AWS Event Driven Architecture

Jae Hyeon Kim

Contents

소개

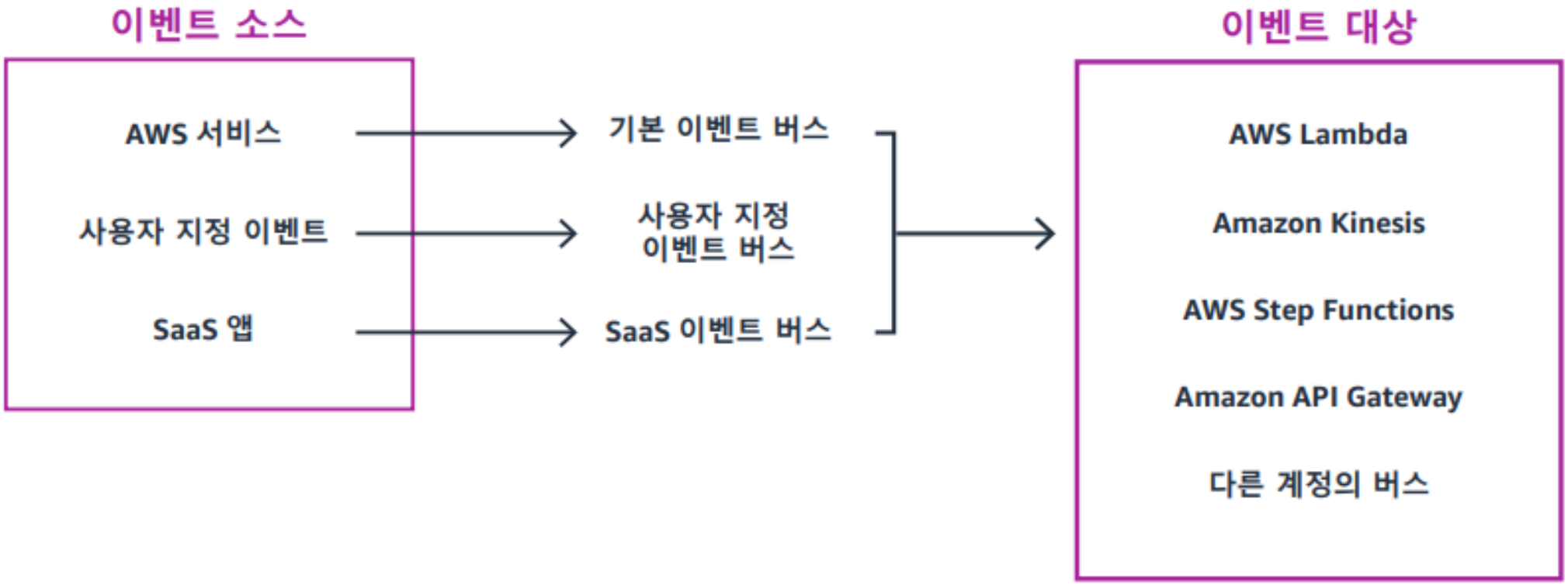
Event Driven Architecture

- 이벤트 기반 아키텍처(EDA)는 서비스 구성 요소를 연결하고 복잡한 시스템의 통신을 구현하도록 설계된 아키텍처 패턴
- 이벤트 생산자(ex: 서비스형 소프트웨어 앱, 모바일 앱, 전자 상거래 사이트, POS)
- 이벤트 브로커(ex: 이벤트 라우터, 이벤트 스토어)
- 이벤트 소비자(ex: 데이터베이스, 마이크로서비스, SaaS 애플리케이션)

Event Driven Architecture

- 이벤트는 상태의 변경(주문접수, 사용자 생성)을 말하며, 특정 의미 체계를 사용한 메시지로 구성 요소 간에 전달된다.
- 생산자는 브로커에 이벤트를 게시하며, 브로커는 생산자와 소비자의 비동기적 통신을 통해 둘을 분리 추상화한다.
- EDA는 이벤트를 사용해 약결합 서비스 간의 통신을 조정한다.

