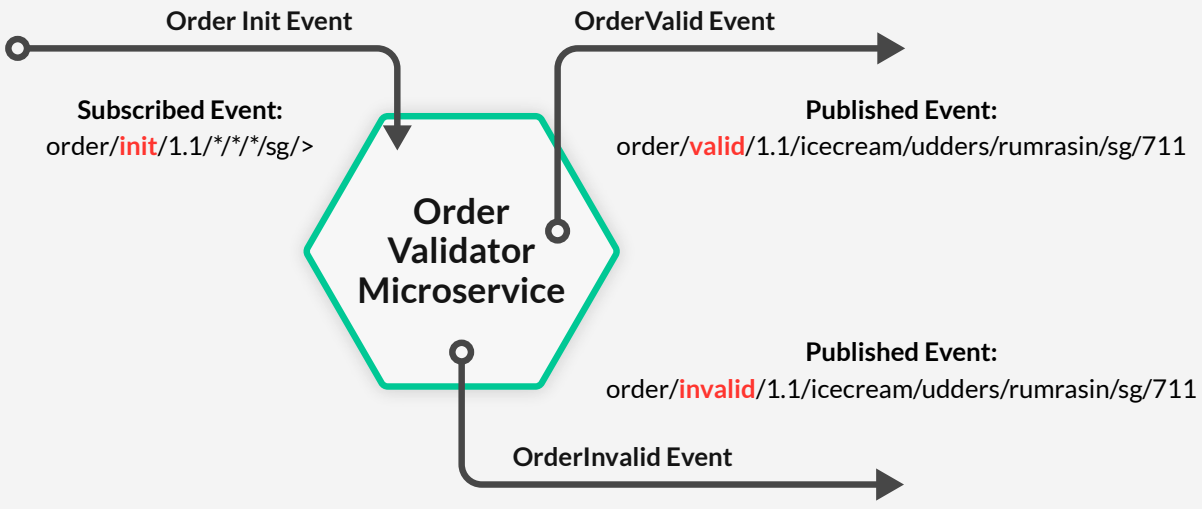
Udders Ice Cream의 주문 관리 절차에서는 POS 후 이벤트 흐름에서 다음의 마이크로서비스가 등장할 수 있습니다.

- 주문 검증 도구
- 신용 확인
- 재고 확인
- 결제 처리 도구
- 주문 처리 도구

POS 시스템이 럼 앤 레이즌 맛 아이스크림 주문을 제출하기 위한 API 호출을 실행하면 이 주문이 시작됩니다.



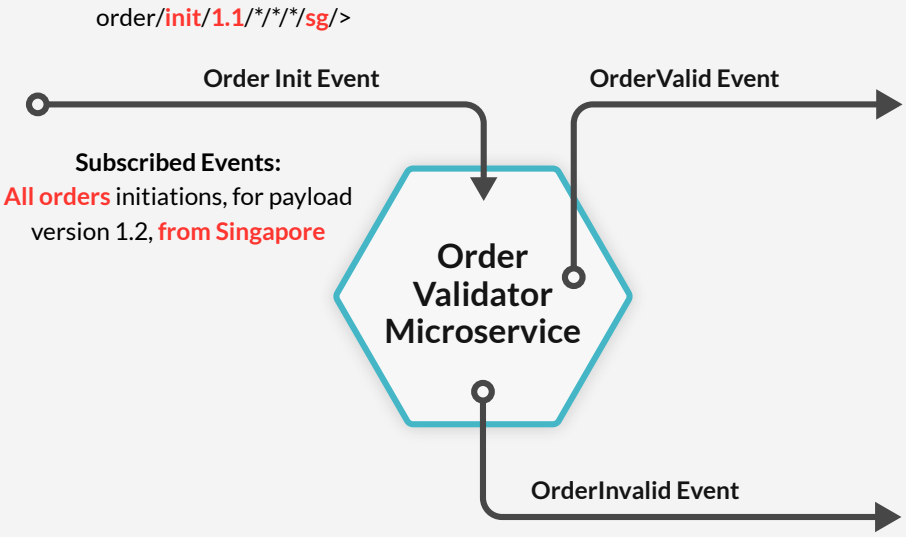
주문 검증 도구 마이크로서비스는 신규 주문 이벤트를 소비하고 유효한 주문 이벤트나 무효한 주문 이벤트를 생산합니다. 유효한 주문 이벤트는 신용 확인 마이크로서비스에 의해 소비되며, 이 마이크로서비스 또한 다음 이벤트를 생산합니다.



