



# S3 Sink Connector 적용기

Kafka 한국 사용자 그룹 meetup - 김대호

# 안녕하세요!

김대호 입니다 :)

---



## 반갑습니다 :)

- 우아한형제들 재직 중
  - 카프카를 운영하고 관련 기능을 개발합니다.
  - 근래는 Kafka Connect에 관련된 기능을 개발하고 운영 중입니다!
  - 처음 경험하는 다양한 이슈들로 허우적 허우적...
- 카프카 한국 사용자 그룹에서 소소하게 활동 중
- 오픈소스 문화에 기여하기 위해 노력 중

---

그리고

오늘은 S3 Sink Connector를  
직접 사용해본 경험을 공유하고자 합니다.



# 오늘의 이야기

1. Kafka Connect 란?
2. S3 Sink Connector
  - a. 도입 배경
  - b. 주요 설정
  - c. 예제 시연
  - d. 이슈와 한계
3. 마무리

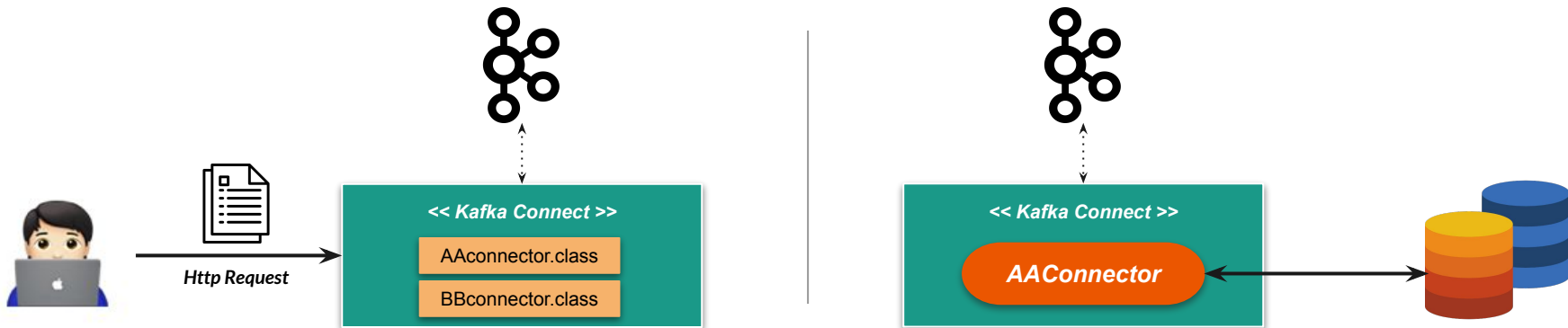
## Kafka Connect란?

- 카프카는 브로커를 중심으로 프로듀서와 컨슈머가 메시지 파이프라인을 구성
- 만약,
  - MySQL의 데이터를 읽어서 카프카에 프로듀싱하고 싶다면?
  - 카프카의 메시지를 컨슈밍하여 ElasticSearch에 저장하고 싶다면?
  - ... Slack으로 쓰고 싶다면?
  - ... S3에 저장하고 싶다면?
- 데이터 저장소 별로 개별적인 클라이언트를 구성하는 것은 **매우 힘든 일**
- 각 데이터 저장소에 대해 프로듀싱/컨슈밍하는 **구현체가 제공된다면?**
  - => *Kafka Connect!!*



