DEVIEW 2021

데이터 드리븐으로 서비스 운영 100% 자동화하기

로깅 데이터와 카프카를 활용한 만능 이벤트 프로세싱 아키텍처

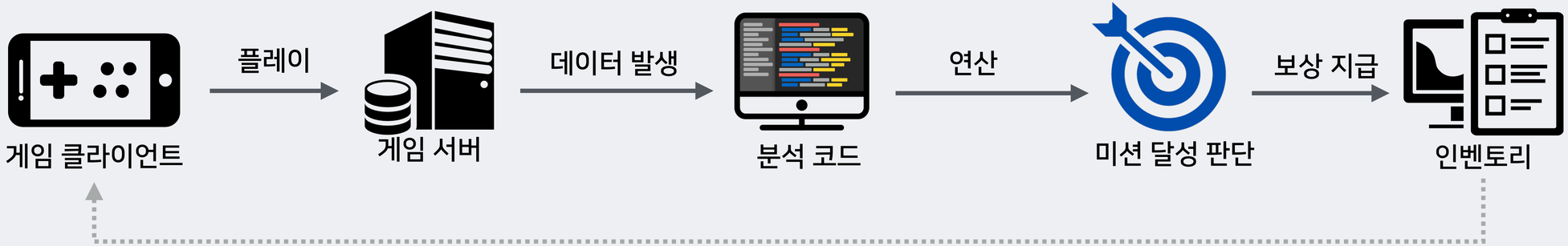
이재은 데브시스터즈

CONTENTS

1.쿠키런:킹덤에서 레벨 3을 달성하면 보석 100개를 드립니다

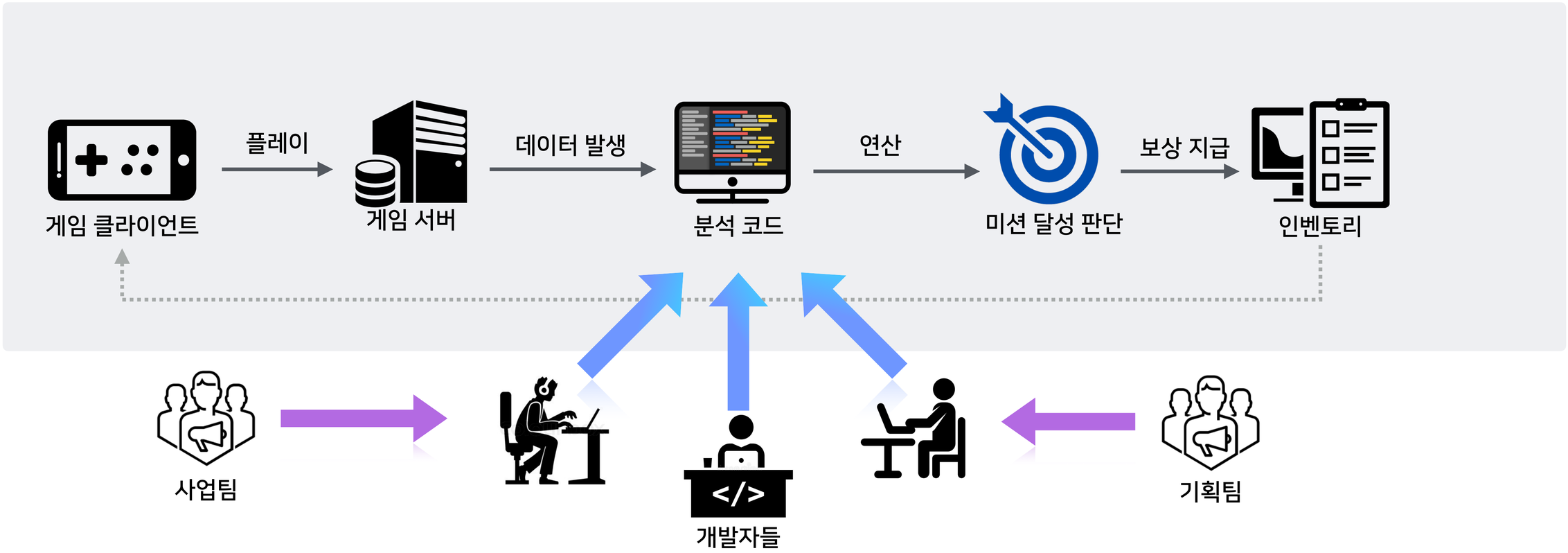
1.1 게임 내 보상 지급 시스템의 구성

게임 서버에서 발생한 데이터를 분석 코드로 연산 ➡ 미션 달성 판단시 보상 재화를 인벤토리로 지급



1.1 게임 내 보상 지급 시스템의 구성

게임 서버에서 발생한 데이터를 분석 코드로 연산 ➡ 미션 달성 판단시 보상 재화를 인벤토리로 지급



1.2 보상 지급 처리 자동화의 어려움

문제 1)

보상 조건의 다양화

문제 2)

자동화에 필요한 작업 범위의 광범위함

1.2 보상 지급 처리 자동화의 어려움

DEVIEW 2021

문제 1) 보상 조건의 다양화

- 로그인 횟수, 레벨 달성, 특정 건물 건축, PVP 승리 N 회 등..

