Event Driven MicroService Introduction

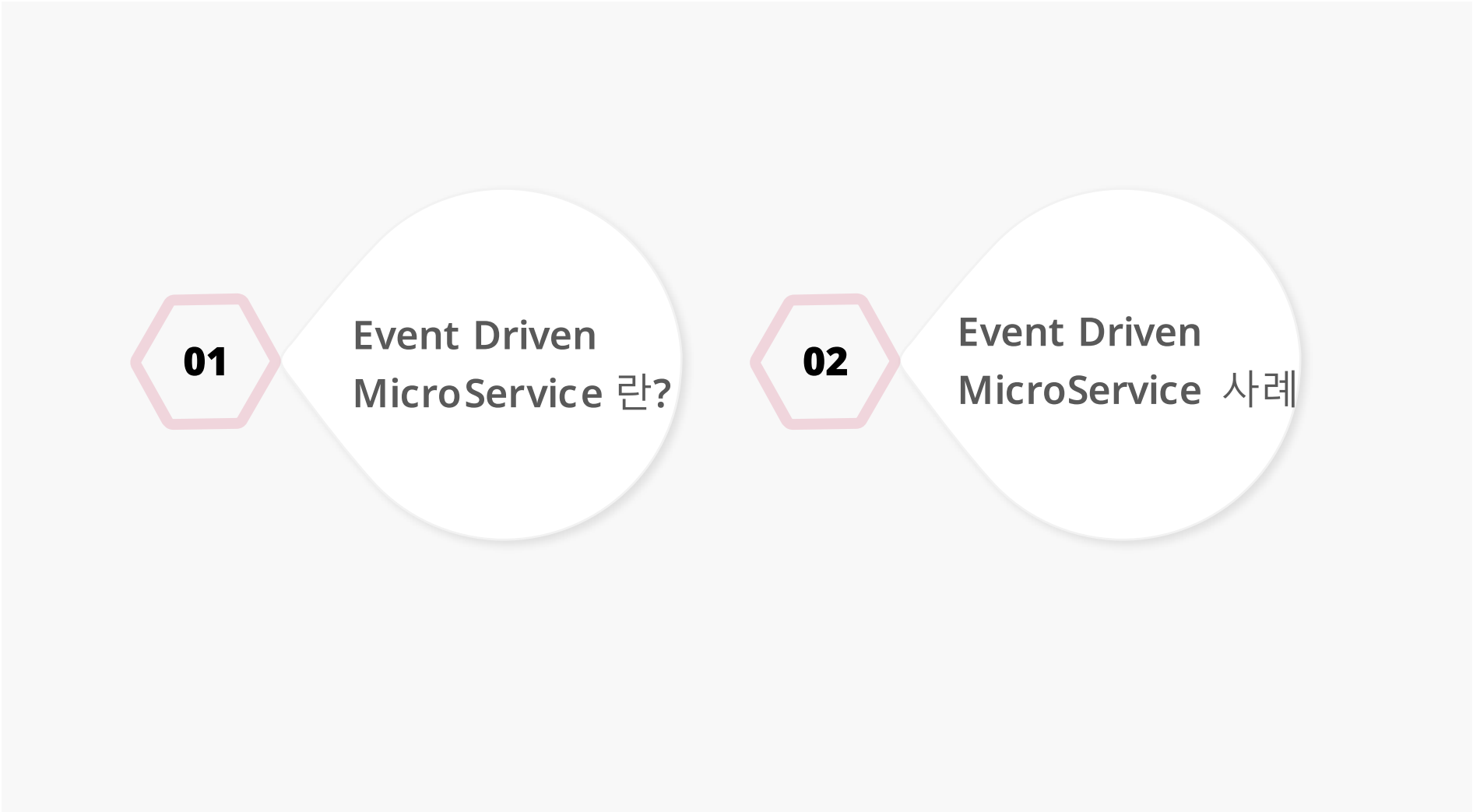
V.1.0.0

2018.10

SK 주식회사

Table of Contents

PART 01. Event Driven MicroService 란?



MicroService 란 ?

MicroService에 대해 알아봅니다.

- **MicroService 란 ?**
  - 단독으로 실행 가능하고
  - 가벼운 통신 프로토콜을 사용하며
  - 독립적으로 배치될 수 있는
  - 작은 단위로 기능이 분해된 서비스

Event Driven 이란 ?

Event Driven에 대해 알아봅니다.

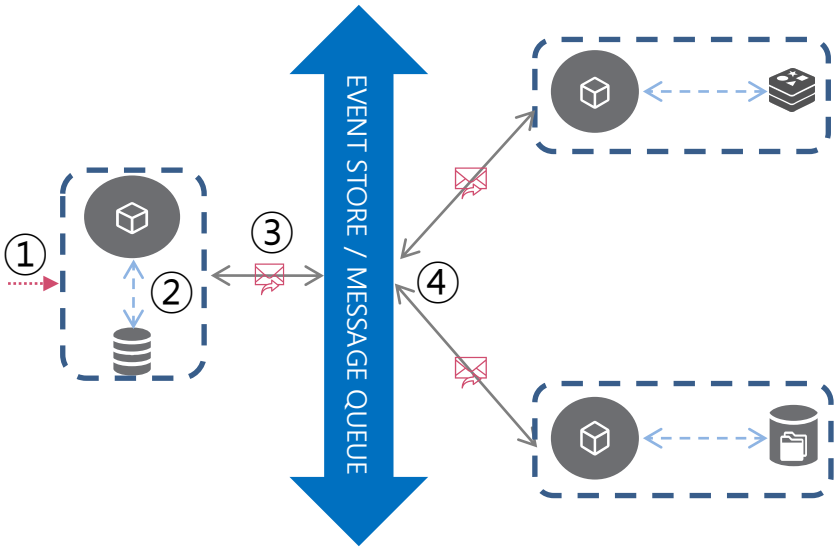
Event Driven 은

- IT 영역에서 아주 오래된 키워드입니다.
- 2018년 가트너에서 선정한 유망한 기술 트렌드 중 하나로 뽑혔습니다.

  (Top 10 Strategic Technology Trends for 2018: Event-Driven Model)
- Programming, Architecture와 연결되어 다양한 정의로 표현됩니다.
  - 컴퓨터 회로를 구동시키기 위해 동작하는일 – 마우스 클릭, 키보드 타이핑 등
  - IOT 기기 등의 임베디드 센서로부터 유입되는 스트리밍에 반응하는 동작
  - **시스템 내 외부에 발생한 주목할 만한 상태의 변화에 기반한 동작**

Event Driven MicroService 란 ?

Event Driven MicroService에 대해 알아봅니다.



이벤트란?

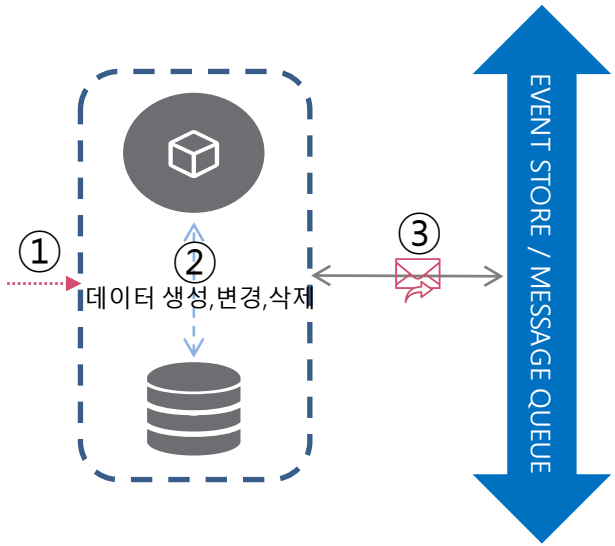
의미 있는 데이터의 생성,변경,삭제

Event Driven MicroService 란 ?