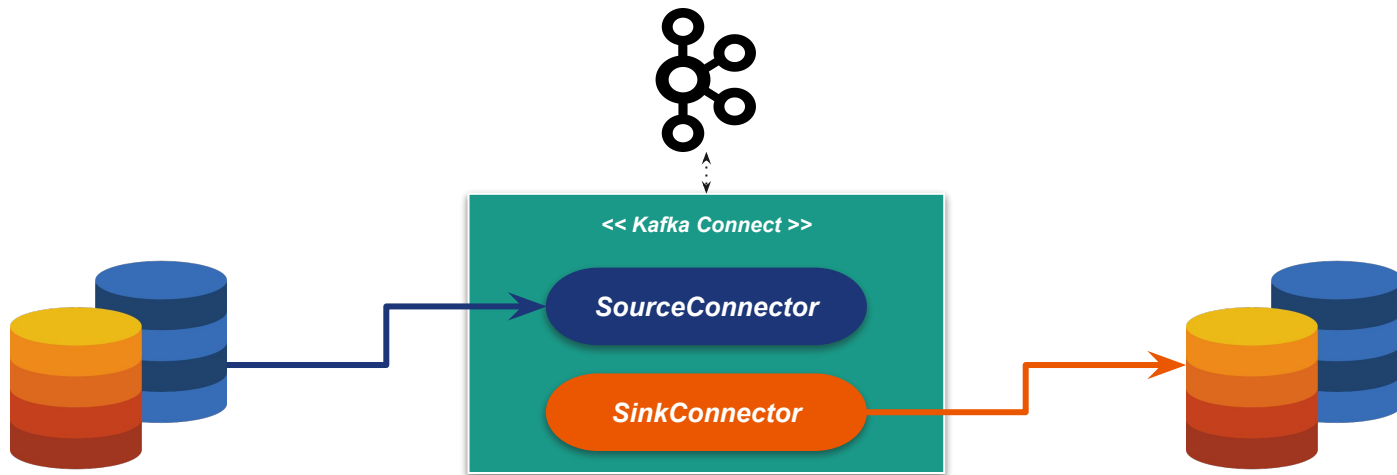
# Kafka Connect 의 구성

- Connect는 Connector가 배포되어 구동하는 리소스
  - 일반적으로 클러스터로 구성하여 사용한다
- Connector는 특정 데이터 저장소에 대한 pub/sub 구현체
  - Connect에 배포되고 실행된다.
  - 구현체를 기반으로, 사용자의 설정을 주입 받아 실행된다.



## Source Connector vs Sink Connector

- 데이터 저장소의 데이터를 읽어, 카프카에 프로듀싱하는 **Source 커넥터**
- 카프카의 메시지를 컨suming하여, 데이터 저장소에 저장하는 **Sink 커넥터**





## S3 Sink Connector란?

- 말 그대로 **카프카의 메시지를 S3 Bucket에 적재하는 커넥터**
- Confluent 에서 오픈소스로 제공하고 있다.
  - Confluent Community License



그렇다면 왜

S3 Sink Connector 를

썼을까?

—

## 왜 쓰게 됐을까?

카프카를 사용하던 사내 개발자들의 요구사항

- 카프카에 저장된 **메시지**를 **백업**하고 싶어요!
- **AWS Athena**로 메시지를 집계하고 싶어요!

카프카를 운영하던 우리 팀의 요구사항

- 개발, 배포, 운영, 유지보수가 **간편하면 좋겠다.**

대표적인 스토리지, S3에 적재하자!



Kafka Connect를 활용하자!



## 적용해 본 결과!

- 다양한 서비스의 이벤트들이 **성공적으로 S3에 백업**되고 있다.
  - 보관이 필요한 메시지의 영구적인 저장
  - Athena 를 이용해서 데이터 집계 가능
- http 스크립트 관리로 **손 쉽게 커넥터 관리**
  - 코드로 커넥터의 현재 상태 관리
  - 배포/재시작/중지 등 http 요청으로 관리

```
$ curl -vvX POST http://localhost:8083/connectors/s3-sink-local/restart
* Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8083 (#0)
> POST /connectors/s3-sink-local/restart HTTP/1.1
> Host: localhost:8083
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 204 No Content
< Date: Sun, 04 Jul 2021 08:01:10 GMT
< Server: Jetty(9.4.40.v20210413)
<
* Connection #0 to host localhost left intact
* Closing connection 0
```

Amazon S3 > kafka-example-bucket > backup/ > examples-s3-sink/ > year=2021/ > month=07/ > day=04/ > hour=10/

hour=10/ [Copy S3 URI](#)

**Objects** | Properties

**Objects (6)**


Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#)

[Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">examples-s3-sink+0+0000000000.json</a>	json	July 4, 2021, 10:55:01 (UTC+09:00)	23.3 KB	Standard
<input type="checkbox"/>	<a href="#">examples-s3-sink+0+0000000122.json</a>	json	July 4, 2021, 10:56:01 (UTC+09:00)	46.9 KB	Standard
<input type="checkbox"/>	<a href="#">examples-s3-sink+0+0000000368.json</a>	json	July 4, 2021, 10:57:01 (UTC+09:00)	44.4 KB	Standard
<input type="checkbox"/>	<a href="#">examples-s3-sink+0+0000000602.json</a>	json	July 4, 2021, 10:58:01 (UTC+09:00)	46.8 KB	Standard
<input type="checkbox"/>	<a href="#">examples-s3-sink+0+0000000848.json</a>	json	July 4, 2021, 10:59:01 (UTC+09:00)	47.8 KB	Standard
<input type="checkbox"/>	<a href="#">examples-s3-sink+0+000001099.json</a>	json	July 4, 2021, 11:00:01 (UTC+09:00)	45.4 KB	Standard



## 미리 알아두면 ~~덜~~ 삽질할 수 있는 좋은 설정들

1. Converter 설정
2. Partitioner 설정
3. Flush 설정
4. S3 설정

```
{
  "connector.class": "io.confluent.connect.s3.S3SinkConnector",
  "topics": "examples-s3-sink",
  "tasks.max": "1",

  "key.converter": "org.apache.kafka.connect.storage.StringConverter",
  "value.converter": "org.apache.kafka.connect.json.JsonConverter",
  "value.converter.schemas.enable": "false",

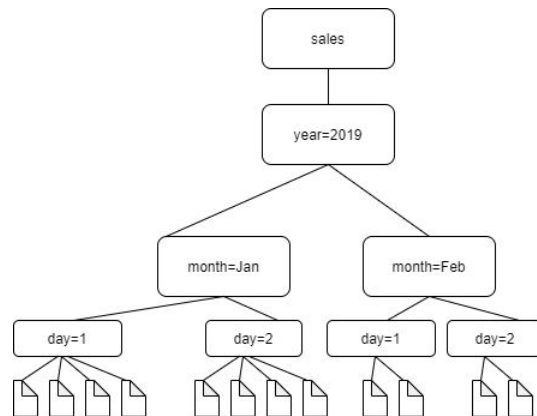
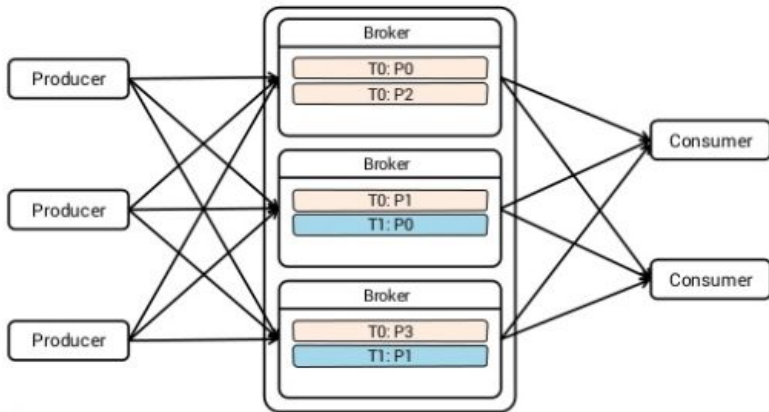
  "s3.region": "ap-northeast-2",
  "s3.bucket.name": "demo-bucket",
  "s3.part.size": "5242880",
  "s3.proxy.url": "http://localhost:4566",
  "storage.class": "io.confluent.connect.s3.storage.S3Storage",
  "topics.dir": "backup",
  "aws.access.key.id": "user",
  "aws.secret.access.key": "secret",

  "format.class": "io.confluent.connect.s3.format.json.JsonFormat",
  "flush.size": "500",
  "rotate.schedule.interval.ms": "60000",

  "partitioner.class": "io.confluent.connect.storage.partitioners.TimeBasedPartitioner",
  "path.format": "'year'=YYYY/'month'=MM/'day'=dd/'hour'=HH",
  "partition.duration.ms": "3600000",
  "locale": "ko_KR",
  "timezone": "Asia/Seoul",
  "timestamp.extractor": "Record"
}
```

