



시작하세요

분산 메시지 시스템 아파치 카프카



+jead0.k0
haibane84@gmail.com

목차

아파치 카프카 개요 overview

주요 컨셉

Quick Start

아파치 카프카 기반 메시지 전송/수신

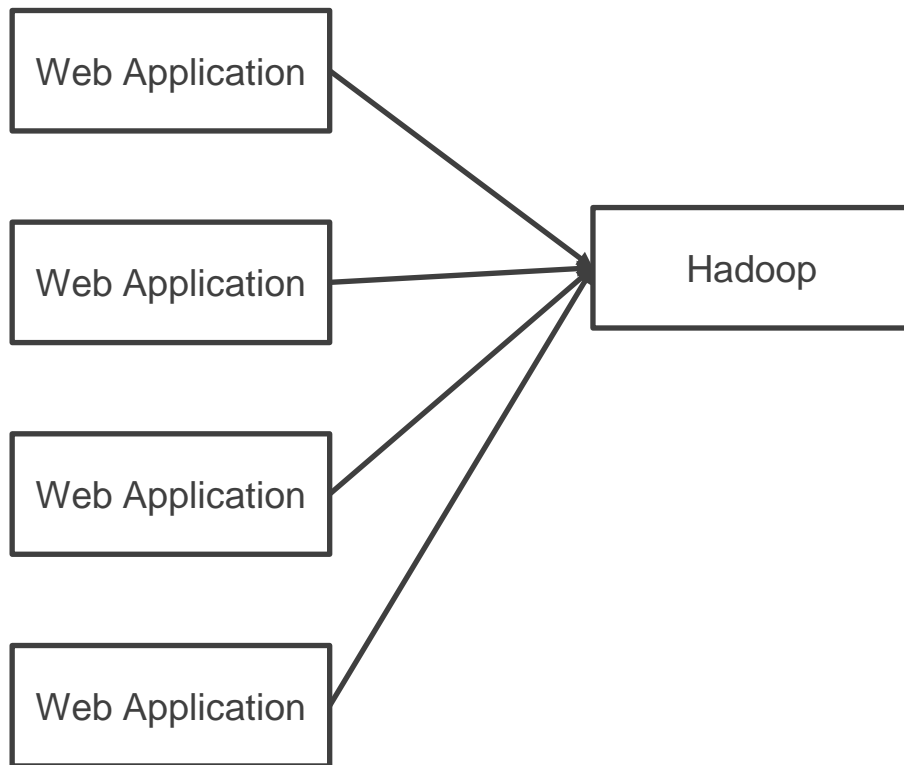