레벨 3을 달성하면 보석 100개를 드려요

로그인을 할 때마다 하트를 10개씩 드려요

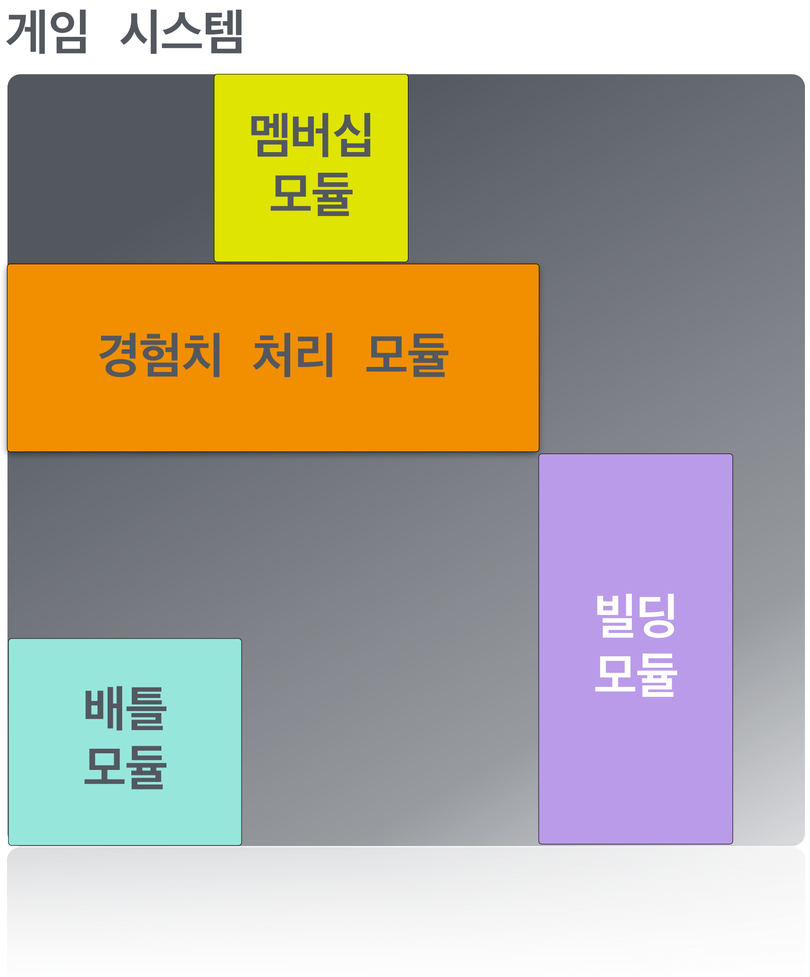
제빵소 3개를 지으면 고기 젤리 5개를 드려요

PVP 5번을 승리하면 보석 30개를 드립니다

1.2 보상 지급 처리의 어려움

DEVIEW 2021

문제 2) 자동화에 필요한 작업 범위의 광범위함



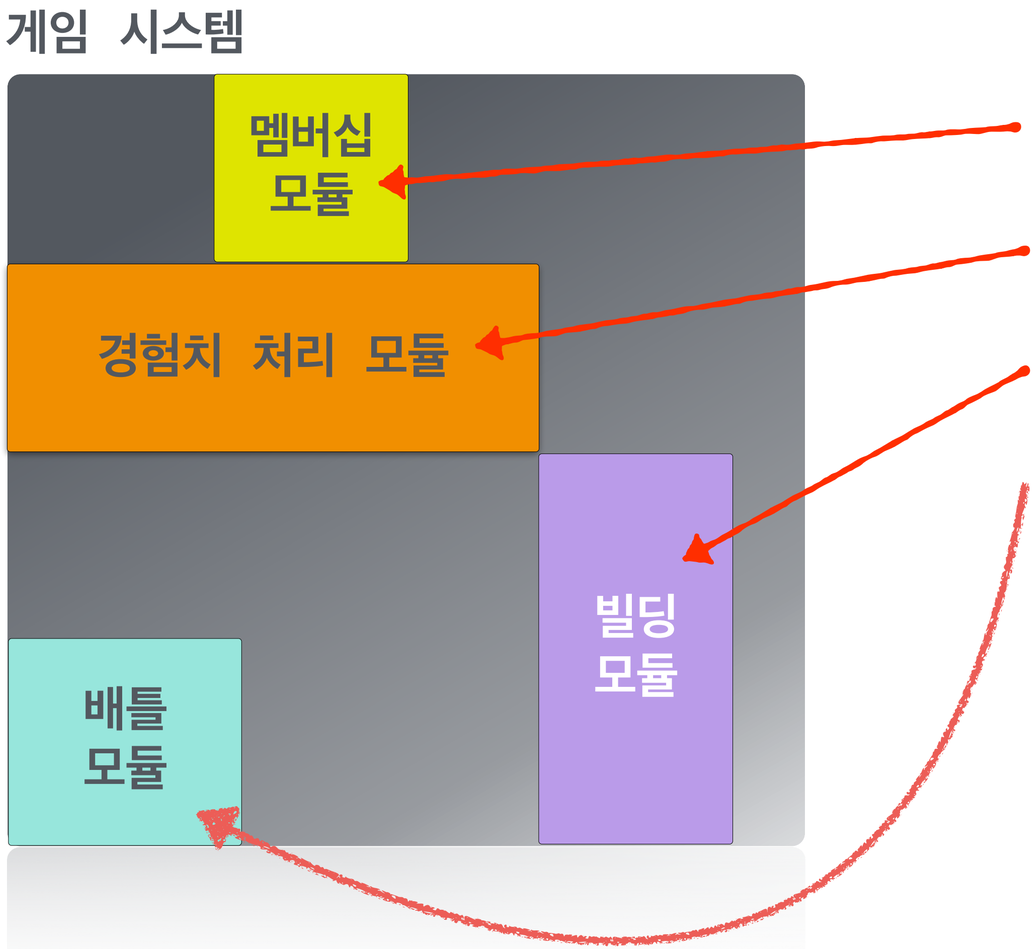
?

- 로그인 보상
- 레벨업 보상
- 건물 짓기 보상
- PVP 보상

1.2 보상 지급 처리의 어려움

DEVIEW 2021

문제 2) 자동화에 필요한 작업 범위의 광범위함



- 로그인 보상은 멤버십 모듈에서 담당
- 레벨업 보상은 경험치 모듈에서 담당
- 건물 짓기 보상은 빌딩 모듈에서 담당
- PVP 보상은 배틀 모듈에서 담당

!!자동화 가능한 공통 모듈이 없음!!

게임 시스템에 국한된 문제는 아님

자동화 하기 어려운 것은 분명하다

그러나
정말 방법이 없을까?

2.로그 데이터에 주목하기

2.1 로그 데이터

일반적으로 로그 데이터는 개발자 임의로 정의 후 사용

- 버그 발생시 디버깅에 사용하기 위해서
- CS 처리에 활용하기 위해서
- 서비스 메트릭으로 활용하기 위해서

형식이 통일되어 있지 않거나, 누락되어도 크게 문제되지 않음

2.2 고품질 로그 데이터

믿을만한 로그 데이터를 보유하기 위해 필요한 것들

1. 로그 데이터를 위한 스키마 정의 및 준수
2. QA 검수 시 로그 데이터 검수 단계를 포함
3. 개발 단계에서 로그 데이터 고려
4. 로그 수집 단계에서 자동화된 로그 스키마 검수 및 변환

2.3 데브시스터즈의 로깅 플랫폼

2.4 로그 데이터의 활용

DEVIEW 2021

대부분의 보상 미션 토폴로지의 로그 데이터는 일반화 가능

```
레벨 3을 달성하면 보석 100개를 드려요
```

```
로그인을 할 때마다 하트를 10개씩 드려요
```

```
제빵소 3개를 지으면 고기 젤리 5개를 드려요
```

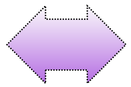
```
PVP 5번을 승리하면 보석 30개를 드립니다
```

2.4 로그 데이터의 활용

DEVIEW 2021

대부분의 보상 미션 토폴로지의 로그 데이터는 일반화 가능

레벨 3을 달성하면 보석 100개를 드려요



A를 N번 달성하면 B를 M개 보상

로그인을 할 때마다 하트를 10개씩 드려요

제빵소 3개를 지으면 고기 젤리 5개를 드려요

PVP 5번을 승리하면 보석 30개를 드립니다

2.4 로그 데이터의 활용

대부분의 보상 미션 토폴로지의 로그 데이터는 일반화 가능

레벨 3을 달성하면 보석 100개를 드려요	⇔	A를 N번 달성하면 B를 M개 보상
로그인을 할 때마다 하트를 10개씩 드려요	⇔	A를 N번 달성하면 B를 M개 보상
제빵소 3개를 지으면 고기 젤리 5개를 드려요	⇔	A를 N번 달성하면 B를 M개 보상
PVP 5번을 승리하면 보석 30개를 드립니다	⇔	A를 N번 달성하면 B를 M개 보상

Event x N ⇨ Reward x M

대다수의 토폴로지가 일관된 형태로 수렴

2.4 로그 데이터의 활용

DEVIEW 2021

JSON 데이터셋은 비교 연산의 결합으로 전환할 수 있다

레벨 3을 달성하면 보석 100개를 드려요

```
{
  game_id : “kingdom”
  event : “level_up”
  properties : {
    level : 3
    player_id : “G01-54785”
  }
}
```

```
game_id == “kingdom”
+
event == “level_up”
+
level == 3
```

결국 스키마가 강제된 로그 쿼리 데이터셋은
조건 처리 연산에 사용 가능하다

그렇다면 자동화된 시스템을 만들 수 있지 않을까

1. 로깅 데이터 기반으로 동작
2. 준실시간 이벤트 프로세싱
3. 대규모 트래픽을 커버
4. 동적 이벤트 태스킹
5. 이벤트 태스크 스케줄링 자동화 가능

CEP Application

(Complexed Event Processing)