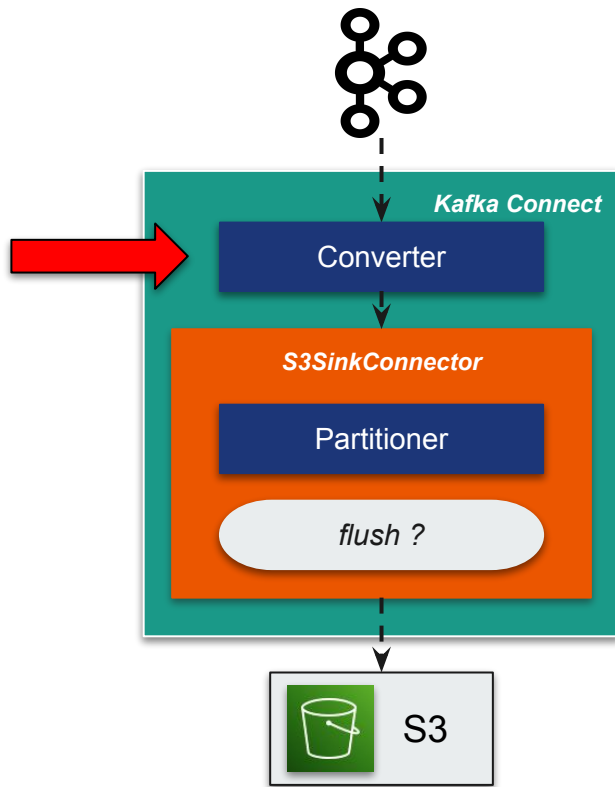
## 들어가기 전에 잠깐!

- 카프카 토픽의 파티션 = **토픽 파티션**
- S3 버킷에 오브젝트 저장 경로 = **S3 파티션** 혹은 파티션



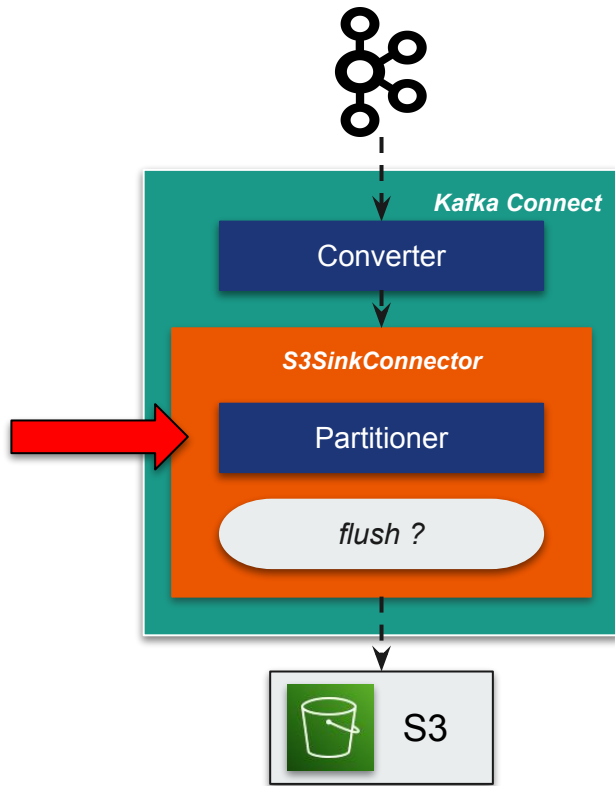
## 주요 설정 - 1. Converter 설정

- 메시지의 key와 value의 직렬화와 관련된 설정
  - `key.converter`
  - `value.converter`
- 일반적으로 사용되는 설정 값
  - `org.apache.kafka.connect.storage.StringConverter`
  - `org.apache.kafka.connect.json.JsonConverter`
- **프로듀서가 직렬화한 방식에 따라 개발이 필요할 수 있음**
  - 예를 들면, CBOR...