Event Driven Architecture



현대적 애플리케이션 구축에 EDA를 도입하는 이유

Modern Applications on EDA

- 이벤트 기반 아키텍처를 사용하면 확장 가능하고 신뢰할 수 있는 방식으로 반응형 고객 경험을 제공할 수 있다.
- 점점 더 많은 조직이 다양한 이유로 현대적 마이크로서비스 애플리케이션 구축에 EDA를 사용하고 있다.

Modern Applications on EDA

- 여러 팀을 아우르는 민첩성
- 현대적 애플리케이션은 복잡성이 증가하고 있으며, 많은 경우 여러 개의 사용자 지정 클라우드 기반 서비스, SaaS 서비스 및 온프레미스 서비스로 구성된다. 이러한 서비스를 관리하는 개발팀은 새로운 서비스를 업데이트, 관리, 구축하기 위해 민첩성을 갖추고 독립적으로 운영해야 한다.

Modern Applications on EDA

- 대규모 복원력
- 현대적 애플리케이션은 복원력과 신뢰성이 뛰어나야 하며, 서비스가 수요에 따라 확장되고 어느 한 곳의 장애가 전체 서비스에 영향을 미치지 않아야 한다.

Modern Applications on EDA

- 고객 기대치
- 현대적 애플리케이션은 고객의 필요에 신속하게 응답해야 한다. 고객은 이벤트 발생 즉시 시스템에서 응답하기를 원한다.

Modern Applications on EDA

- 비용 효율성
- 현대적 애플리케이션은 비용 효율적이어야 한다. 즉, 고객이 필요한 서비스에 대한 비용을 서비스가 필요할 때만 지불할 수 있도록 하여 애플리케이션의 총 소유 비용(TCO)을 낮추어야 한다.

Modern Applications on EDA

- 이벤트 기반 아키텍처가 도입되기 전에는 애플리케이션을 업데이트하거나 애플리케이션에 새 서비스를 추가할 때 개발 팀의 긴밀한 공조가 필요했다. 한 가지 서비스를 변경하면 의도치 않게 애플리케이션의 다른 서비스에 영향을 미칠 수 있었기 때문에 팀은 담당하지 않는 작업의 운용 방식도 파악해야 했다.
- 이로 인해 개발자가 민첩하게 독립적으로 운영할 수 있는 능력에 제약이 있었다.
- 이러한 서비스는 서로 강결합되어 있었기 때문에 각 구성 요소의 확장성 및 가용성이 전체 시스템에 영향을 미쳤고, 한 시스템의 장애가 다른 시스템의 장애를 유발했다.

이벤트 기반 아키텍처의 이점

Benefits of EDA

1. 민첩성 향상
2. 확장성 및 내결함성
3. 온디맨드 리소스
4. 실시간 응답성

Benefits of EDA

- 민첩성 향상
- 이벤트 기반 아키텍처는 이벤트 브로커를 사용하기 때문에 생산자 및 소비자 서비스를 구축하고 유지 관리하는 개발자 팀이 서로 긴밀하게 공조해야 할 필요가 없다. 따라서 팀은 빠르고 독립적으로 움직일 수 있다.
- EDA는 관리형 애플리케이션 통합 서비스를 사용하기 때문에 개발자가 이벤트와 이벤트 서비스를 관리하기 위해 작성해야 하는 사용자 지정 코드의 양이 대폭 줄어들며, 클라우드 제공업체에서 관리를 담당한다.

Benefits of EDA

- 확장성 및 내결함성
- 이벤트 기반 아키텍처는 약결합 형태이다. 즉, 독립적으로 확장하고 실패할 수 있어 애플리케이션의 복원력이 향상된다.
- 수요에 따라 확장하는 경우 EDA가 예측 불가능한 트래픽을 처리할 수 있기 때문에 전체 애플리케이션 아키텍처의 유연성이 높아지는 한편 단일 장애점이 사라진다.