kt IoT 플랫폼 소개

아파치 카프카 개요

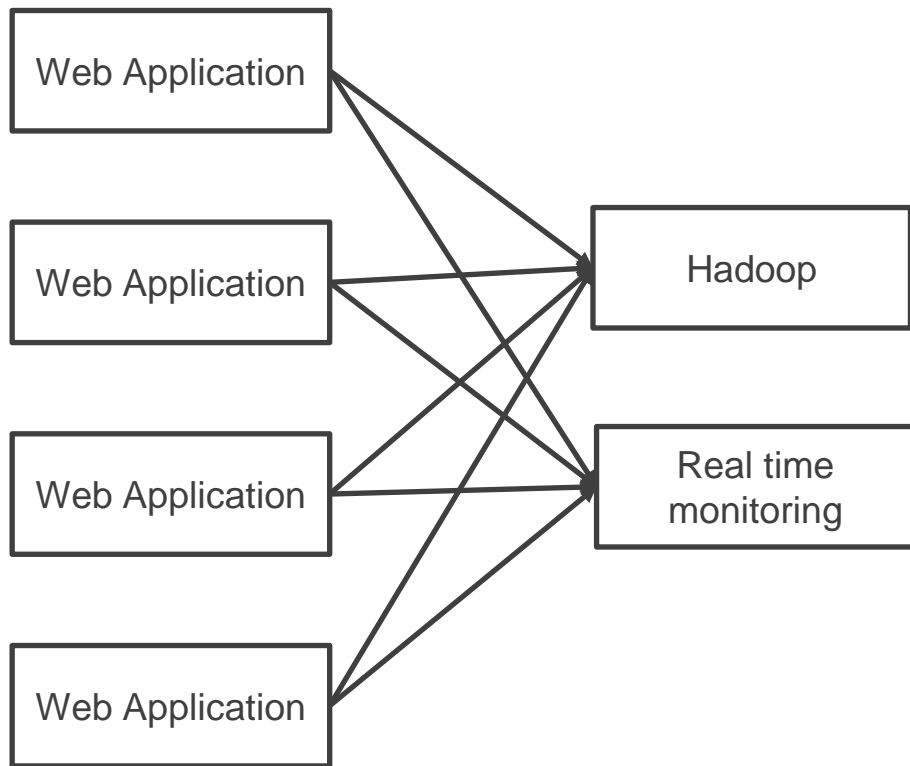
주로 하나의 데이터 파이프라인으로 시작



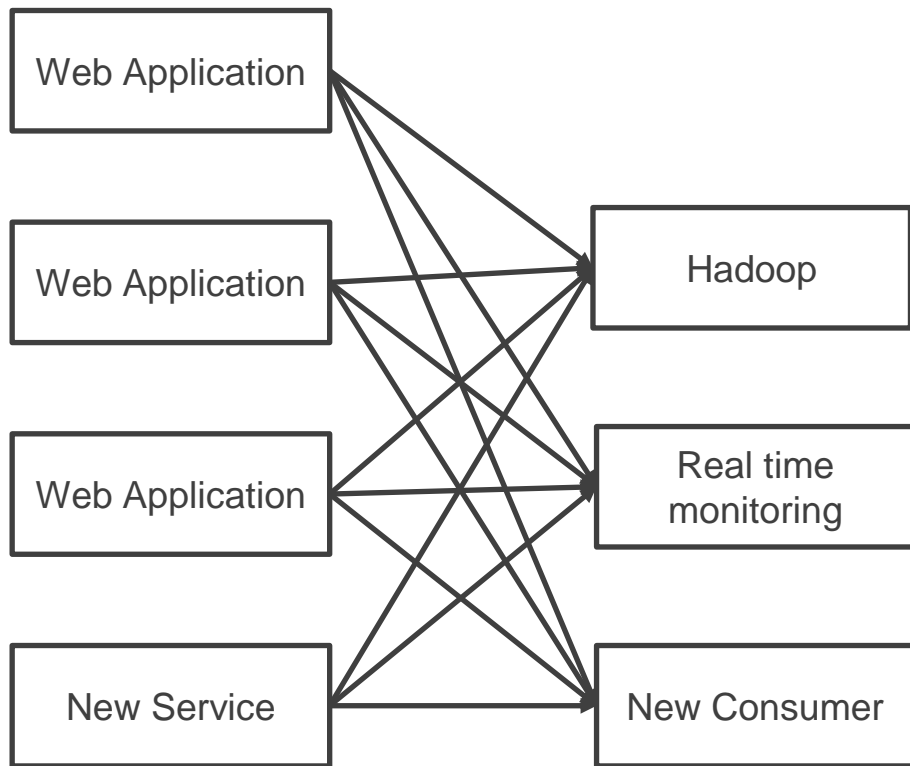