## 주요 설정 - 2. Partitioner 설정

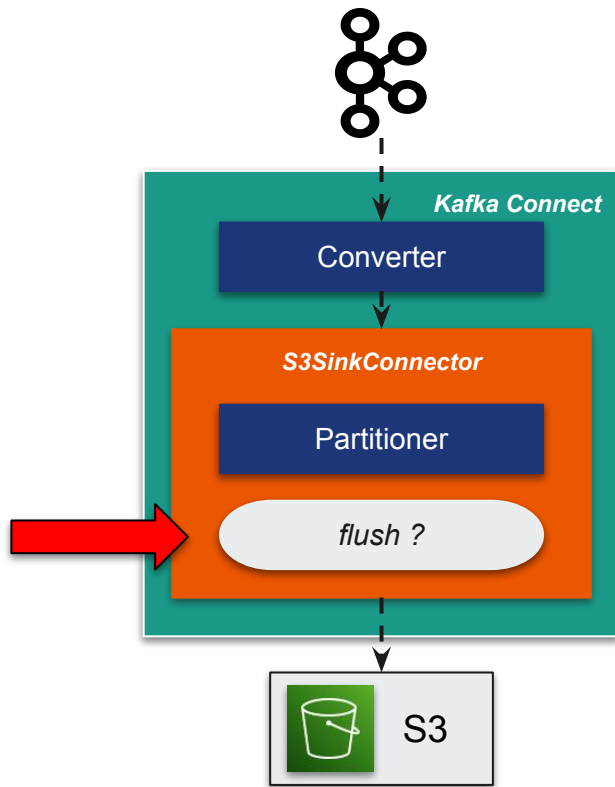
- S3에 어떤 경로로 오브젝트를 적재할 것인가에 관한 설정
  - 크게 2가지 파티셔너로 나뉜다.
- **FieldPartitioner**
  - 메시지의 필드 값에 따라 파티셔닝
  - 현재 struct 타입만 가능 (schema registry 연동)
  - nested field 설정 불가
- **TimeBasedPartitioner**
  - 메시지의 시간 값에 따라 파티셔닝
- **두 파티셔너를 같이 사용할 수 없다.**





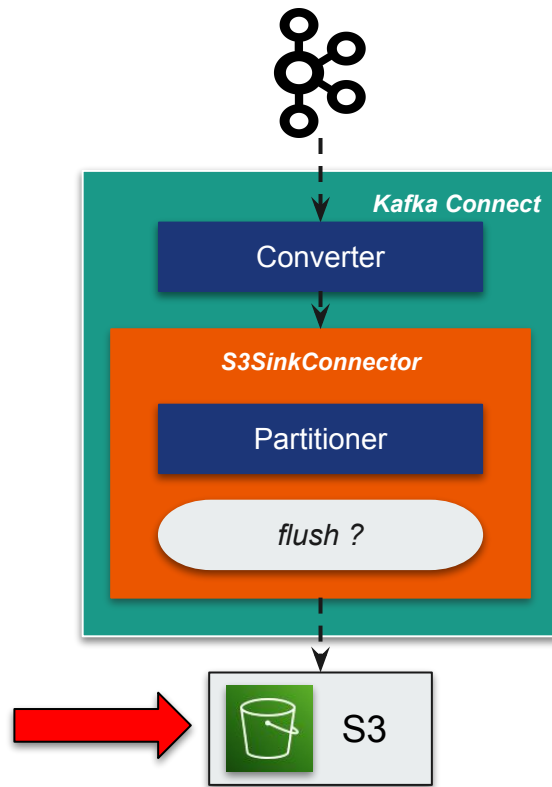
## 주요 설정 - 3. Flush 설정

- 메모리에서 버퍼하고 있는 메시지들을 S3에 flush 하는 설정
  - `s3.part.size`
  - `flush.size`
  - `rotate.interval.ms`
  - `rotate.schedule.interval.ms`
- flush.size :**
  - 토픽 파티션 별로 몇 개의 메시지마다 flush 할 것인가?
- rotate.interval.ms :**
  - 메시지의 어떤 시간 값을 기준으로 몇 ms마다 flush 할 것인가?
  - TimeBasedPartitioner를 사용해야만 적용 가능한 설정**
- rotate.schedule.interval.ms :**
  - 00:00 를 기준으로 몇 ms마다 flush 할 것인가?