CEP Application + Dynamic

= DCEP

~~디셉티콘?~~

3.DCEP을 구현하기 위한 기술

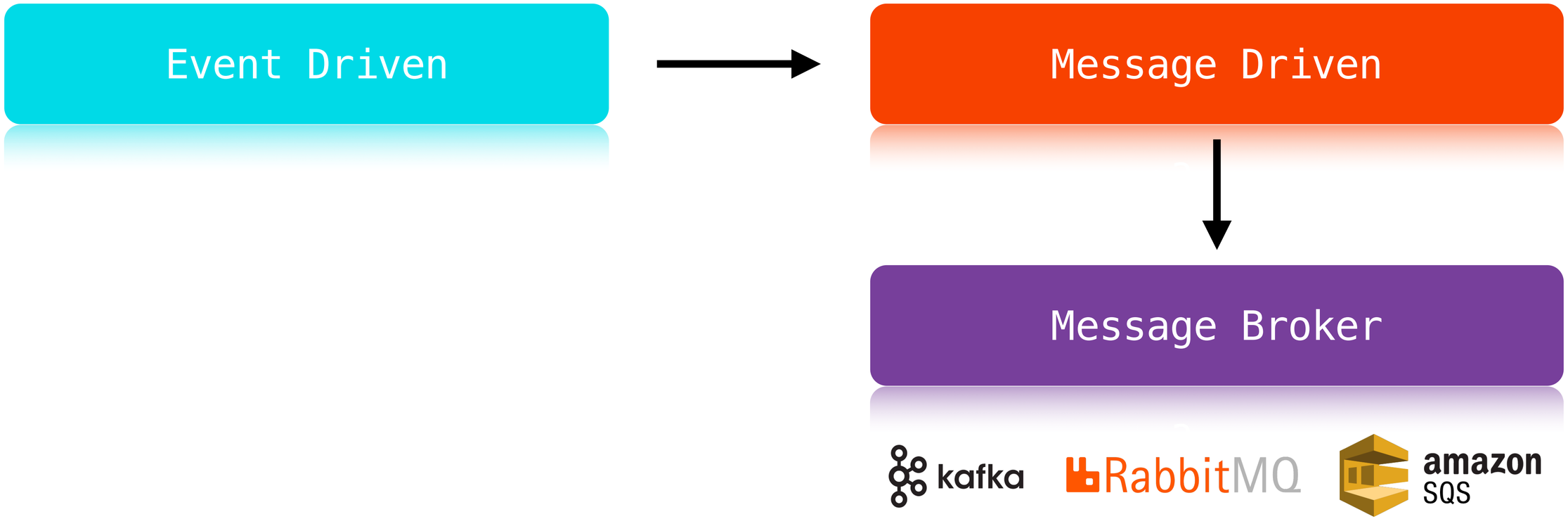
이벤트 드리븐 아키텍처와 스트림 프로세싱 기술이 필요

3.1 Event Driven Architecture

DEVIEW 2021

이벤트 드리븐 패턴을 구현한 분산 서비스 아키텍처

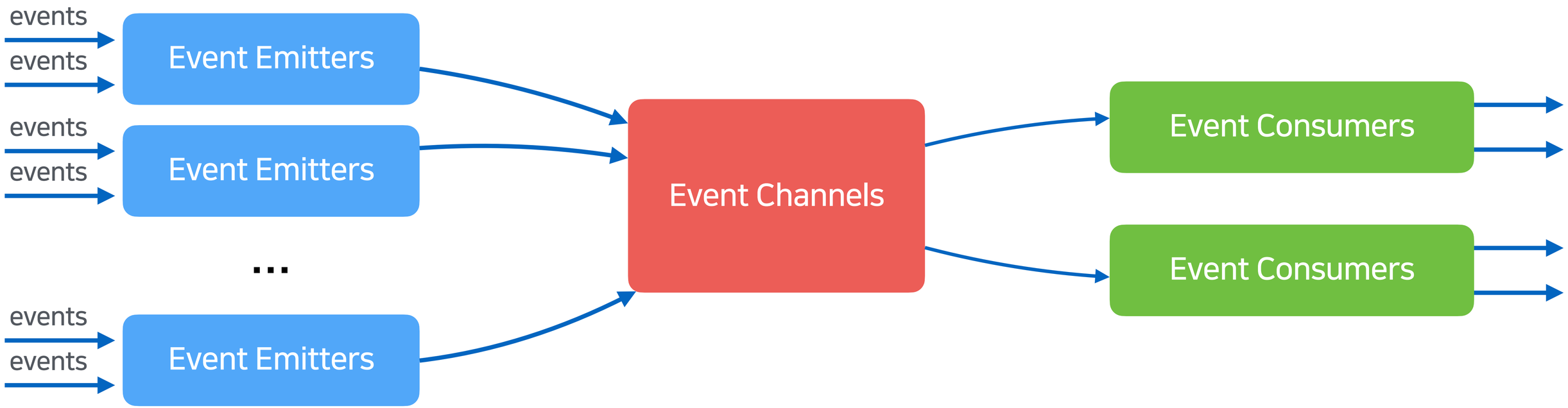
프로세스가 자동/순서에 따라 배치 처리되는 것이 아니라, 특정 이벤트의 반응으로 동작



3.1 Event Driven Architecture

DEVIEW 2021

EDA의 세 가지 구성 요소



3.1 Event Driven Architecture

DEVIEW 2021

수신한 이벤트 처리 방식은 4가지 타입으로 구분

SEP(Simple event processing)

ESP(Event streaming processing)

CEP(Complexed event processing)

OLEP(Online event processing)

3.2 스트림 프로세싱(Stream Processing)

DEVIEW 2021

스트리밍 프로세싱 연산이 배치 연산과 다른 점

- 입력이 무한히 이어짐
- 전체 실행 시간은 중요하지 않음
- 빠른 속도로 이벤트를 처리하고, 빠르게 계산 결과를 제공해야 함
- Latency(지연율)와 Throughput(처리율)이 중요한 메트릭

3.2 스트림 프로세싱(Stream Processing)

스트림 프로세싱을 지원하는 기술들

AWS Kinesis

AWS에서 제공하는 매니지드 서비스
메시지 브로커를 관리할 필요가 없지만 비용 발생

클로저 기반의 분산 스트림 프로세싱 연산 프레임워크
클러스터링을 통한 수평 확장 가능 + Fault Tolerancy
At least once 지원

Apache Storm

Spark Streaming

카프카, Flume, Kinesis 등에서 실시간 스트리밍 처리 지원
Streaming Context를 이용한 스트리밍 처리 시작, 종료 등을 수행

3.2 스트림 프로세싱(Stream Processing)

스트림 프로세싱을 지원하는 기술들

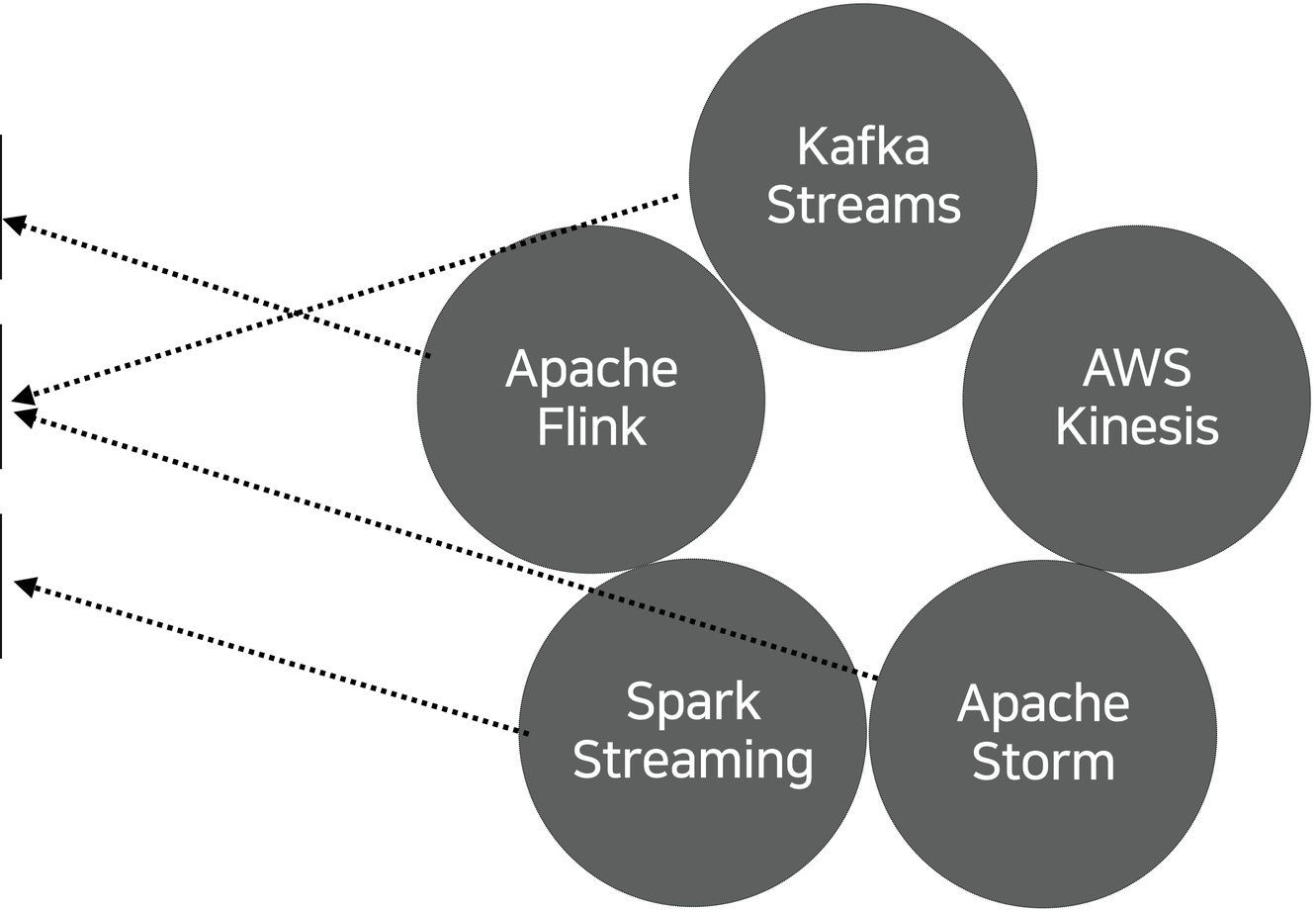
Scala로 개발된 분산 스트림 프로세싱 프레임워크
Exactly Once 레벨까지 지원