- 이벤트 발생시
- 해당 이벤트 로그를 보관하고
- 이를 기반으로 시스템이 동작하며
- 비동기 통신을 통해 시스템 내 통합을 수행하는 Architecture
- Application은 이벤트와 이벤트 사이의 기능을 수행

Event Driven MicroService 란 ?

이벤트를 데이터의 생성,변경,삭제로 정의했기 때문에 **MSA**의 데이터 관리와 밀접한 연관성을 갖습니다.



1. 특정 서비스에서 기능을 수행
2. 데이터의 생성,변경,삭제에 따른 이벤트 생성
3. 생성된 이벤트는 저장 공간에 보관
4. 비동기 메세지 큐로 해당 이벤트에 관심이 있는 서비스들에게 전달
5. 이벤트를 구독한 서비스는 biz. logic을 수행
6. 수행 도중 오류가 발생하면 저장된 이벤트 로그를 기반으로 retry/rollback을 수행

Event Driven MicroService 적용 사례

이벤트 기반으로 마이크로서비스를 확장하려는 의도는 여러 공개된 정보에서 읽을 수 있습니다

coupang

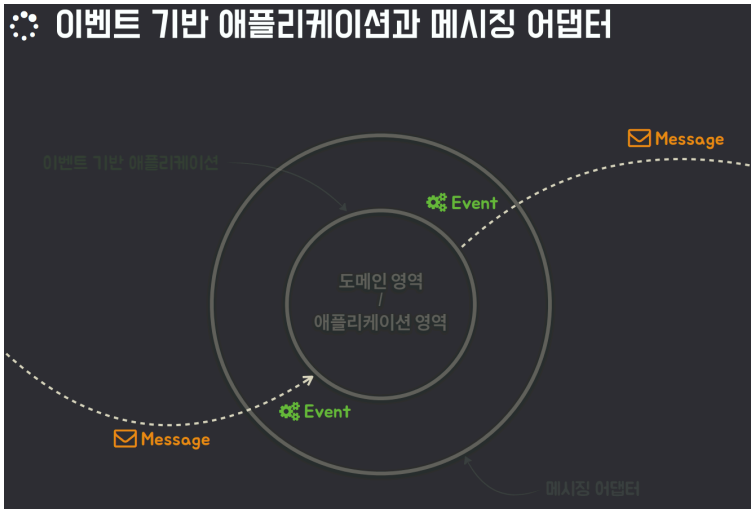
NETFLIX

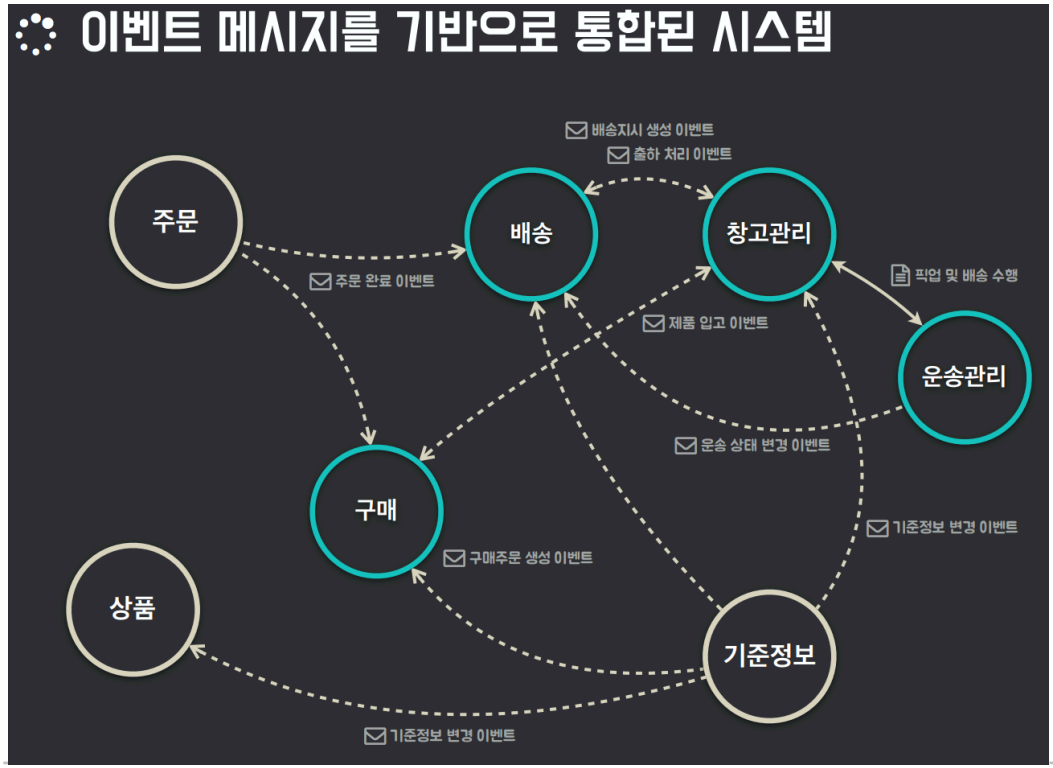
Event Driven MicroService 적용 사례

이벤트 기반으로 마이크로서비스를 확장하려는 의도는 여러 공개된 정보에서 읽을 수 있습니다

출처 : https://www.slideshare.net/arawnkr/ss-94475606