## 주요 설정 - 4. S3 설정

- S3에 관한 설정
  - 버킷, 데이터 포맷 등의 설정
- CP 5.5.1 버전 이후로 **AssumeRole** 기능 지원
- **s3.part.size**
  - Multipart upload 시 part의 최소 크기
  - S3 파티션 별 버퍼 사이즈를 결정
  - 최소 5MB



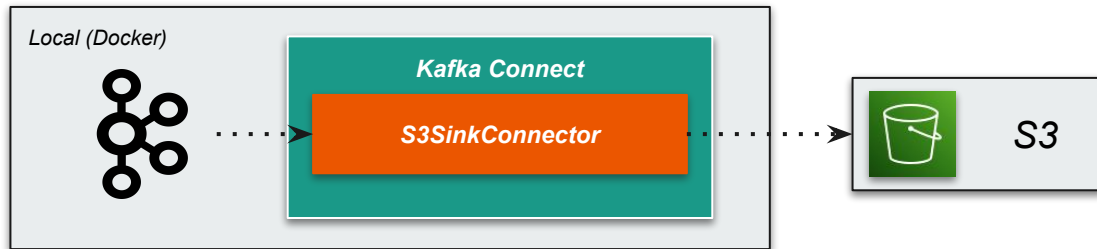
# Show me the Example !

살짝 지루하셨을 여러분을 위해!



## 그럼 어떻게 쓸 수 있을까요?

- <https://github.com/daehokimm/kafka-examples/tree/main/s3-sink-connector>
  - 로컬 도커 환경을 기반으로 구성할 수 있습니다.
  - README.md 를 참고하세요! (이슈 & PR은 환영!)
- 시연은 간단하게!
  - 토픽에 메시지가 프로듀싱 되고 있는지 확인하고
  - S3 Sink Connector를 배포/중단/재시작하고
  - S3에 오브젝트가 잘 적재되는지 확인해봅니다.



---

# 해피 엔딩?

크게 개발할 것도 없고,

설정만 주입해서 배포하면 되고,

커넥터 조작도 쉽고,

해치웠나?