Apache Kafka 프로젝트 내에 포함된 이벤트 스트리밍 플랫폼
카프카와 통합되어 있어 안정적인 스트리밍 처리 가능

3.2 스트림 프로세싱(Stream Processing)

스트림 프로세싱을 지원하는 기술들

- Flink의 태스크 매니징 컴포넌트 분산 구조
- Kinesis, Storm의 토폴로지 확장 메커니즘
- Spark Streaming의 Spark Context



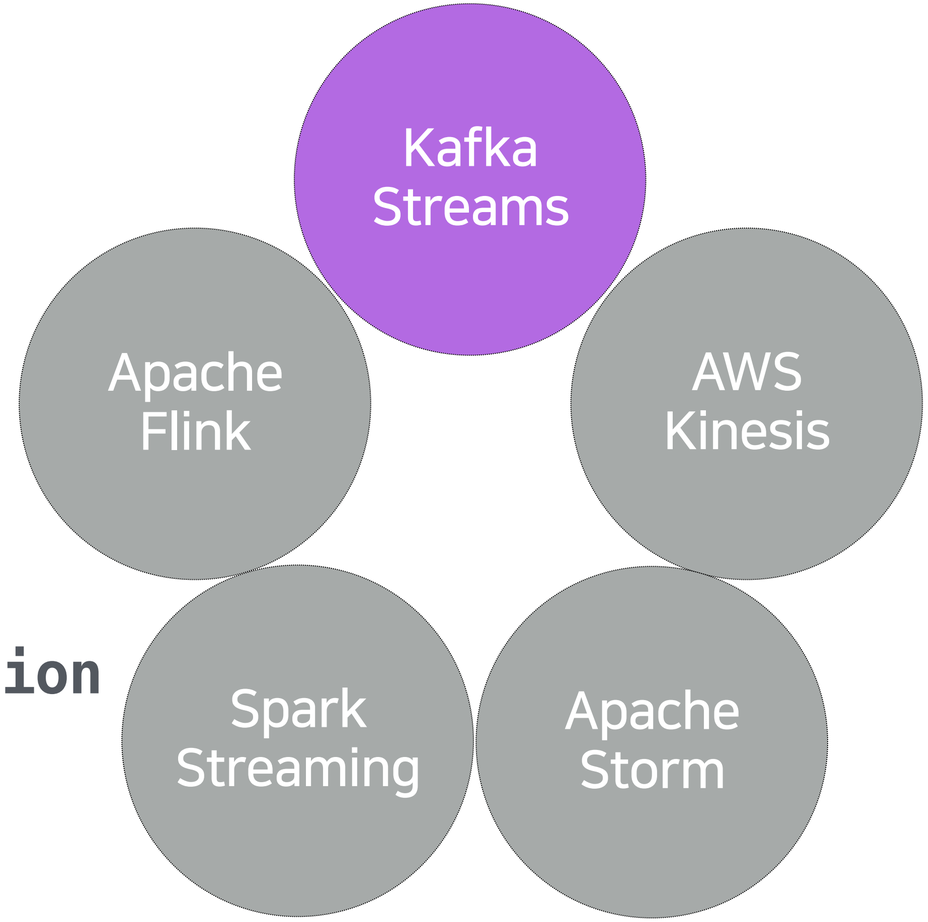
3.2 스트림 프로세싱(Stream Processing)

NDEVIEW 2021

각 스트림 처리 기술의 주요 부분을 결합한 프로세싱 시스템

토폴로지 확장
+ 태스크 매니징 컴포넌트 분산 구조,
+ Spark Context
+ Kafka Streams

⇨ **Dynamic Complexed Event Processing Application**



4.쿠버네티스에서 DCEP 애플리케이션 구현하기

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

- *Schedule Manager*
- *Job Manager*
- *Event Processing Manager (Worker)*
- *Dispatcher API*
- *Event Context*

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

Dispatcher API

- 어드민 콘솔의 입력을 받는 API 서버
- Event Context를 정의하기 위한 데이터 관리
- 이벤트 프로세싱 스케줄을 생성
- 스케줄러 매니저와 상호작용

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

Schedule Manager

- 운영자가 등록한 이벤트 플랜에 따라 스케줄을 관리
- 정해진 시각에 Job Manager를 호출하여 해당 이벤트 실행을 요청
- 시간이 완료되면 스트림 애플리케이션의 종료 요청을 수행

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

Job Manager

- 이벤트 프로세싱 매니저 실행을 제어하는 마스터 프로세스
- 스케줄 매니저로부터 요청을 받아 Kube API를 통해 워커 노드 생성
- 이벤트 세부 정보를 렌더링하여 Event Context 생성
- 워커 노드의 사이즈 및 스트림 소스 결정

4.1 DCEP 애플리케이션의 구성요소

Event Processing Manager

- 이벤트 스트림 프로세싱 컴포넌트
- 워커(Worker) 그룹 - 워커의 구조로 구성
- 이벤트와 워커 그룹은 1:1 매칭 관계
- 여러 개의 이벤트 프로세스가 스케줄에 따라 동시 구동
- 카프카 파티셔닝으로 분산된 스트림을 프로세싱

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

Event Context

이벤트를 트리거하기 위해 필요한 내용을 JSON / YAML로 작성

주요 포함 내용

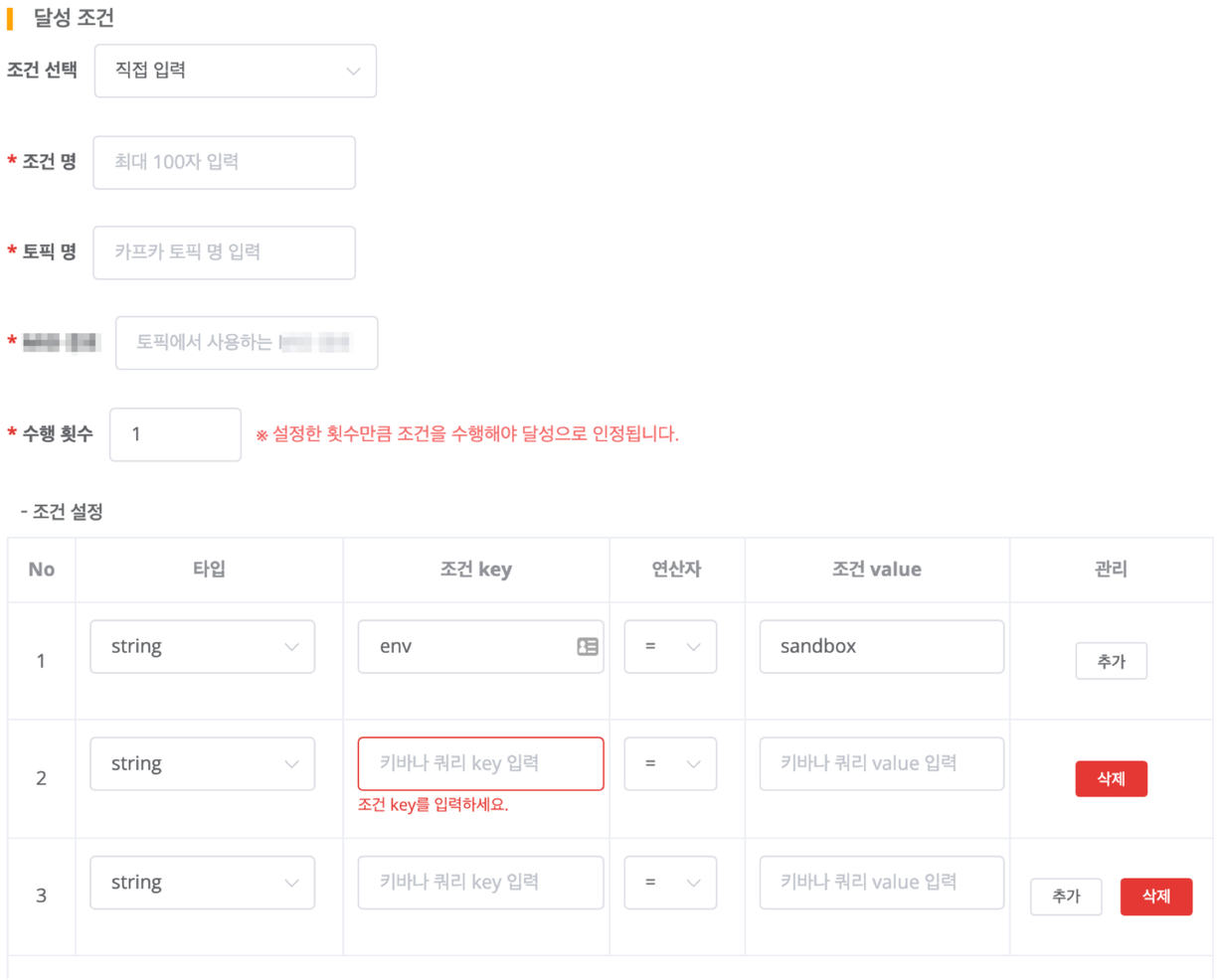
1. 스트림 토픽
2. KeyedStream 연산 대상 필드 선정
3. 이벤트 내용 정의
4. Type Information 처리 정보

```
{
  "event_list": [{
    "topic": "sample-access",
    "conditions": [
        { "field": "event",    "value": "kill"},
        { "field": "gameCode", "value": 1302 },
        { "field": "useSkill", "value": true }
    ],
    "count": 3,
    "keyed": "member.playerId"
  }]
}
```