새로운 데이터 제공자^{producer}는 기존의 파이프라인을 재사용

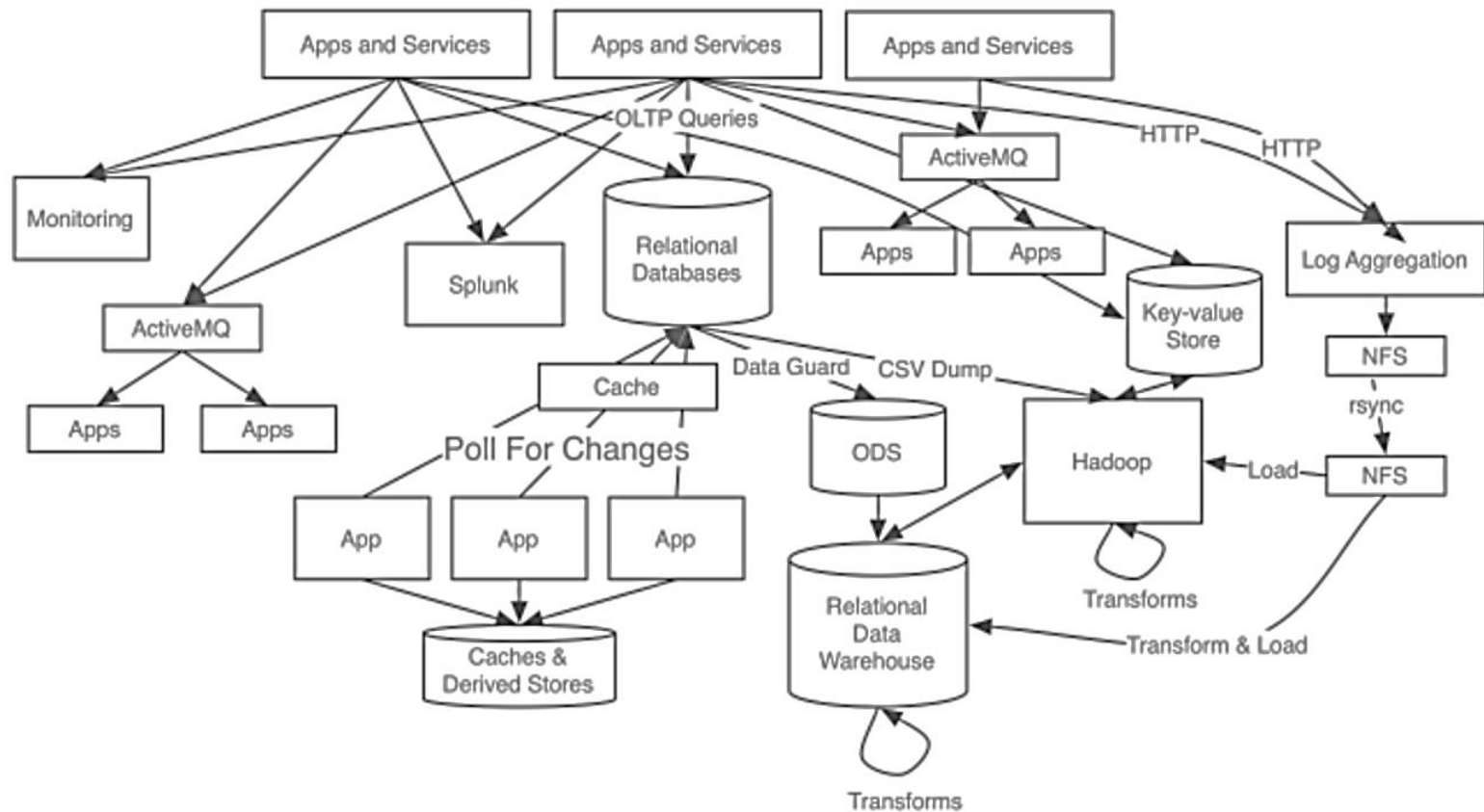


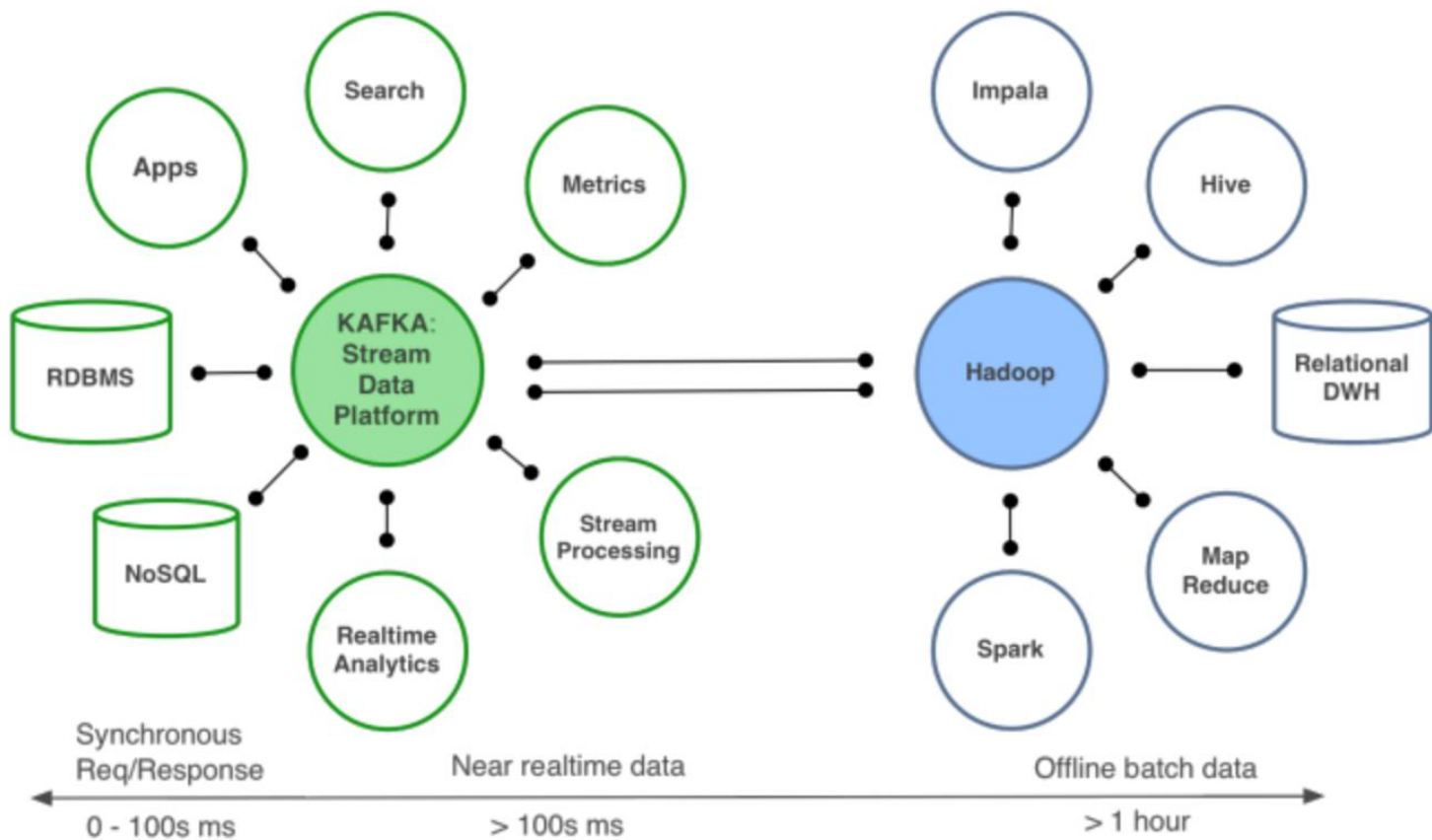
하지만 새로운 데이터 소비자^{consumer}가 나타납니다.