Spring Camp 2018 – 우아한형제들 – 배민찬 – SCM 시스템

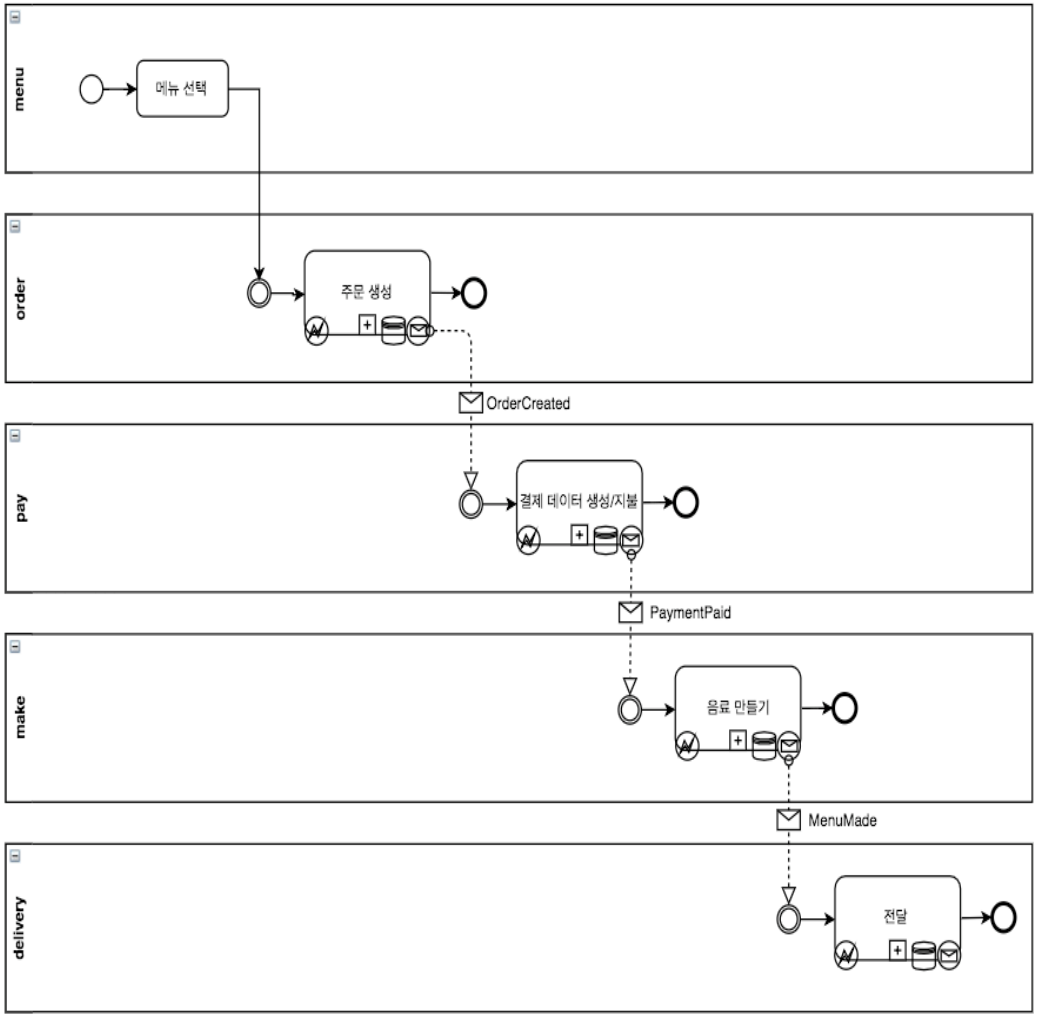




Part 02. Event Driven MicroService 적용 효과

비즈니스 흐름

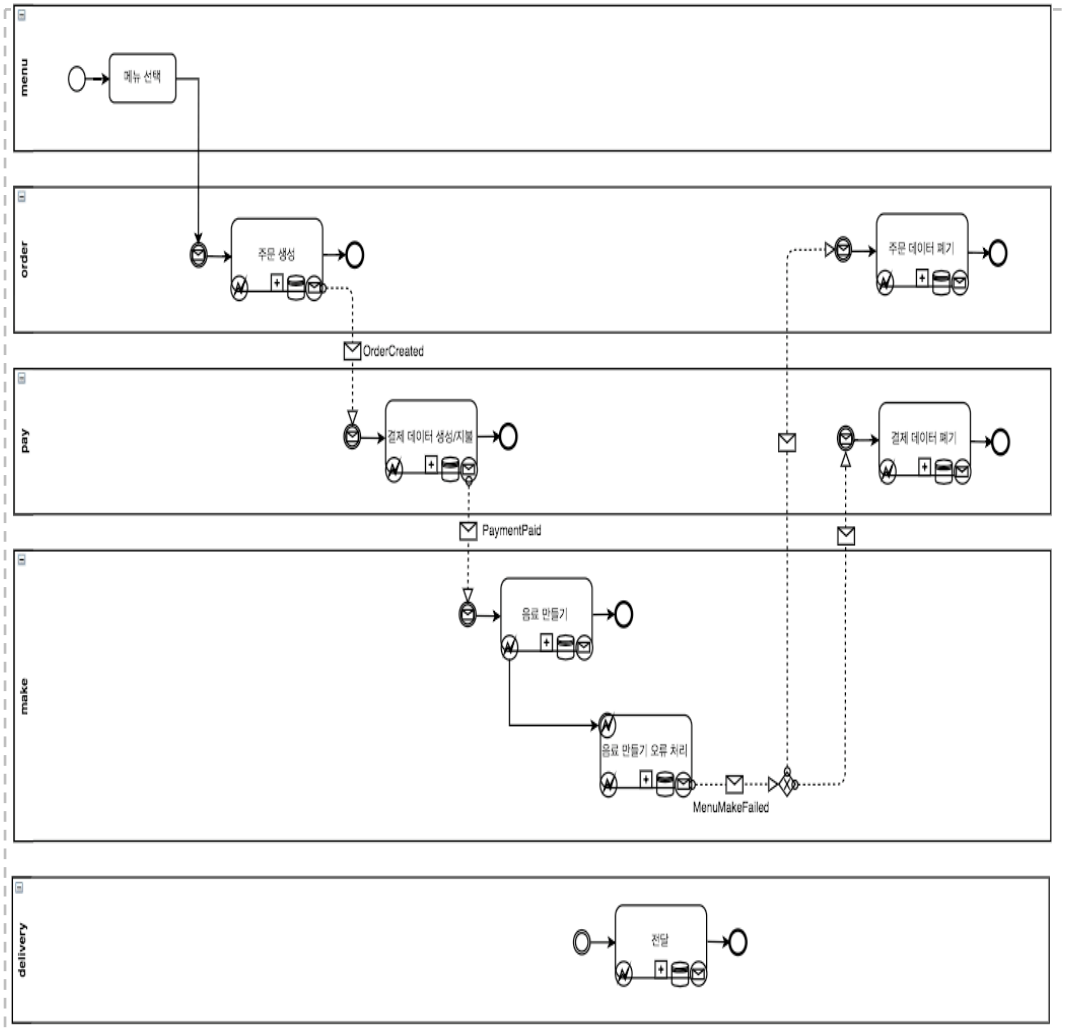
기능은 여러 단계에 걸쳐서 수행됩니다. 각 단계마다 상태 변화를 동반합니다. 이전 단계를 완료하면 그에 반응해서 다음 단계를 수행합니다.



- 메뉴 선택
- 카운터에서 핫초코 주문 ← 상태 변화
- 계산 ← 상태 변화
- 핫초코 만들기 ← 상태 변화
- 핫초코 먹기 ← 상태 변화

분산 트랜잭션 처리

기능은 여러 단계에 걸쳐서 수행됩니다. 각 단계마다 상태 변화를 동반합니다. 이전 단계를 완료하면 그에 반응해서 다음 단계를 수행합니다.

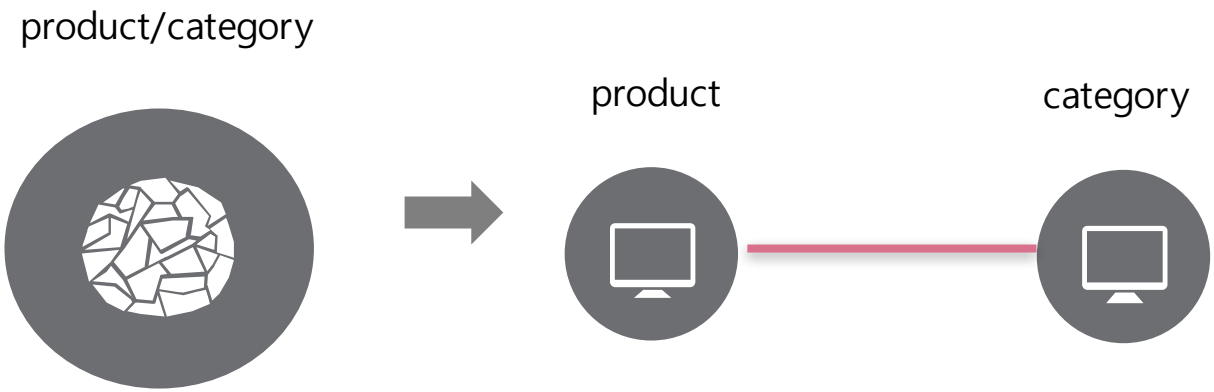


- 메뉴 선택
- 카운터에서 핫초코 주문 ← 상태 변화
- 계산 ← 상태 변화
- 핫초코 재고 부족 ← 오류
- 환불, 주문 폐기 ← rollback

반정규화 데이터의 동기 처리

MSA가 적용된 시스템에서는 **biz.logic**과 이를 수행하는데 필요한 데이터가 서로 다른 서비스에 나뉘어진 경우가 있을 수 있습니다. 이 때 **EDM**을 적용해 서비스 간 데이터 동기 처리를 수행할 수 있습니다.