Benefits of EDA

- 온디맨드 리소스
- 이벤트 기반 아키텍처는 푸시 기반이다. 즉, 이벤트가 브로커 및 다운스트림 시스템에 전송되면서 수요에 따라 모든 것이 발생한다. 그래서 종속 서비스의 이벤트에 대해 알릴 필요가 없다.
- 고객은 지속적인 폴링 인프라에 드는 비용을 지불할 필요가 없으며, 리소스는 이벤트 볼륨에 따라 스케일 업 및 스케일 다운할 수 있어 비용이 절감된다. 이 부분은 EDA를 사용하는 애플리케이션의 TCO를 낮추는 요인이다.

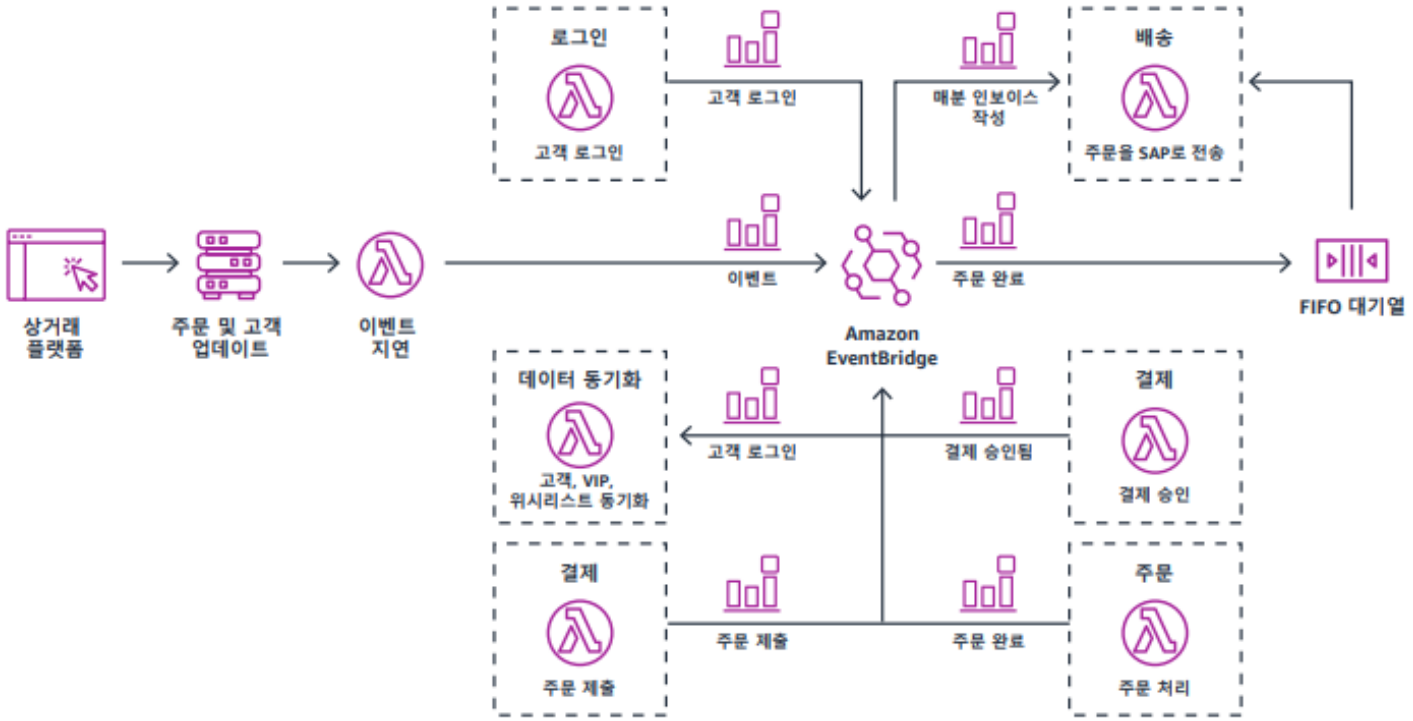
Benefits of EDA

- 실시간 응답성
- 이벤트 기반 아키텍처를 사용하면 애플리케이션에서 이벤트에 실시간으로 응답할 수 있다.
- 서비스에서 이벤트에 응답하므로 지연이나 장애가 다른 서비스의 응답 방식 또는 여부에 영향을 미치지 않는다. 따라서 다른 서비스에서 일어나고 있는 상황에 관계없이 고객이 더 나은 경험을 누릴 수 있다.