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

Event Context 등록화면 예시



4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

Event Context 예시

```
{
  "event_list": [{
   "topic": "sample-access",
   "conditions": [
      { "field": "event",    "value": "kill", "type": "string",  "operator": "equals" },
      { "field": "gameCode", "value": 1302,   "type": "integer", "operator": "equals" },
      { "field": "useSkill", "value": true,   "type": "bool",    "operator": "equals" }
   ],
   "count": 3,
   "keyed": "member.playerId"
  }]
}
```

4.1 DCEP 애플리케이션의 구성요소

DEVIEW 2021

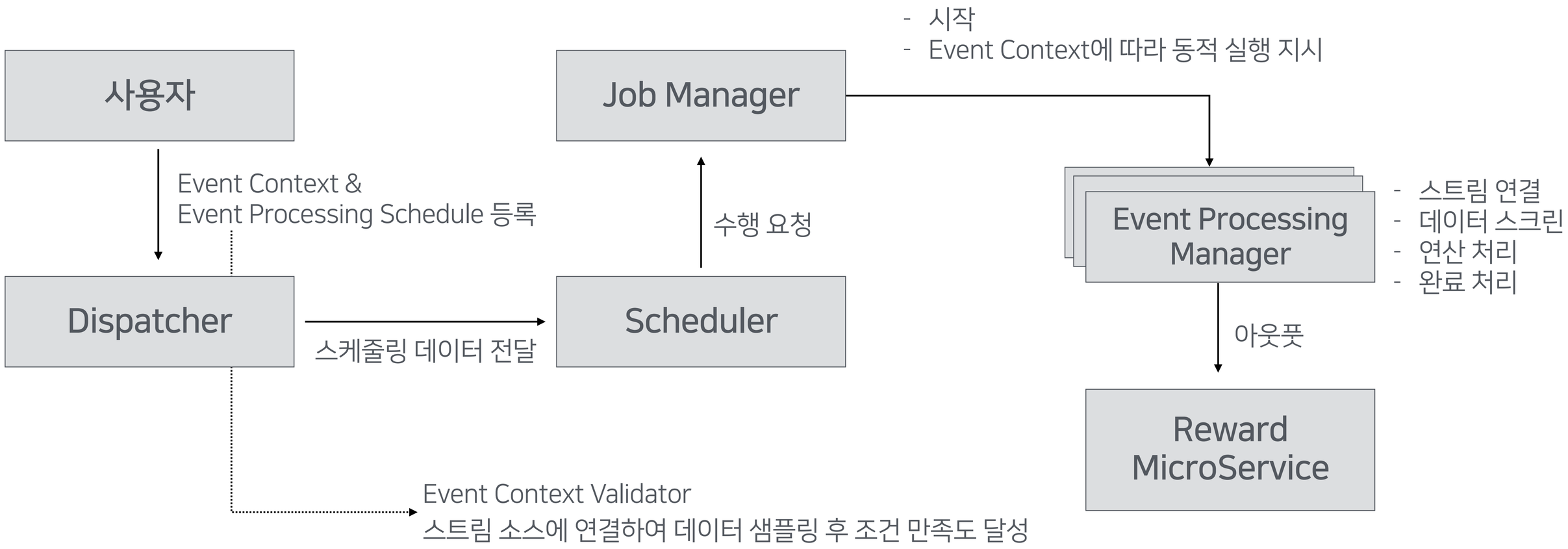
Event Context 예시

```
{
  "event_list": [{
   "topic": "sample-access",
   "conditions": [
      { "field": "event",    "value": "kill", "type": "string",  "operator": "equals" },
      { "field": "gameCode", "value": 1302,   "type": "integer", "operator": "equals" },
      { "field": "useSkill", "value": true,   "type": "bool",    "operator": "equals" }
   ],
   "count": 3,
   "keyed": "member.playerId"
  }]
}
```

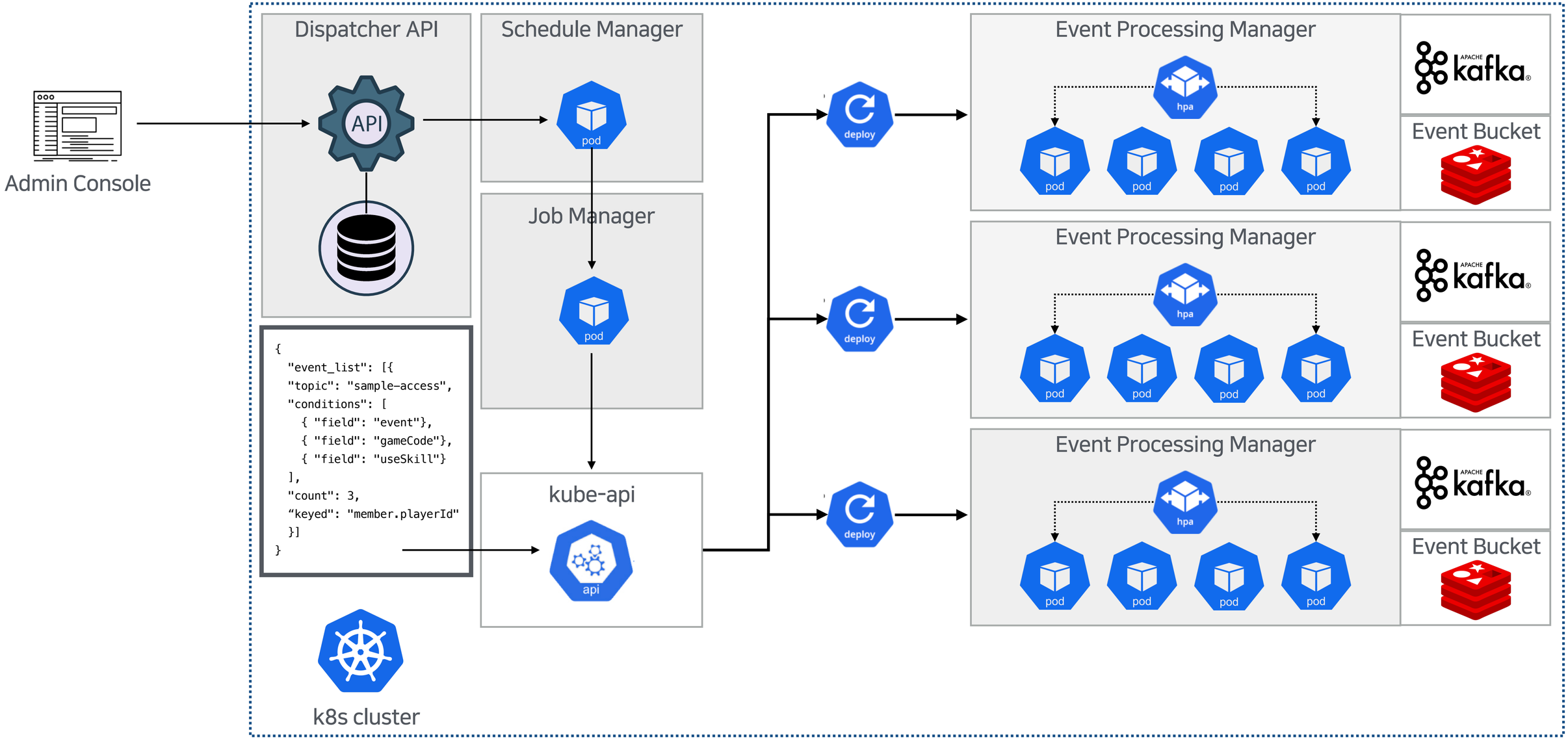
“1302 게임에서 스킬을 사용하여 적을 3번 잡으면 보상”