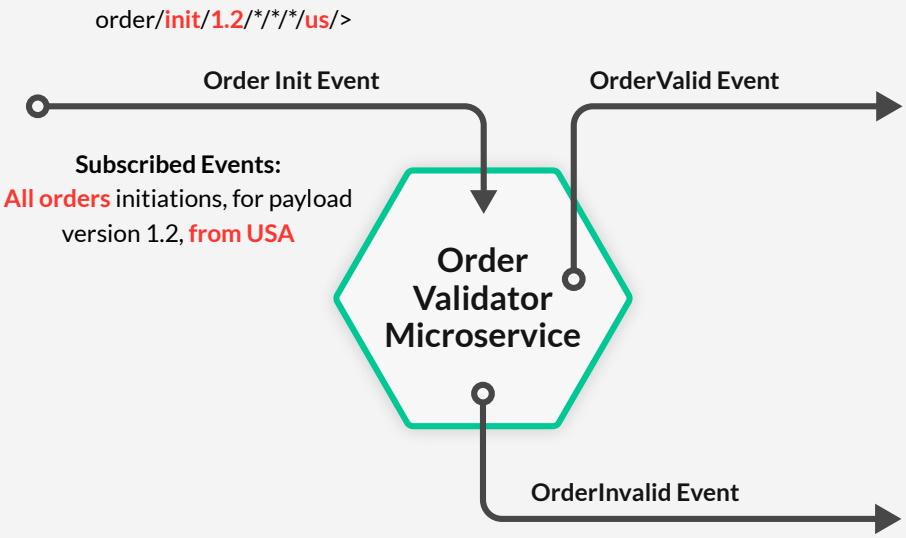
- **구독한 이벤트:** 버전 및 제품과 관계없이 모든 주문 시작
- **발행한 이벤트:** 주문 확인 상태 및 기타 메타데이터

Udders Ice Cream 예시: 이벤트 라우팅

문제: 주문 확인 규칙이 싱가포르에서 미국으로 변경되었으며, 페이로드가 JSON에서 protobuf로 변경되었습니다.

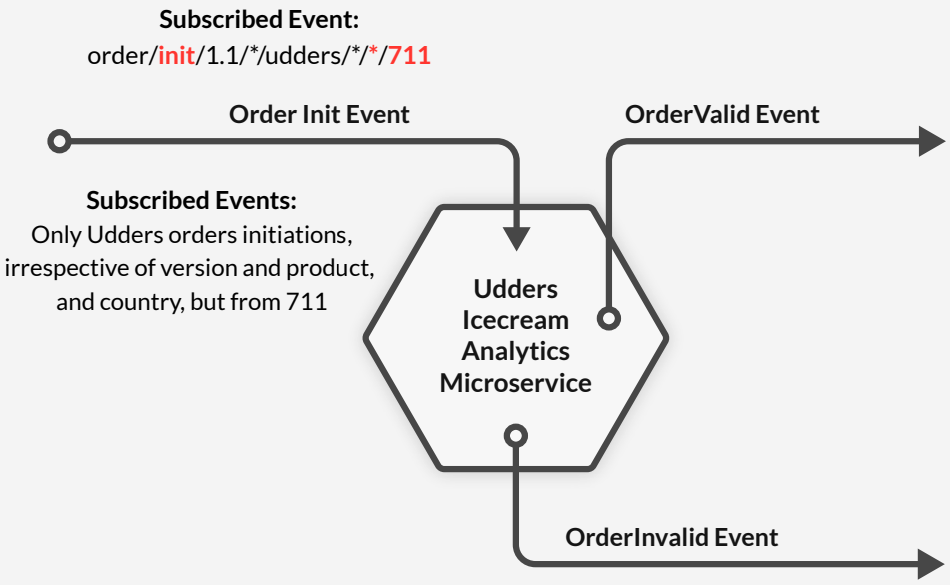


해결책: 새로운 페이로드로 미국의 주문을 확인하는 비즈니스 로직을 실행하고 주제 구독을 재연결하기 위해 새로운(재사용된) 마이크로서비스를 생성합니다. 새로운 마이크로서비스는 관련 주제를 듣고 버전 1.2의 미국에서 발생한 주문 시작 이벤트를 소비합니다. 이외에는 아무것도 변경되지 않습니다.



문제: 711 체인에서 유입되는 모든 주문을 모니터링하려고 합니다.

해결책: 관련 인사이트 비즈니스 로직이 포함된 마이크로서비스를 구현하고, 이벤트 카탈로그에서 해당 이벤트를 찾아보고 와일드카드를 사용하여 이벤트 구독을 시작합니다.



6단계: 더 빠르게 수익 창출

디자인을 완료하고 첫 애플리케이션이 이벤트 네이티브 애플리케이션으로 제공되면 이벤트 카탈로그도 채워지기 시작합니다.

이벤트 카탈로그를 주 제공품으로 하여 빠르게 수익을 거두는 것은 다른 비즈니스 로직만큼이나 중요하며, 이는 혁신과 재사용을 촉진합니다.

더 많은 실시간 수행이 가능해짐으로써 애플리케이션의 민첩성과 반응성도 증가하여 고객 경험이 향상되는 것이 목표입니다. 이해관계자의 참여와 함께 조직 전체를 혁신해 보세요.