# 그러나 세상에 쉬운 일 하나도 없네!

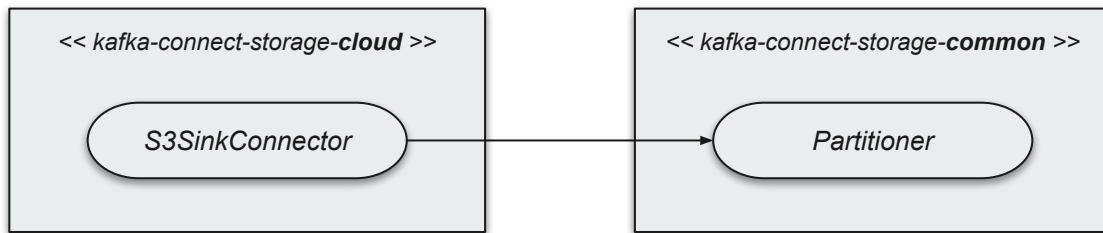
개발/운영하면서 만난 이슈들

---



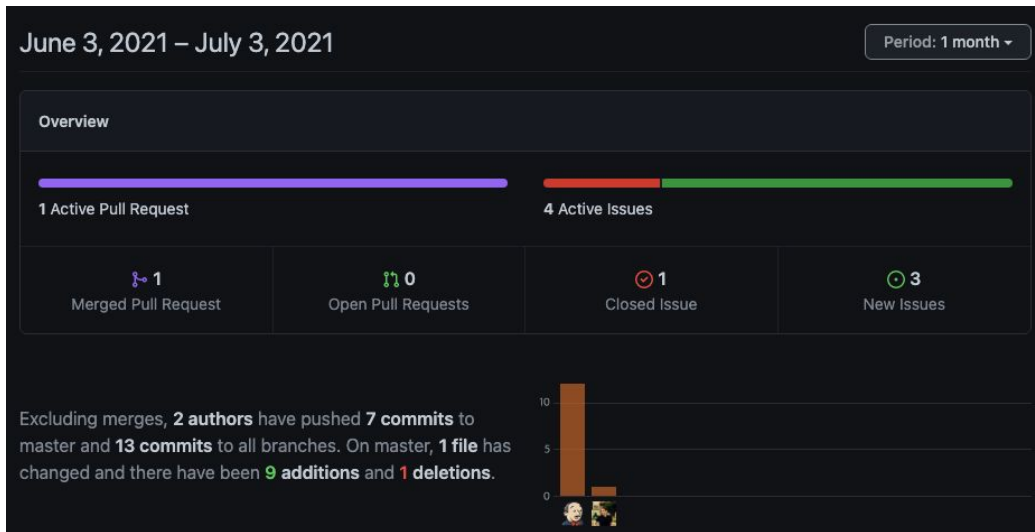
## 적용하면서 만난 이슈 - 개발편

- S3 Sink Connector가 정상적으로 구동되기 위해선 **2개의 프로젝트 빌드 필요**
  - kafka-connect-storage-cloud 는 S3 Sink Connector가 구현된 프로젝트
  - kafka-connect-storage-common 은 Confluent 커넥터들이 공용으로 사용하는 프로젝트
- Partitioner 기능 구현 후 배포하는 과정
  - S3 Sink Connector 배포를 다운로드 받고,
  - kafka-connect-storage-common을 빌드한 결과물(jar)을 교체



## 적용하면서 만난 이슈 - 개발편

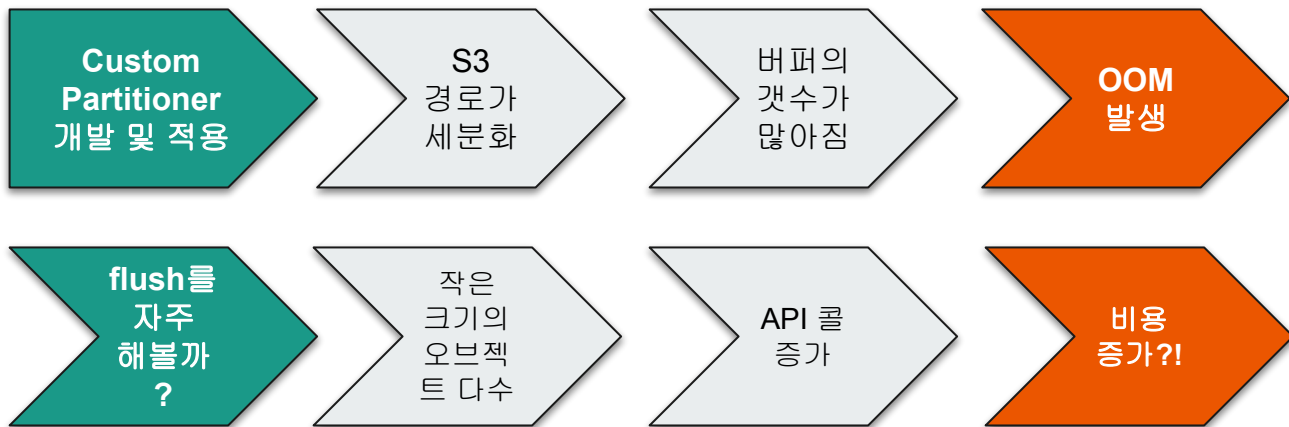
- 조용한 커뮤니티
  - 우리만 쓰나...?