이벤트 기반 아키텍처의 일반 사용 사례

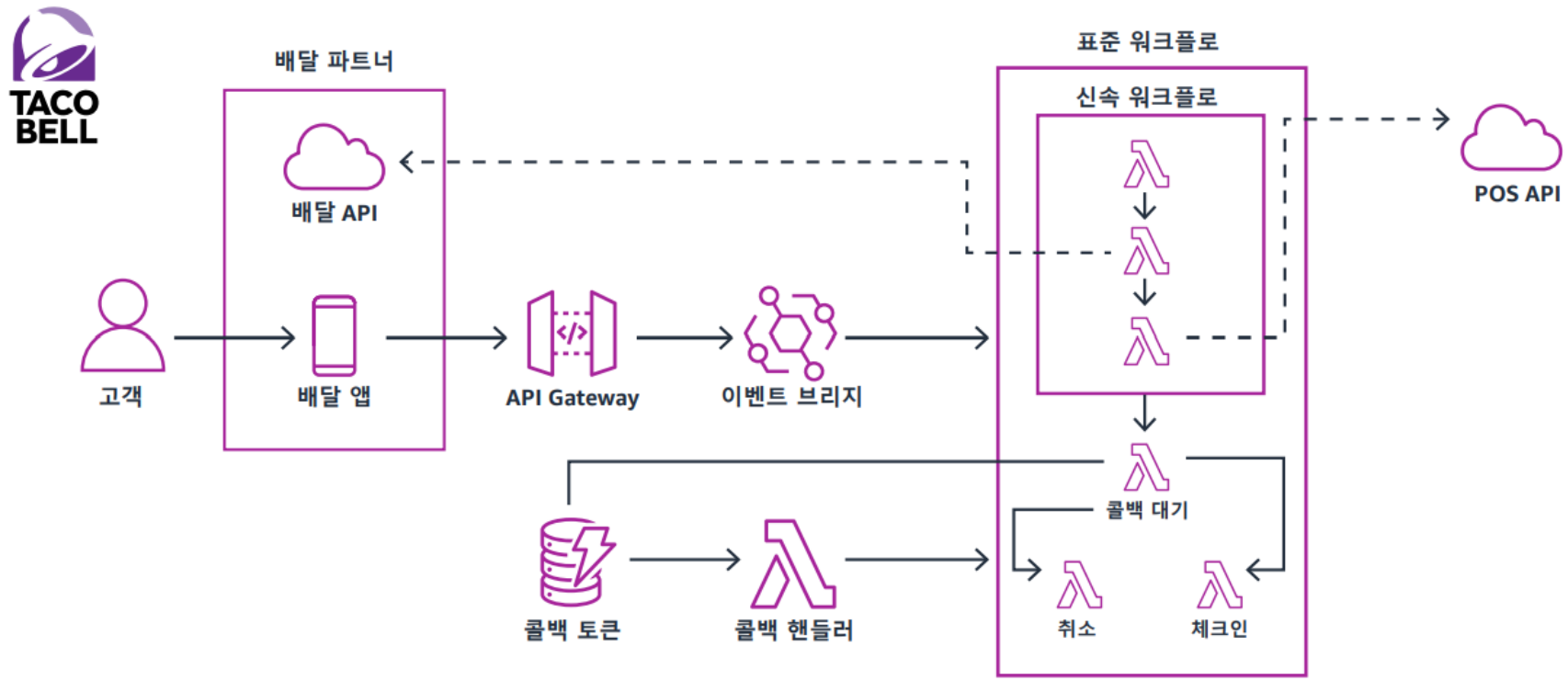
Microservice communication on web and mobile backends

- 예측 불가능한 트래픽을 처리하기 위해 스케일 업이 필요한 소매 또는 미디어 및 엔터테인먼트 웹사이트에서 흔히 찾아볼 수 있는 사용 사례.