4.1 DCEP 애플리케이션의 구성요소

Event Processing Sequence



4.1 DCEP 애플리케이션의 구성요소



4.2 DCEP 애플리케이션의 핵심 기능

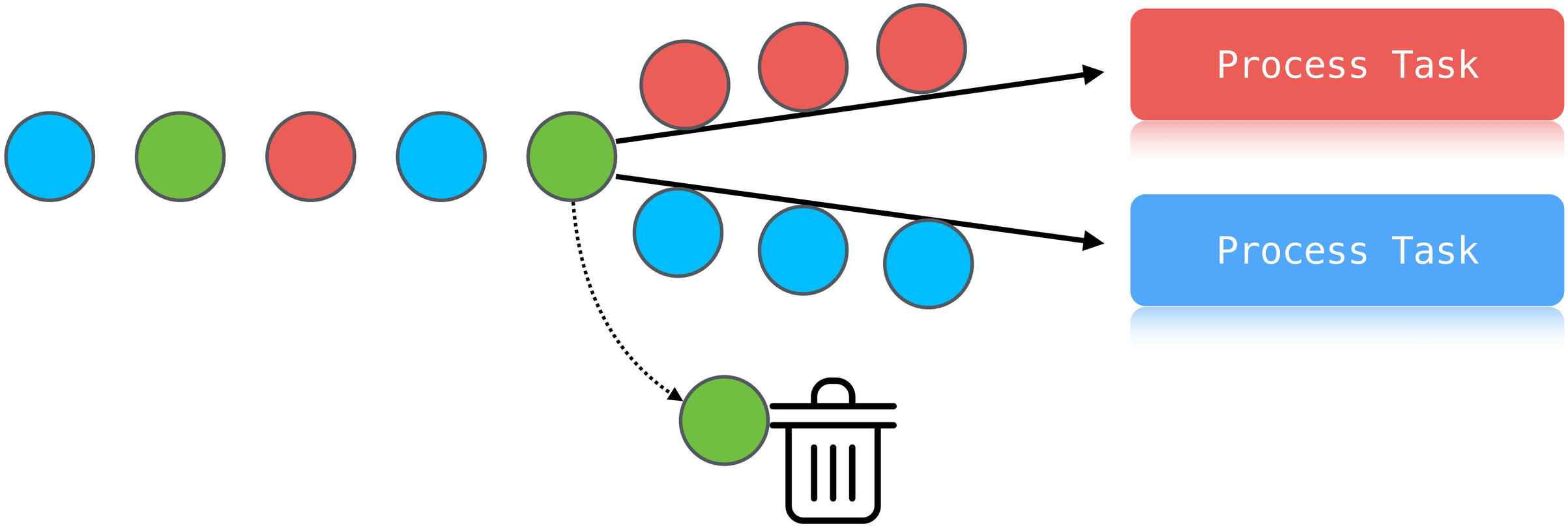
- KeyedStream 변환 연산
- Flexible Topology
- 윈도우 연산(Time window, Count window)
- 스트림 리플레이(Stream Replay)
- Aggregation & Output

4.2 DCEP 애플리케이션의 핵심 기능

DEVIEW 2021

KeyedStream 변환 연산

특정 속성을 공유하는 이벤트를 그룹핑하는 연산



4.2 DCEP 애플리케이션의 핵심 기능

Flexible Topology

토폴로지 모델링 : 모든 조건은 일반화를 거치면 IF ~ THEN 형태로 변환

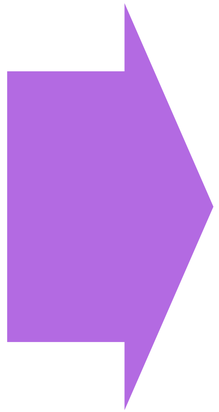
특정 이벤트를 N회 수행했다 = 특정 로그가 N회 발견되었다

특정 로그가 N번 발견되면 매핑된 프로시저를 호출한다.

A 로그가 N번 발견

B 로그가 M번 발견

C 로그가 K번 발견



매핑된 프로시저를 호출한다.

가장 중요한 것은 조건의 일반화

4.2 DCEP 애플리케이션의 핵심 기능

Flexible Topology

로그 데이터 기반 동적 토폴로지 설계

- 레벨 3를 달성한 킹덤 유저에게 다이아 100개를 지급한다

```
{
  game_id : “kingdom”
  event : “level_up”
  properties : {
    level : 3
    play_id : “G01-54785”
  }
}
```

`game_id == “kingdom”`

AND

`event == “level_up”`

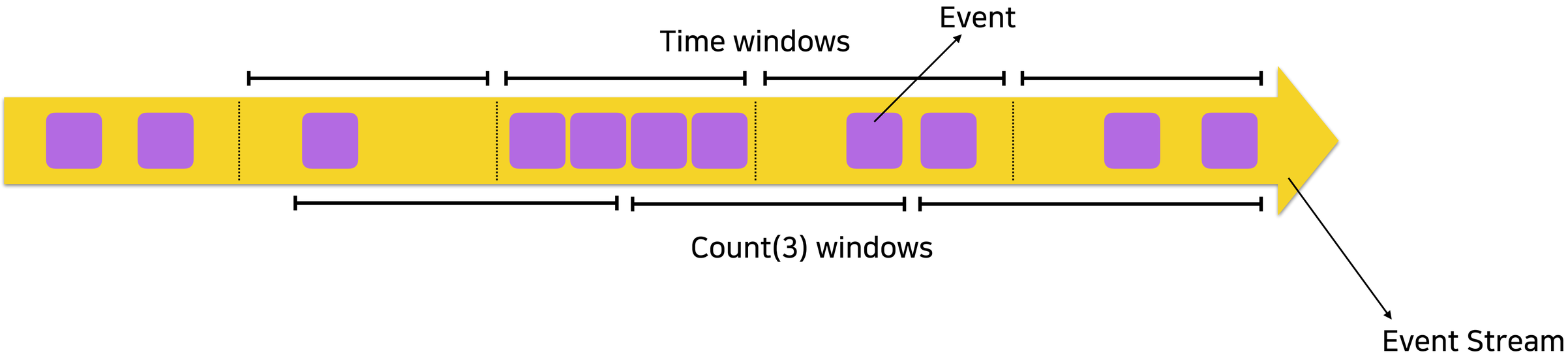
AND

`level == 3`

4.2 DCEP 애플리케이션의 핵심 기능

윈도우 연산(Time window, Count window)

시간/건수 등 특정 단위 윈도우에서 조건 만족 여부를 찾아내는 집합 연산



4.2 DCEP 애플리케이션의 핵심 기능

스트림 리플레이(Stream Replay)