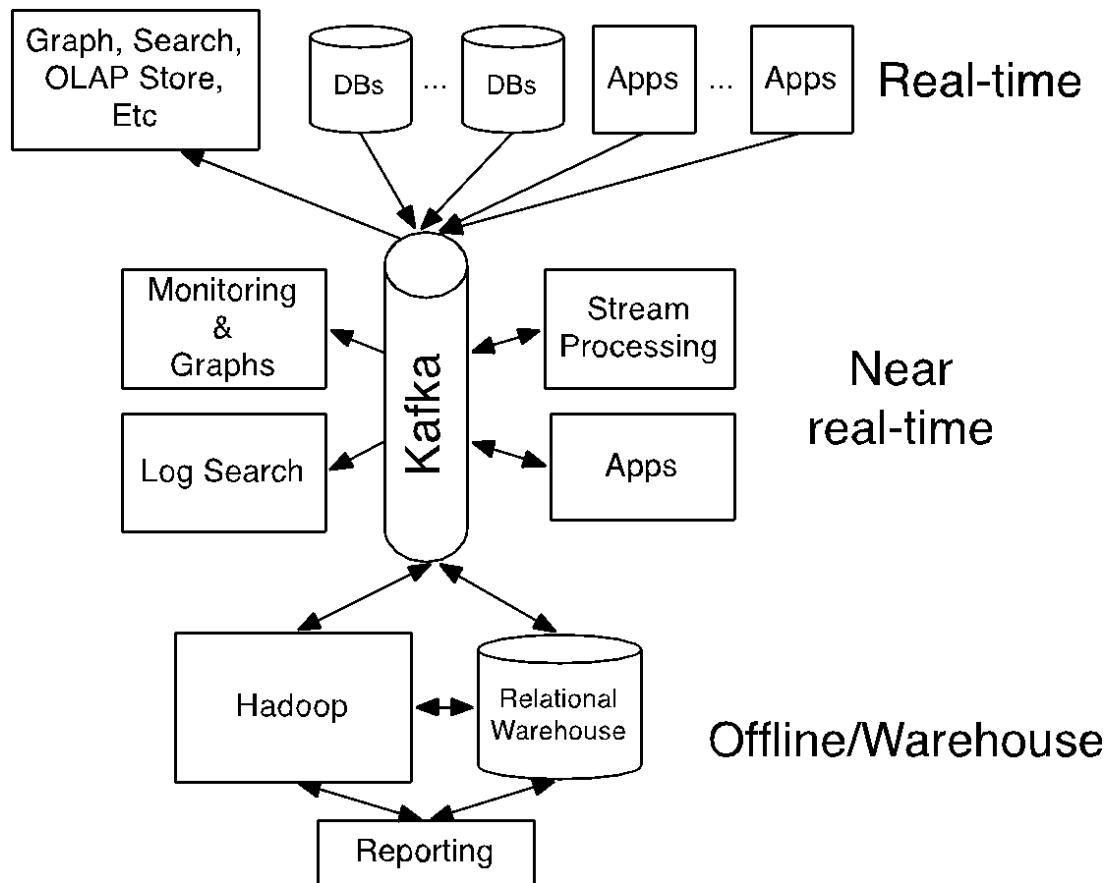
그것도 계속.... 결국 복잡한 골칫거리만 만들죠







이미지 출처 : Putting Apache Kafka To Use: A Practical Guide to Building a Stream Data Platform (Part 1)
(<http://www.confluent.io/blog/stream-data-platform-1/>)