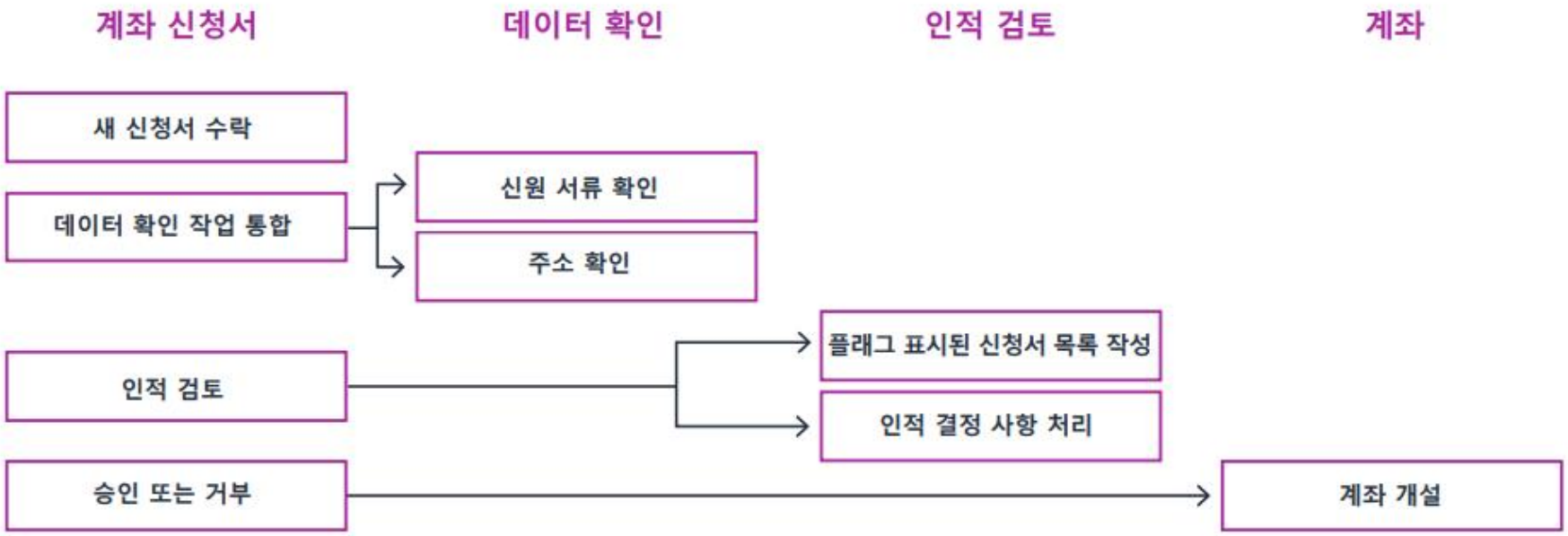
SaaS Application Integration

- 고객은 사일로화된 데이터를 활용하기 위해 이벤트 기반 아키텍처를 구축하여 SaaS 애플리케이션 이벤트를 모으거나 이벤트를 SaaS 애플리케이션으로 전송한다.