마이크로서비스 product, category로 구성된 백엔드

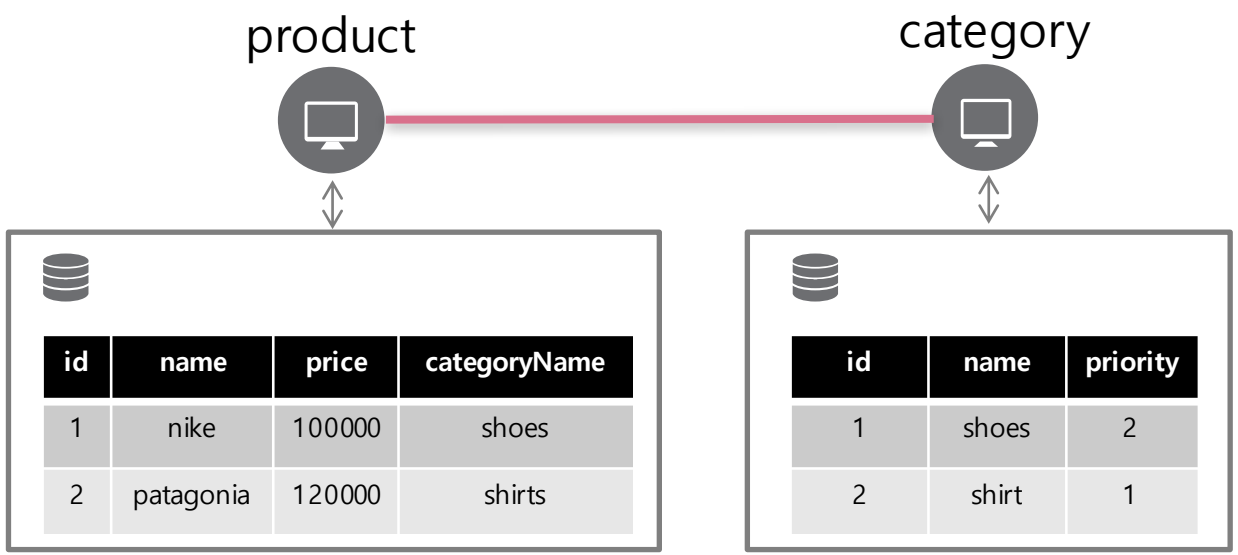


- product는 category의 api를 호출해서 기능 수행
- biz. logic이 복잡해진다던지, 성능 문제가 생긴다던지 등의 반정규화 needs가 발생

반정규화 데이터의 동기 처리

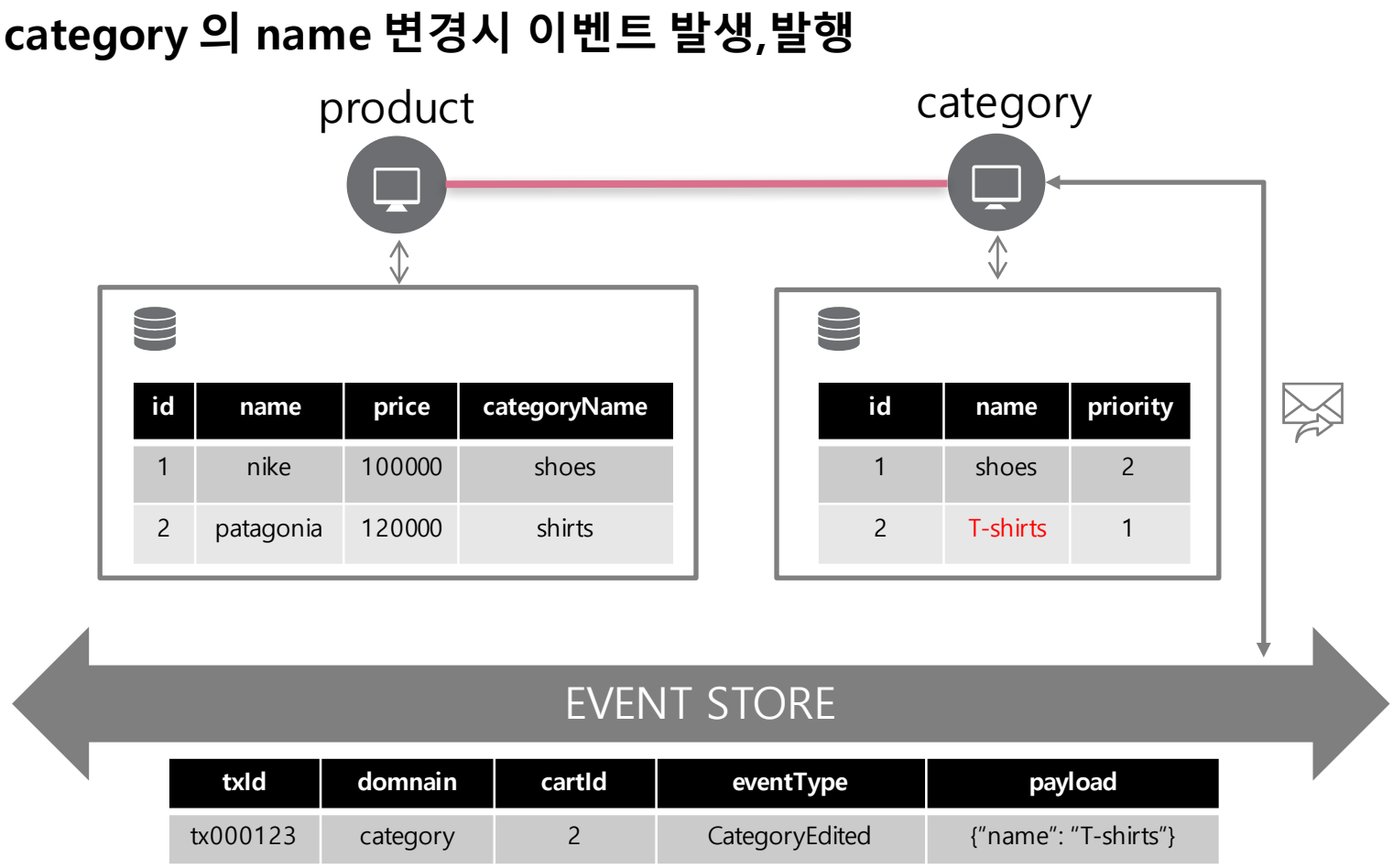
MSA가 적용된 시스템에서는 **biz.logic**과 이를 수행하는데 필요한 데이터가 서로 다른 서비스에 나뉘어진 경우가 있을 수 있습니다. 이 때 **EDM**을 적용해 서비스 간 데이터 동기 처리를 수행할 수 있습니다.