보너스 단계: 초기화 및 반복

이제 이벤트 중심으로 전환할 수 있는 템플릿을 확보했으며 어쩌면 첫 성공도 거두셨을 것입니다. 다음으로는 처음으로 돌아가 이 과정을 반복하면 됩니다. 다음 이벤트 흐름을 선택하세요!

이제 다음 프로젝트용으로 단계를 다시 거치면 됩니다. 이 과정에서 이벤트 포털의 이벤트 카탈로그가 점점 더 많은 이벤트로 채워지며, 신규 소비자와 프로듀서가 기존의 이벤트를 재사용하게 됩니다. 더불어 풍부한 카탈로그를 활용하여 실시간 처리와 인사이트 확보의 기회도 늘어나게 됩니다.

성과가 나타나면 문화 변화도 더욱 용이해집니다. 통합/미들웨어 팀은 이벤트 카탈로그와 이벤트 메시를 소유하는 반면, 애플리케이션 및 LOB 팀은 이벤트 카탈로그와 이벤트 메시를 사용하고 이에 기여합니다. 조직의 기술팀과 기술 프로세스의 연합 및 중앙화 정도에 따라 LOB별 이벤트 카탈로그와 현지화된 이벤트 브로커 또한 바람직한 패턴이 될 수 있습니다.

규모를 확장할수록 이벤트를 생성하고 소비하는 애플리케이션도 증가하며, 이는 보통 기존 이벤트의 재사용에서 시작됩니다. 이것이 바로 이벤트 중심 여정이 규모를 확장하면서 확대되는 원리입니다.

결론

끊임없이 변화하는 실시간 비즈니스는 무엇보다도 디지털 혁신을 추구해야 합니다. 세상의 속도가 느려질 일은 없으므로, 비용 효과적 및 효율적으로 기업의 아키텍처를 업그레이드하여 시대의 변화에 적응하는 것이 가장 좋은 방안입니다. 하지만 쉬운 일이 아닙니다.

대다수의 IT 전문가들은 학교에서부터 점진적으로 사고하도록 배웠습니다. 저처럼 Fortran, C, Java나 Node에서 시작한 사람들은 대부분 동기식 함수 호출, 동기식 RPC 호출, 동기식 웹 서비스와 API 또는 동기식 ESB 조정 흐름에 대한 IT 교육을 받고 이에 관한 경험을 쌓았습니다. 때문에 비동기식 메시지에 익숙해지려면 대대적인 변화와 사고 프로세스의 전환이 필요합니다.

마이크로서비스는 서로를 호출하여 분산되고 고립된 요소들을 만들 필요가 없습니다. 이벤트는 이벤트 메시에서 이동하며 마이크로서비스에 의해 소비되고 실행될 수 있습니다. 설계자와 개발자는 협업을 통해 조직을 위한 실시간 이벤트 중심 목표를 달성할 수 있는 플랫폼과 도구를 필요로 합니다.

이 6단계를 실시하면 최적의 전략과 도구를 활용하여 실시간 이벤트 중심 여정을 신속하게 진행할 수 있습니다. Solace PubSub+ Platform은 기업의 이벤트 중심 아키텍처 설계, 배포 및 관리에 도움이 되며, 모든 클라우드와 플랫폼에 서비스로 배포할 수 있습니다. Solace는 조직에서 이벤트 중심 아키텍처를 구현하려는 설계자의 고유한 니즈를 충족하는 안정적이고 탁월한 단 하나의 솔루션입니다.

Sumeet Puri 소개

Sumeet Puri는 Solace의 최고 기술솔루션 책임자로서 CIO, CTO 및 수석 설계자의 실시간 이벤트 중심 기술 전환을 지원하며, 다수의 기술 포럼에서 연사로도 활동합니다.

오타와 | 뉴욕 | 실리콘밸리 | 런던 | 파리 | 취리히 | 도쿄 | 서울 | 시드니 | 싱가포르 | 뭄바이

Solace 소개

Solace는 대규모 엔터프라이즈가 이벤트 중심의 비즈니스 운영과 고객 상호작용을 실현하는 데 필요로 하는 모든 사항을 제공하여 최첨단 및 실시간 엔터프라이즈로 전환되도록 도와드립니다. 업계 최초이자 유일한 이벤트 관리 플랫폼인 PubSub+을 사용하면, 안전하고 안정적이면서도 빠르고 확실하게 이벤트를 생성, 기록 및 검색할 수 있을 뿐만 아니라 이벤트가 생성된 위치에서 사용될 위치로 스트리밍할 수 있습니다. solace.com에서 자세히 알아보세요.

Solace 계정

Solace 고객사

airtel BARCLAYS bet365 NASA SAP

RBC Standard Chartered FxPro Liquidnet TMX

Solace의 주요 파트너

boomi A Dell Technologies Business SAP f5 openfin

VMware Tanzu