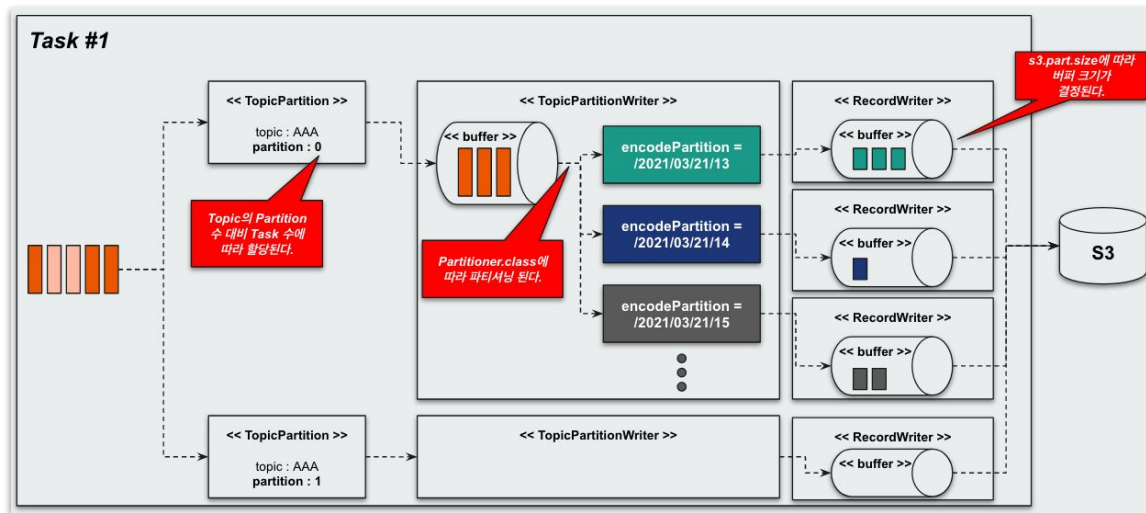
## 적용하면서 만난 이슈 - 운영편

- 개발된 기능을 적용하고, 이슈가 발생했던 과정



## 왜 이슈가 발생했을까?

- 이슈의 가장 큰 원인은 **모든 메시지가 메모리에서 처리된다**는 점



S3 Sink Connector가 메모리를 점유하는 구조

## 깊어지는 고민...

뚝뚝해지는 커넥트

S3 Sink Connector  
때문에 **Connect**  
**스펙을** 올려야하나?

Flush 조건들을  
**매번 계산해서**  
**최적화**해야하나  
?

치솟는 운영 비용

메시지 유입  
상황에 따라  
버퍼의 갯수가  
**유동적**인데..

트래픽이  
적은데 버퍼를  
무조건 **최소**  
**5MB**씩  
잡아야하나?

낭비되는 버퍼  
메모리

언제 OOM이 터질지  
모르는 짜릿함





## 어떻게 해결할 수 있을까?

- 토픽 별 Worst case를 고려한 메모리 산정해보면 어떨까?
  - 터무니 없이 큰 메모리 구성
  - **추후 메시지 값의 확장에 취약함**
- Kafka Streams로 정제해서 토픽 파티션 별 메시지 다양도를 축소시켜보면 어떨까?
  - 동일한 트래픽 2배가 되므로 브로커의 부하 증가 예상
  - **관리 포인트 증가**
    - 부하 분산을 위한 토픽 파티셔너 개발 필요
    - Streams 애플리케이션 개발/운영

⇒ *In-memory 처리가 가지는 한계는 해소하지 못했다.*



## 어떻게 해결할 수 있을까?

- 디스크 처리를 기반으로한 S3 Sink Connector가 있으면 해결할 수 있지 않을까?
  - 디스크에 메시지들을 버퍼했다가 flush 하는 방식
  - 조금 느릴 수 있지만 안정적이고 확장 가능한 커넥터



## 마무리

- S3 Sink Connector는 카프카 메시지를 백업하기에 가장 손 쉬운 방법
- 꽤 간편했던 Kafka Connect 운영
  - 한 번 만들어보니 두 번 만들기 쉽더라!
  - 커넥터 조작이 매우 쉬웠다!
- 하지만 조금 더 다양하게 사용하면 부딪히는 이슈들
  - S3 Sink Connector 에 많은 관심과 기여 부탁드립니다~
  - 관련한 질문 혹은 조언은 페이스북을 통해 환영!

—

감사합니다~!