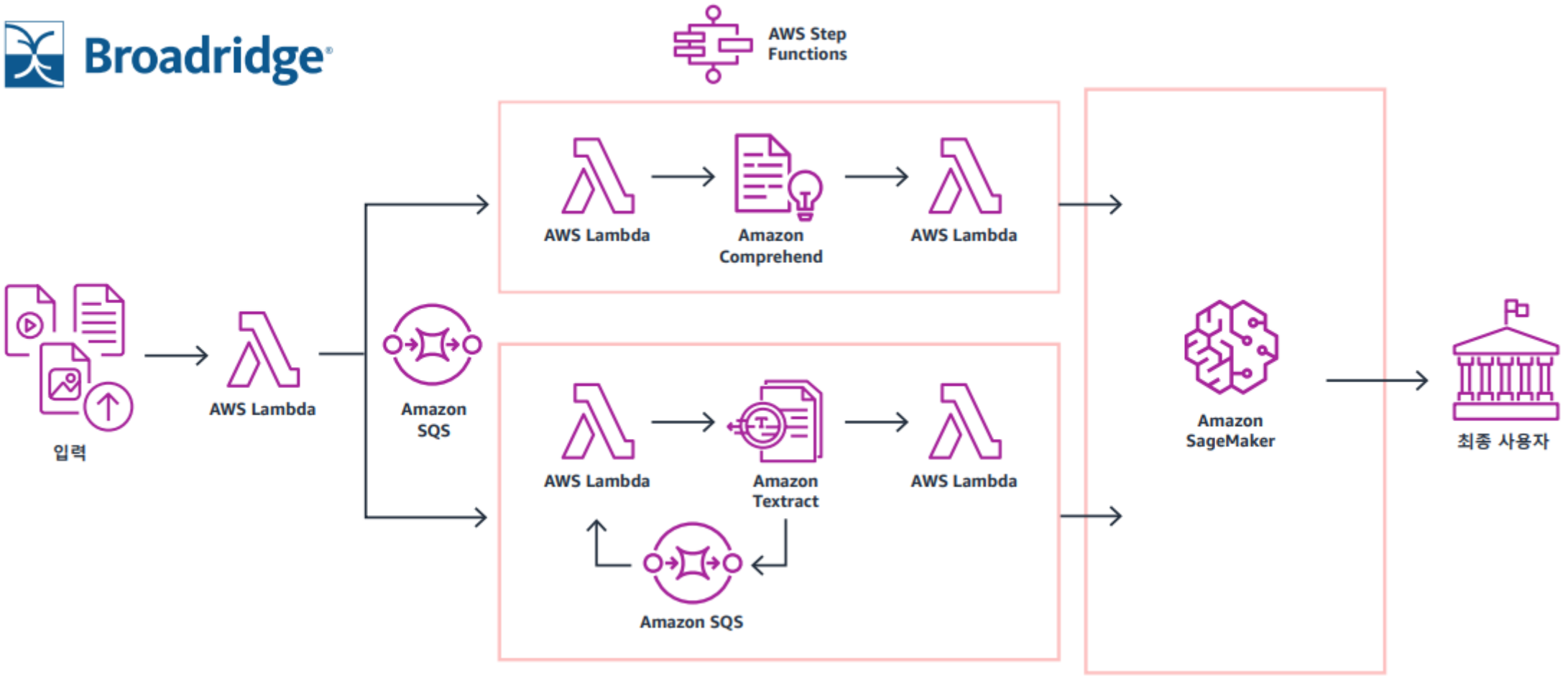


Business Workflow Automation

- 흔히 금융 서비스 거래 또는 비즈니스 프로세스 자동화에서 흔히 찾아볼 수 있다.
- 많은 비즈니스 워크플로에서 필요한 동일 단계의 반복 작업을 이벤트 기반 방식으로 자동화하고 실행할 수 있다.