product의 Database에 category의 데이터 categoryName 반정규화



반정규화 데이터의 동기 처리

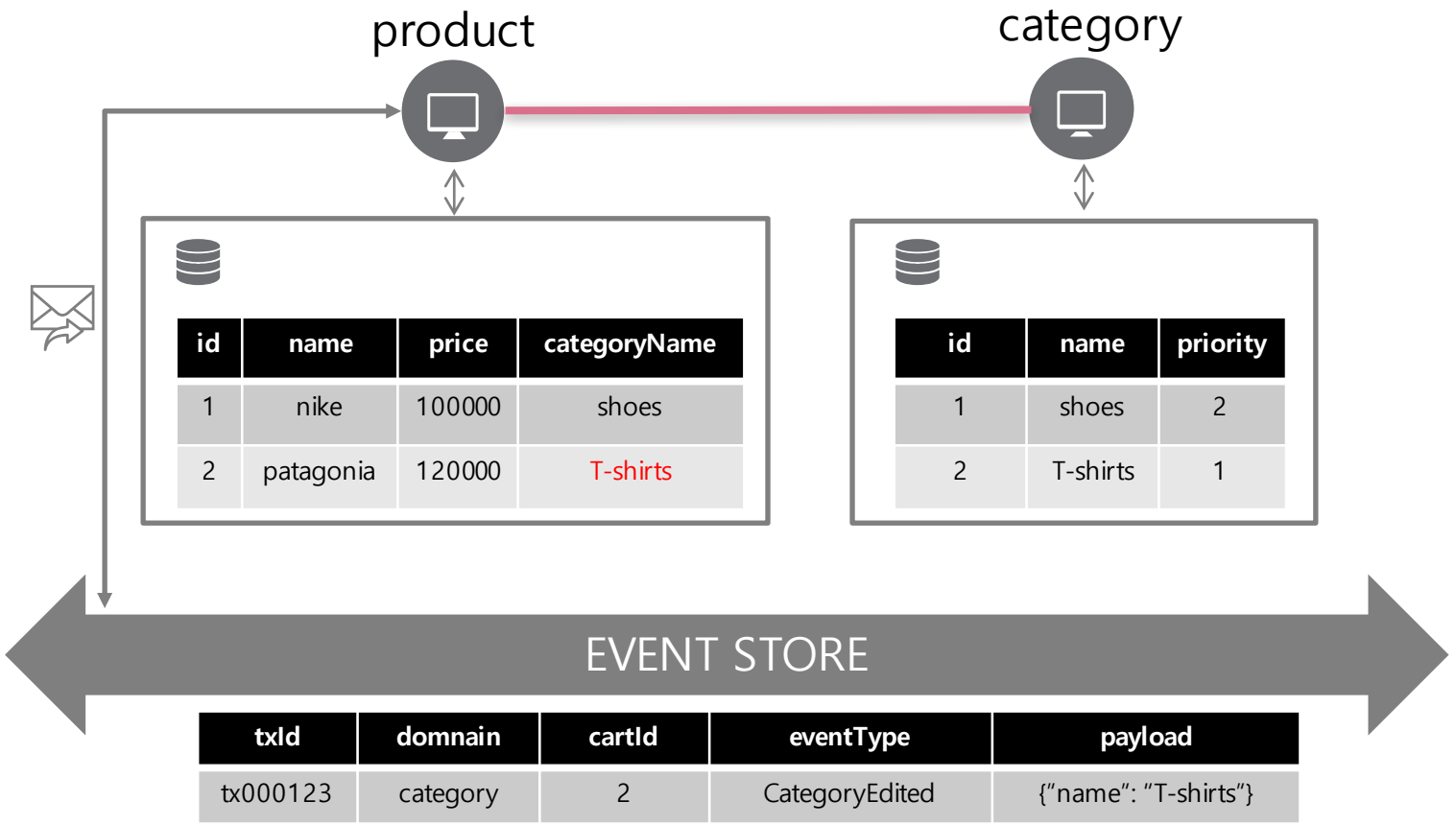
MSA가 적용된 시스템에서는 **biz.logic**과 이를 수행하는데 필요한 데이터가 서로 다른 서비스에 나뉘어진 경우가 있을 수 있습니다. 이 때 **EDM**을 적용해 서비스 간 데이터 동기 처리를 수행할 수 있습니다.



반정규화 데이터의 동기 처리

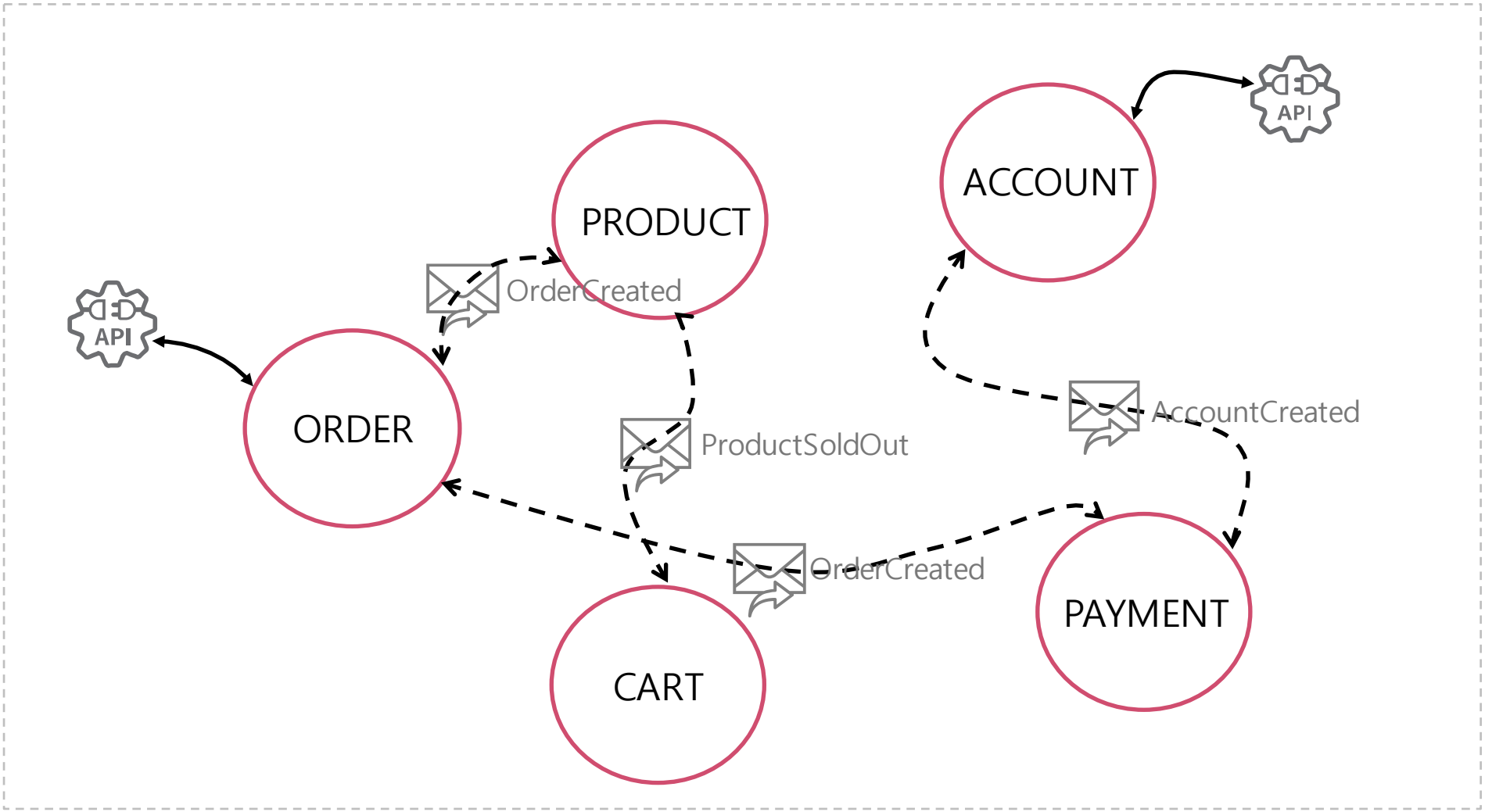
MSA가 적용된 시스템에서는 **biz.logic**과 이를 수행하는데 필요한 데이터가 서로 다른 서비스에 나뉘어진 경우가 있을 수 있습니다. 이 때 **EDM**을 적용해 서비스 간 데이터 동기 처리를 수행할 수 있습니다.

product 에서 CategoryEdited 이벤트를 구독해 categoryName 변경사항 반영



시스템 내 통합(integration)

MSA에서 내부 통신은 크게 2가지가 사용됩니다. 이 중 비동기 메세지 통신을 통해 시스템 내 통합을 적절히 수행합니다.