과거의 스트림 데이터를 리플레이하여 프로세싱 결과를 재처리
프로세싱이 빠를 경우, 현재의 레코드를 따라잡아 실시간 프로세싱에 연결 가능

4.3 정적 VS 동적 이벤트 프로세싱

정적 이벤트 프로세싱

정적 코드 정의를 통해, 스트림 프로세싱 토폴로지를 빌드 타임에 결정

동적 이벤트 프로세싱

스트림 프로세싱 토폴로지가 런타임에 결정됨

동적 이벤트 프로세싱을 위해 해결해야 하는 문제들

문제 1) 순차적 미션 달성 판단의 문제

문제 2) Window 연산

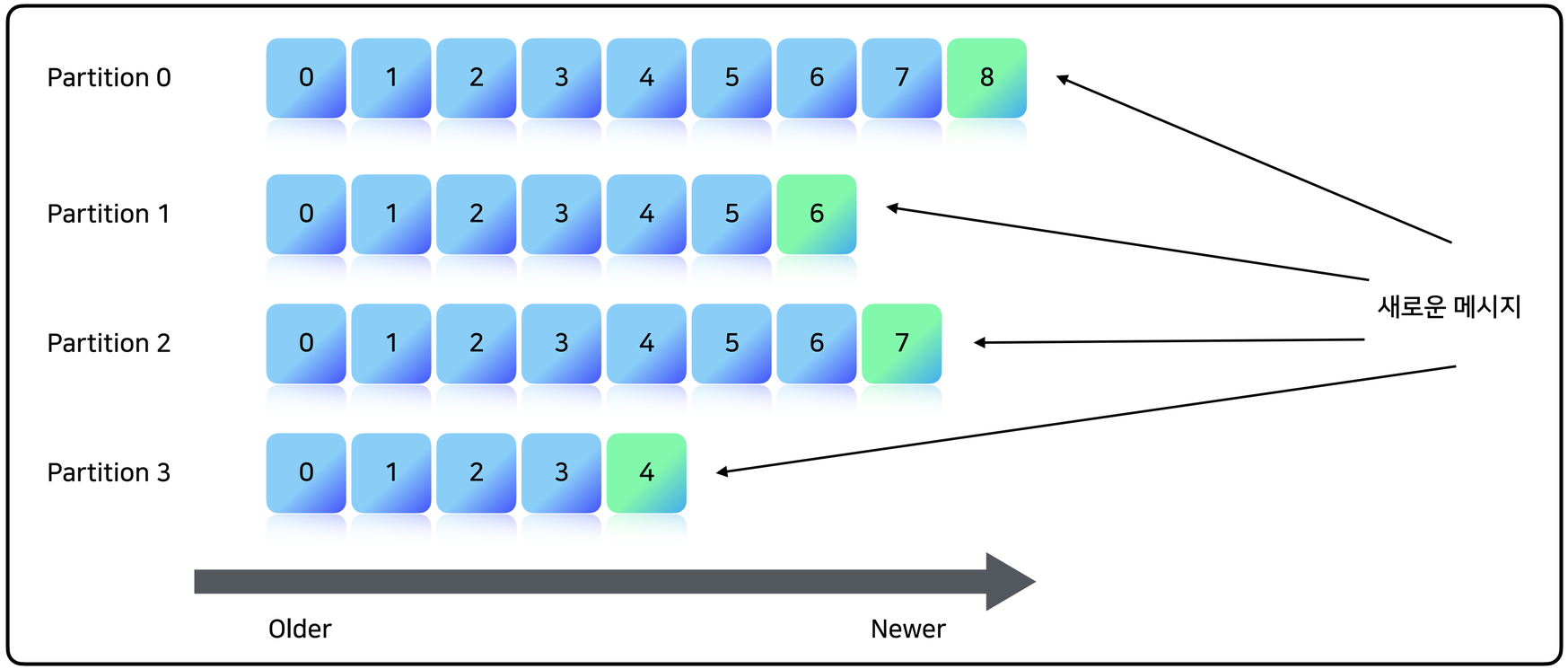
문제 1)

순서가 보장되는 않는 분산 큐 구조에서

순차적 미션 달성 여부를 어떻게 판단해야 하는가

4.4 문제해결) 이벤트 처리 순서 보장하기

순서를 보장할 수 없는 이유 :
카프카는 파티션별로는 순서 보장이 되지만,
분산 특성상 여러 파티션 간 순서 보장이 지원되지 않음



4.4 문제해결) 이벤트 처리 순서 보장하기

DEVIEW 2021

로그 데이터에 실제 이벤트 시간을 반드시 포함

순서가 바뀌더라도 이벤트 시간을 통해 순서를 복구할 수 있음

```
{
  game_id : “kingdom”
  event : “level_up”
  properties : {
    level : 3
    play_id : “G01-54785”
  },
  timestamp: 16874587450
}
```