Business Workflow Automation

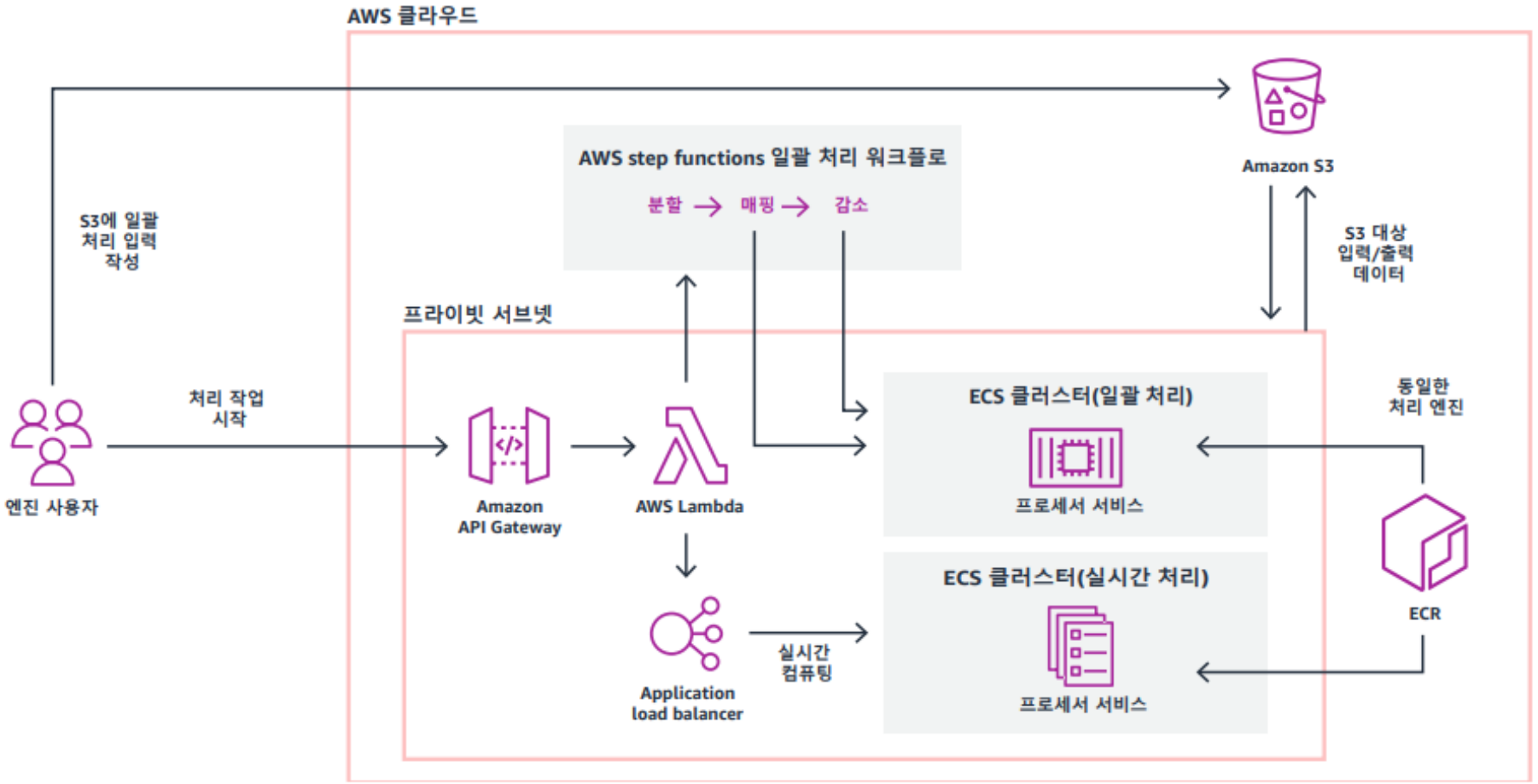


Infra Automation

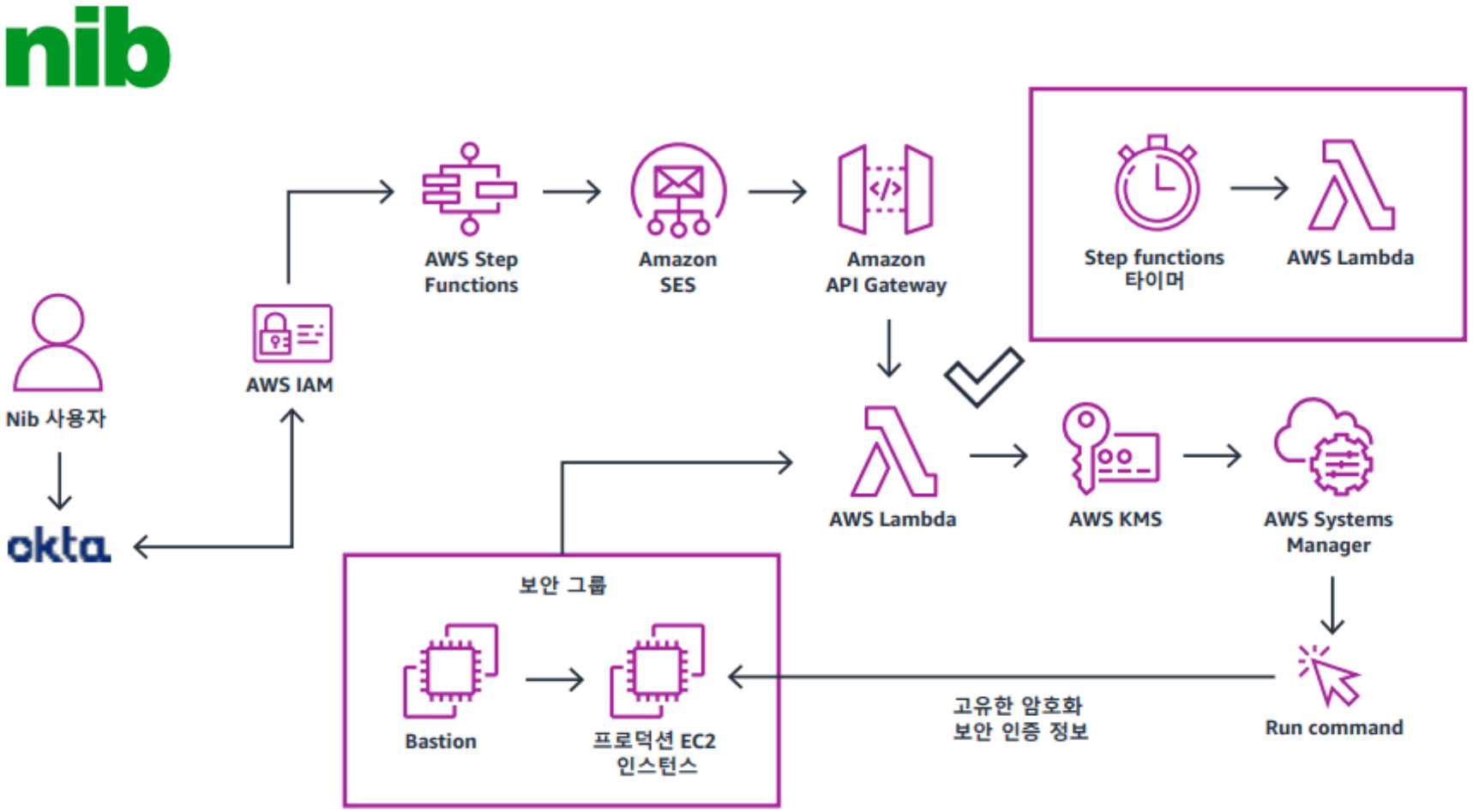
- 고객은 이벤트 기반 아키텍처를 사용해 다양한 방식으로 인프라를 자동화할 수 있다.
- 금융 분석, 유전체 연구, 미디어 트랜스코딩 등 컴퓨팅 집약적 워크로드를 실행하는 고객은 컴퓨팅 리소스를 트리거하여 고도의 병렬 처리를 위해 스케일 업한 후, 작업이 완료되면 리소스를 스케일 다운할 수 있다.

Infra Automation

SOCIETE GENERALE



Infra Automation



AWS로 이벤트 기반 아키텍처 구축

Infra Automation

- 이벤트 기반 아키텍처는 독자적인 사용 사례에 맞게 여러 AWS 서비스를 조합하여 만드는 경우가 많다.
- 이벤트를 생성하는 200여 가지의 AWS 서비스 모두 이벤트 생산자가 될 수 있다. 이벤트의 예로는 Amazon S3 버킷에서 생성된 새 파일, AWS Step Function 워크플로에서 완료된 단계 등이 있다.
- 고객은 다양한 브로커를 선택할 수 있으며, 각 서비스는 다양한 사용 사례에 맞는 특성을 갖추고 있다.
- 이벤트 소비자의 경우 서버리스 마이크로서비스로 이벤트를 처리하기 위한 Lambda 함수 호출, AWS Step Function 워크플로 트리거 등이 일반적인 패턴이다.