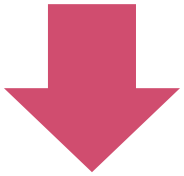


Eventually Consistency

기존의 **DBMS** 활용한 **ACID** 트랜잭션에 따른 데이터 무결성 보장은 **MSA** 에서 서비스, **Database**가 나눠짐으로써 더이상 달성할 수 없습니다. 대신 **EDM**을 통한 데이터의 최종적인 일관성 유지로 변경됩니다.

DBMS Level의 ACID 원칙에 따른 Transaction 처리

All Commit or Rollback



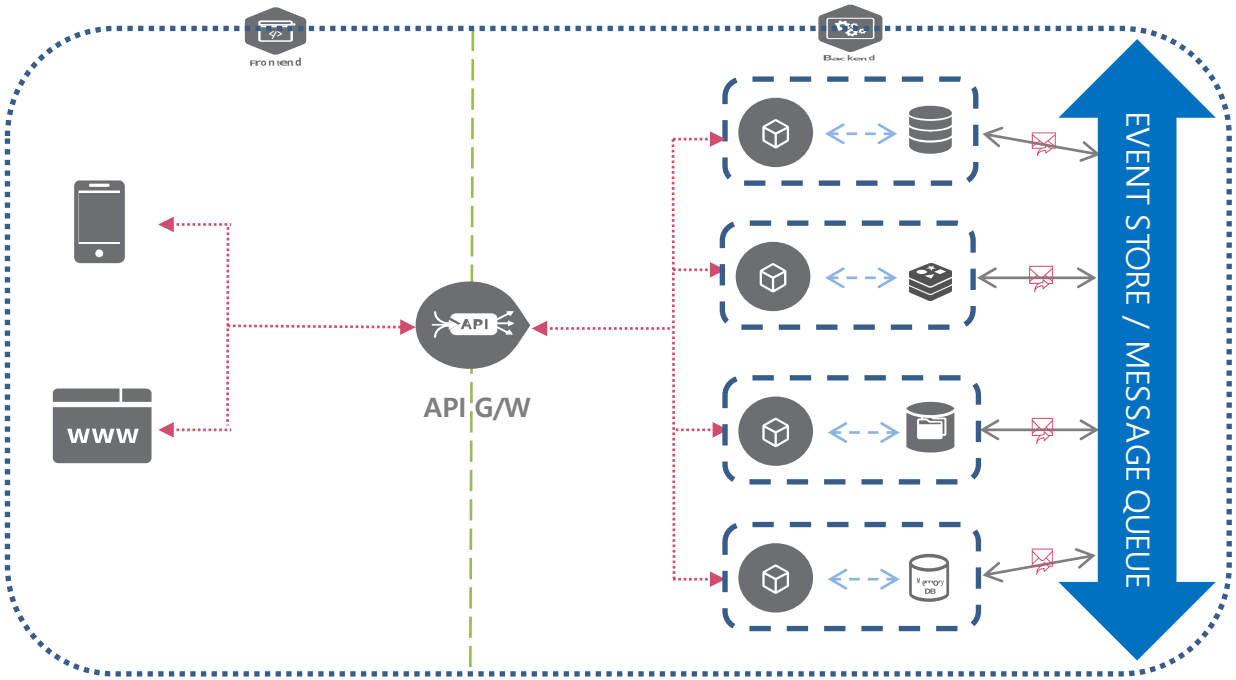
Application Level의 최종적인 일관성 보장

Eventually Consistency

PART 03. Event Driven MicroService 를 적용하는 이유

Event Driven MicroService를 적용하는 이유

Event Driven MicroService를 적용하는 이유는 무엇일까요?



MicroService 에 Event Driven 끼얹기

나뉘다

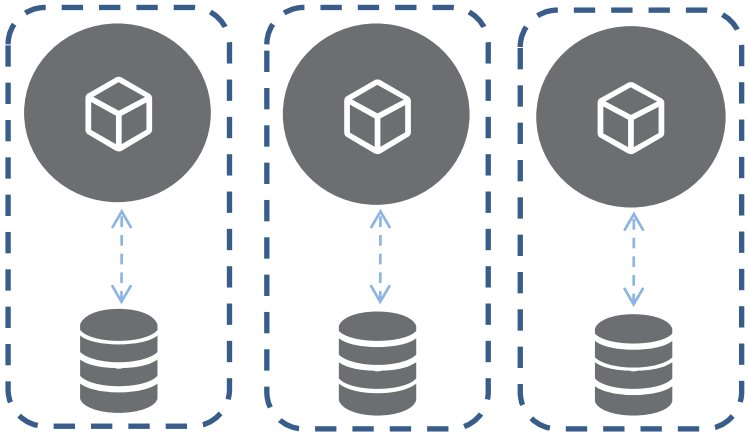
MSA를 적용한 시스템은 서비스가 나뉘고 Database가 나뉘어집니다.

나뉘다

- 서비스 별 자체 biz. Logic 과 데이터
- 서비스 별 최적의 Database

Database Per Service

MSA를 적용한 시스템은 서비스가 나뉘고 Database가 나뉘어집니다.



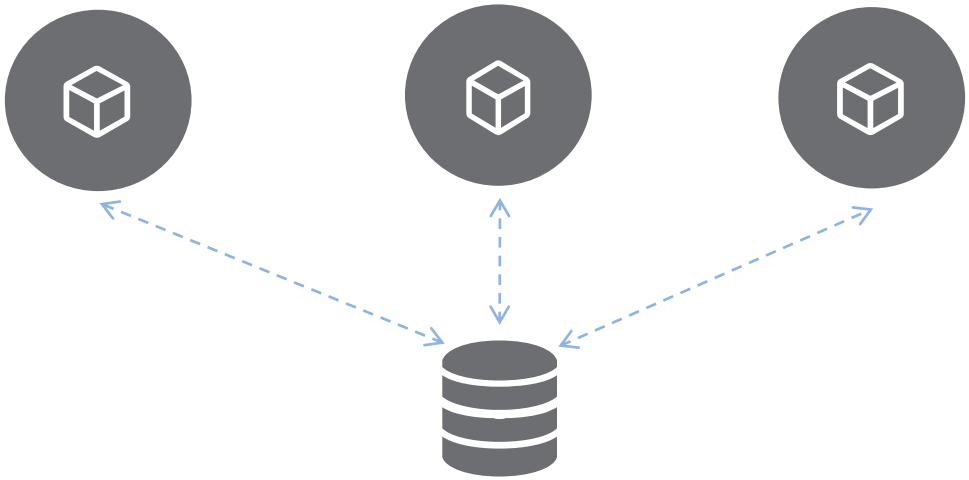
Database Per Service

- 느슨한 결합
- 관심사의 집중
- 폴리글랏 프로그래밍
- 독립적인 배포 주기
- 기타 등등

를 달성하기 위한 **핵심 키워드**

Database Per Service

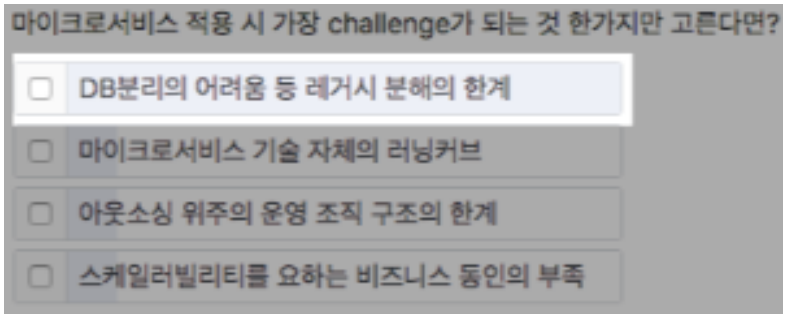
중앙화 **Shared Database**를 사용하는 것은 많은 제약사항이 있습니다.