4.4 문제해결) 이벤트 처리 순서 보장하기

DEVIEW 2021

Event Processing Bucket

무한 이벤트 스트림을 유한 집합으로 치환한 연산 수행 객체

PSA 패턴 적용시 Redis Sorted Set, DBMS Table 등으로 손쉽게 교체 가능

Application
<Linked List>

구현이 간단하지만,
분산 처리의 한계

DBMS
<Table>

분산 처리 가능
트래픽이 늘어날 경우 부하로 이어질 수

Redis
<Sorted Set>

분산 처리 가능
성능이 훌륭하지만 영속성의 문제가 존재

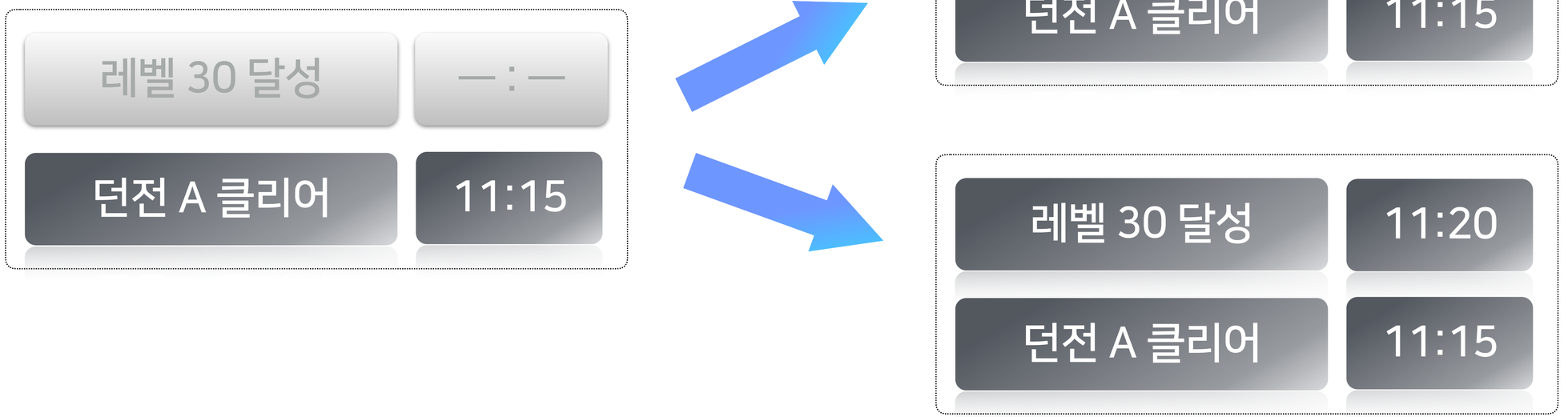
4.4 문제해결) 이벤트 처리 순서 보장하기

DEVIEW 2021

조건이 2개인 경우 - 이벤트 버킷 활용

순서에 상관없이 부분 미션 결과를 저장하고, 나머지 미션 결과 도착시 선후 관계를 판단

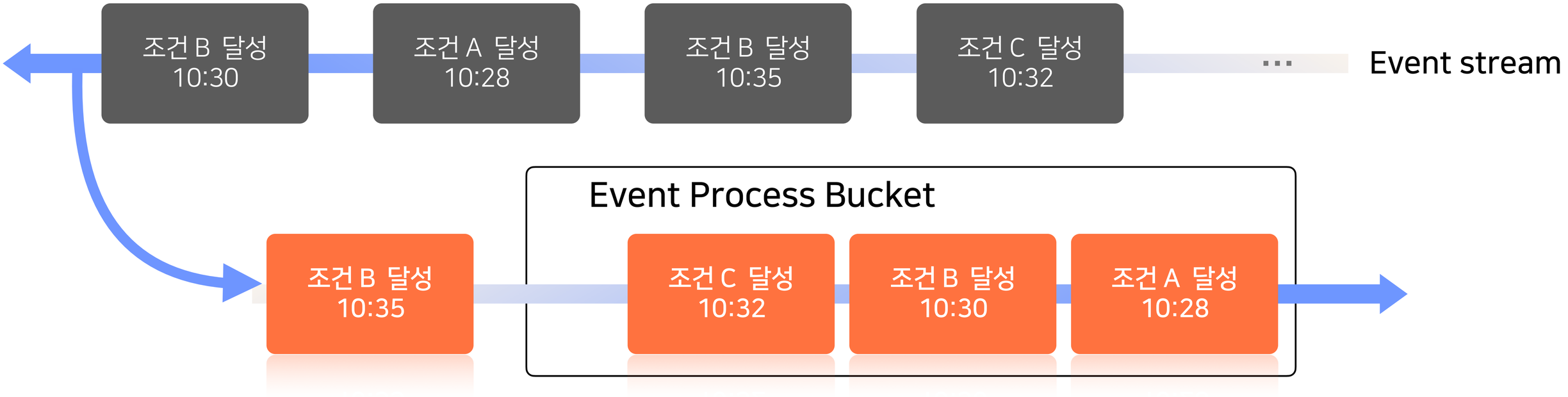
Ex) 레벨 30 달성 후 던전 A 클리어하기



4.4 문제해결) 이벤트 처리 순서 보장하기

조건이 3개 이상인 경우 - 이벤트 버킷 + 이벤트 버퍼링 기법 활용

미션 데이터를 이벤트 버킷에 버퍼링하고, 워터마크 범위에 따라 연산



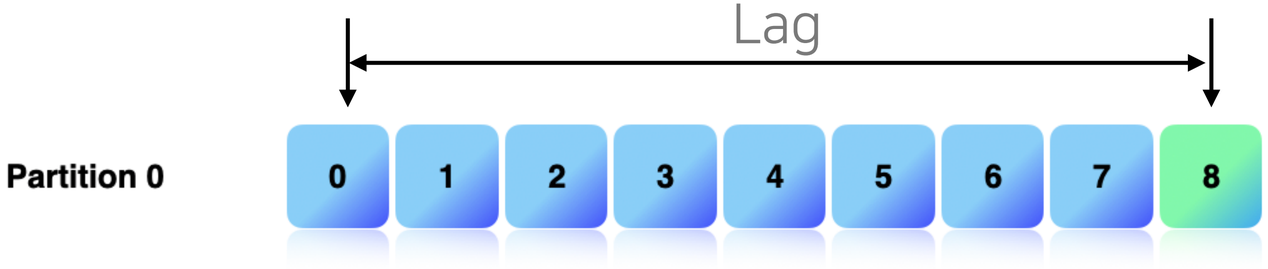
4.4 문제해결) 이벤트 처리 순서 보장하기

워터마크 : 레코드의 지연을 고려한 이벤트 버퍼링 최대 시간

워터마크가 짧으면 데이터 드랍 비율이 상승 ➡ 프로세싱 결과에 대한 신뢰도 저하

워터마크가 길면 프로세싱 레이턴시가 상승 ➡ 애플리케이션 성능 지표 저하

최적의 워터마크 값 : Consumer Lag을 연산 식에 반영하는 것이 중요



워터마크 산출

$$\text{기본 워터마크(초)} + \frac{\text{Max(LAG)}}{\text{Throughput/s}}$$

4.4 문제해결) 이벤트 처리 순서 보장하기

DEVIEW 2021

보상 처리의 경우 워터마크가 민감하지 않으므로 충분히 길게

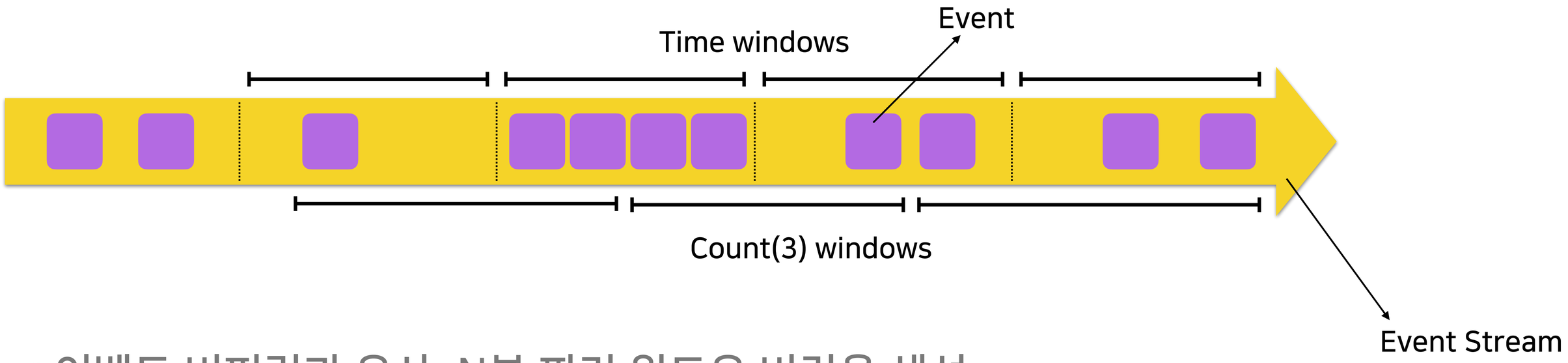
미션 관련 메시지가 도착할 때마다 Event Process Bucket을 순회 : O(n)의 시간 복잡도

문제 2)

Window 연산은 어떻게 구현해야 할까?

4.5 문제해결 2) Window 연산의 구현

Time window와 Count window를 범위에 집어넣는 것이 중요



- ✓ 이벤트 버퍼링과 유사, N분 짜리 윈도우 버킷을 생성
- ✓ 이벤트 데이터를 정렬 후 저장
- ✓ 동일 그룹 데이터 발생시 윈도우 버킷 연산 반복

4.6 그 외, 고려해야 하는 문제

대용량 로그를 준실시간으로 처리하기

- 일일 N x 10억 건, 10TB 이상의 로그 사이즈
- 준실시간 프로세싱을 위해서는 다음과 같은 사항을 고려해야 함

1. 이벤트 프로세싱 매니저를 k8s 노드에 고르게 분산 처리
2. 스트림 배치 단위를 증가 및 병렬 연산 처리
3. 프로세싱 시퀀스의 앞쪽에 선택도(Cardinality)가 높은 조건을 배치
4. 동시에 여러 이벤트 프로세싱을 핸들링하는 경우, Bin Packing 고려

5.이벤트 프로세싱, 그 후..

5.1 실제 사용 현황

1개월당 10~30개의 대규모 이벤트 프로모션 진행에 활용

1일 접속 3회 달성 보상	킹덤	로그인 이벤트 X 3	진행 종료	2021-08-09 14:00:00 KST ~ 2021-08-11 23:59:59 KST
1일 접속 4회 달성 보상	킹덤	로그인 이벤트 X 4회	진행 종료	2021-08-09 14:00:00 KST ~ 2021-08-11 23:59:59 KST
PVP 스킬 사용해서 승리 5회	킹덤	PVP 스킬 사용 승리 X 5회	진행 종료	2021-08-09 14:00:00 KST ~ 2021-08-11 23:59:59 KST
PVP 패배 후 업그레이드하여 승리 3회	모든게임	PVP 패배 X 업그레이드 X PVP 승리 3회	진행 종료	2021-08-09 14:00:00 KST ~ 2021-08-11 23:59:59 KST
상품 구매 후 PVP 승리	킹덤	상품 구매 + PVP 승리	진행 종료	2021-08-09 14:00:00 KST ~ 2021-08-11 23:59:59 KST
푸시 메시지 받고 접속한 유저 보상	ck	푸시 메시지 수신 + 로그인	진행 종료	2021-08-09 14:00:00 KST ~ 2021-08-11 23:59:59 KST

개발팀의 리소스 Free

5.2 if .. then 추상화를 통한 다변화

미션 달성 시

조건 만족 시

Portable Service Abstraction

태그 달기	User Segment System
유저 제재하기	Abusing User Detecting System
Remote API 호출하기	Normal Asynchronous System

Ex)

최근 10분 내 로그인 횟수가 20번 이상인 유저는 1시간 동안 접속 제한하기

한달 내 상품 구매 횟수가 3번 이상인 유저는 #VIP 태그 달아서 특별 혜택주기

매주 월요일 아침 9~10시 사이에 로그인하면 구매 포인트 100 지급하기

5.3 남은 과제들

DEVIEW 2021

Stream Processing을 위한 전용 데이터 파이프라인

- 스트림 프로세싱 애플리케이션이 늘어나면 결국 카프카에 부담

- 전용 카프카 클러스터로 미러링할 경우 해소 가능

Thank You