- 중앙화된 데이터베이스는 시스템의 응집력을 저해하고 종속성을 높인다.
- 단일 트랜잭션 처리에 따라 테이블 락 등 장애 발생 가능성이 있다.
- 데이터베이스 스케일링이 어렵다.
- 중앙화 된 데이터베이스에 장애 발생시 전체 시스템에 장애를 일으킨다.
- 서비스의 특징에 따른 최적의 데이터베이스 선택이 어렵다.

Database Per Service

Database Per Service 는 **MSA**의 느슨한 결합, 관심사의 집중, 폴리글랏 프로그래밍, 독립적인 배포 주기 등을 달성하기 위한 핵심 키워드입니다. 하지만 **MSA**를 적용할 때 **Database Per Service**는 가장 어려운 부분 중 하나입니다.



Database 변경/분리 시 고려 사항

- 엔터프라이즈에서 데이터가 중요한 자산
- 관계형 데이터베이스 장점 사용 불가
  - 데이터를 효율적으로 보관하고 조회/삭제등의 기능을 높이는 장점
  - 테이블 조인을 통한 뷰 제공
  - ACID 원칙에 따른 트랜잭션 기능
- database oriented system
- 데이터베이스 분리시 발생하는 비용
- 기존 DBMS에 최적화된 각종 세팅

Database Per Service

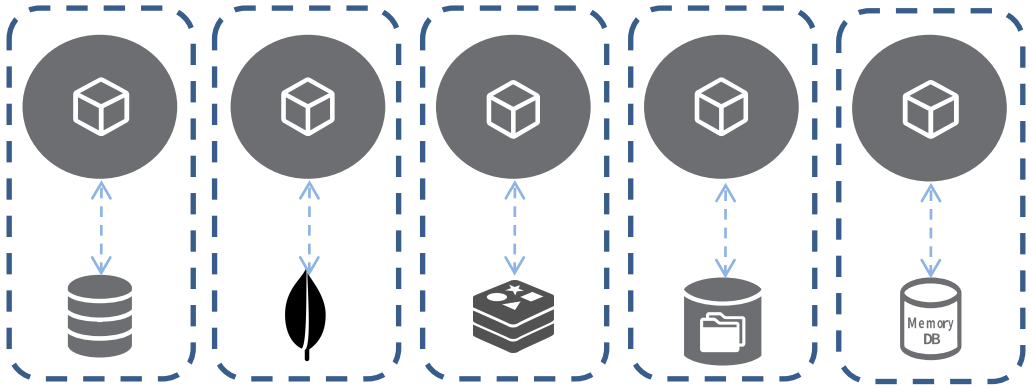
Database Per Service를 적용하면 **Shared Database** 구성의 **DBMS** 레벨에서 제공하던 기능을 **Application** 레벨에서 해결해야 합니다.

Application Level에서 해결해야 하는 요구사항

- 스텝에 따라 처리되는 비즈니스 흐름 수행
- 서비스 간 트랜잭션 처리
- 서비스 간 반정규화 데이터 동기 처리
- 기타 등등

폴리글랏

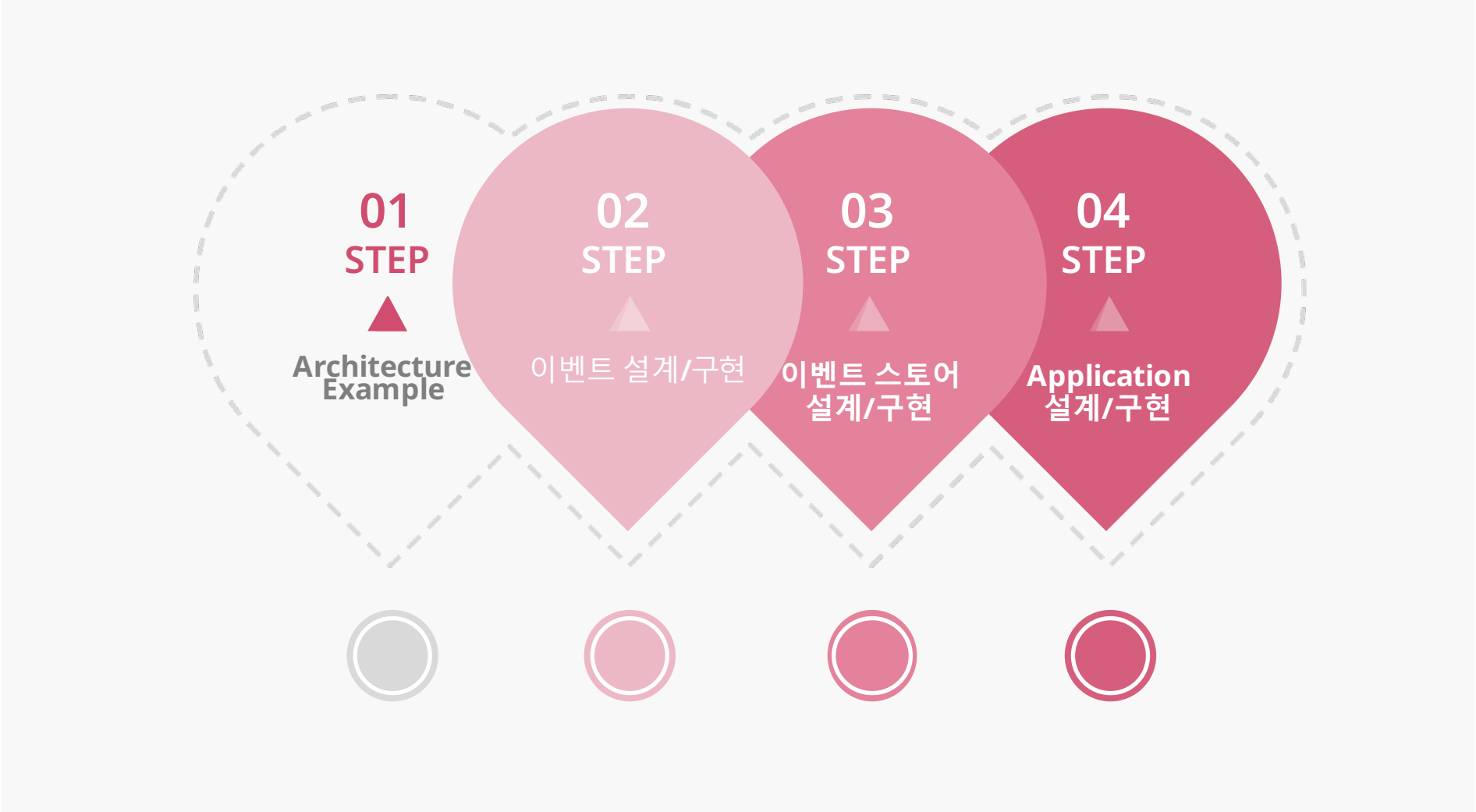
MSA를 적용한 시스템은 서비스가 나뉘고 Database가 나뉘어집니다.



폴리글랏 Needs

- 스트리밍 형태의 비정형 데이터를 다루려는 Needs 등
  - RDB -> NoSQL
- DB 내 데이터 접근 자유 -> 데이터 오너 서비스의 API를 통한 접근
  - 이기종간 Database 선택에 자유

Part 04. Event Driven MicroService 구현시 고려사항



구현

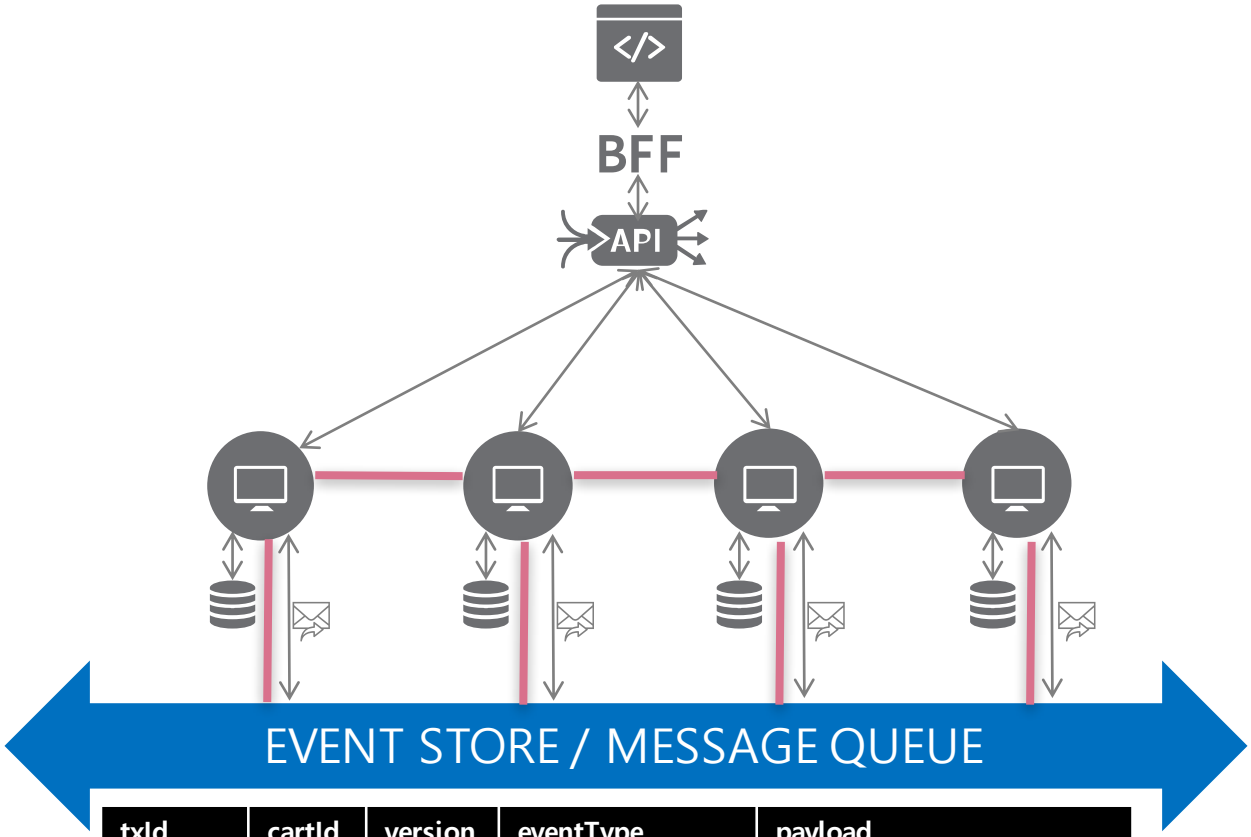
현재 Event Driven MicroService를 구성할 때 널리 알려진 Implements 가 없습니다.

Event Driven을 구현에 필요한 Insight를 도출

Banking Service + Application 직접 적용

Outer Architecture Example

Outer Architecture 예시를 알아봅시다.



txId	cartId	version	eventType	payload
tx000123	b859	1	cartItemAdded	{"itemID": "C"}
tx000145	b859	2	cartItemRemoved	{"itemID": "C"}

Insight – 이벤트 설계/구현

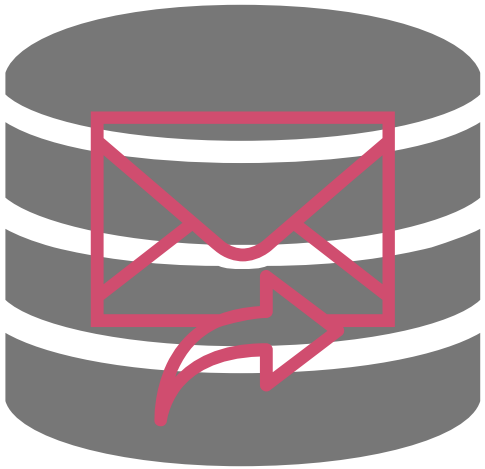
지금까지의 내용을 바탕으로 **EDM**을 구현하기 위해 기존 방식에서 추가로 이벤트 설계를 고려해야 합니다.

업무에 기반한 의미있는 이벤트 설계

- 정상 시나리오에서 이벤트 흐름 도출
- 장애 발생시 retry/rollback 여부에 따른 이벤트 흐름 도출
- 이벤트 흐름 시각화 방법
- Scaling된 Application 간 경쟁적인 event msg 구독

Insight – 이벤트 스토어 설계/구현

지금까지의 내용을 바탕으로 **EDM**을 구현하기 위해 기존 방식에서 추가로 고려해야 하는 사항들이 있습니다.



이벤트 스토어 설계/구현

- Application + Backing Service
- 요구사항 별 별도의 뷰 제공
- 메세지 큐 기능
- exactly once 보장
- dlq 기능
- 고가용성
- 내결함성
- throughput
- event message persistent
- retry 설정
- time out 설정
- 이벤트 라우팅 룰/전략
- 확장성

유용한 Backing Service

- NoSQL
- 관계형 Database
- RabbitMQ
- Kafka
- Geode
- CSP – MQ 서비스
- 기타 등등

Insight – Application 설계/구현

지금까지의 내용을 바탕으로 **EDM**을 구현하기 위해 기존 방식에서 추가로 고려해야 하는 사항들이 있습니다.

Application

- mvc + pub/sub
  - project package 구조 정의
  - event 보관/발행을 위한 기능 설계/구현
- event msg 설계/구현
  - Application 별 event type 도출
  - event type 에 대응되는 fail event type도출
  - event 멱등 처리 고려
- 이벤트 생성+발행 logic 설계/구현
  - 데이터의 생성,변경,삭제와 이벤트 생성+발행의 일관성 유지 필요
- 보상 트랜잭션
  - 오류 발생시 fail 이벤트 생성+발행 logic 설계/구현
  - transactionId 생성,전달 방법 설계/구현

구현

Event Driven MicroService의 요구사항 및 달성 효과는 전혀 새로운 것이 아닙니다.

Event Driven MicroService와 비슷한 문제를 다루는 Case

- DBMS 내 트랜잭션 로그를 활용한 redo/undo 로직
- 이기종 Database 간 분산 트랜잭션 처리
- 외부 시스템과 I/F를 통해 업무를 처리할 때 상이한 시스템 간 정합성 처리

감사합니다