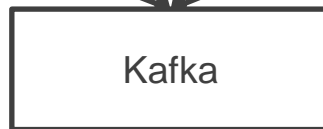


카프카는 메시지 파이프라인을 잊게 해줍니다.

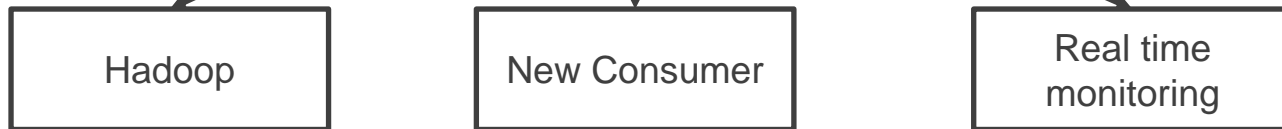
Producers



Brokers



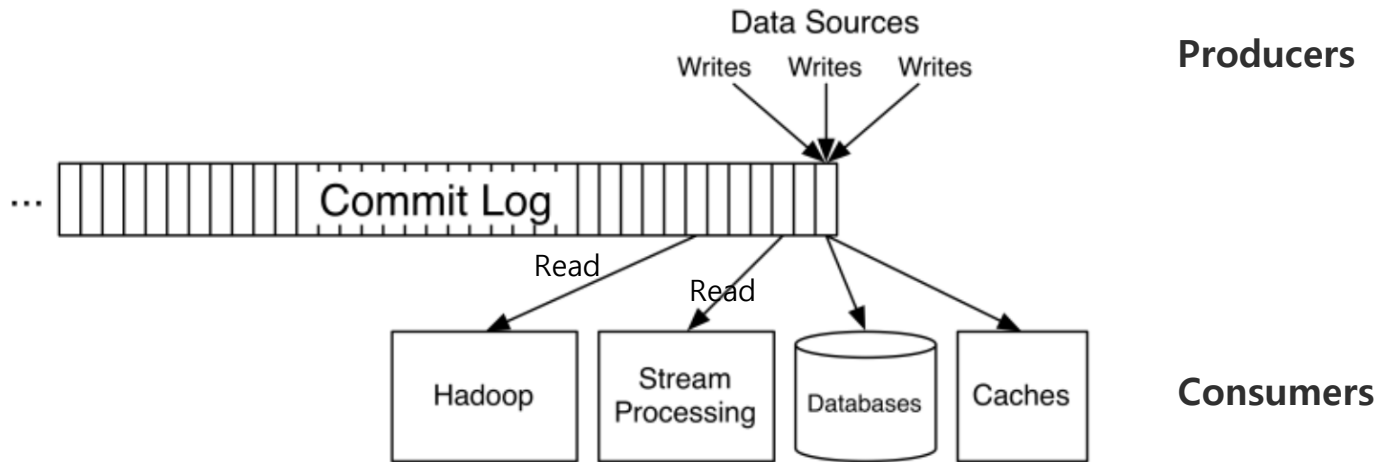
Consumers



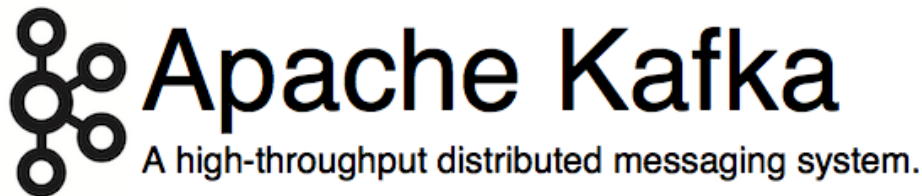
카프카

= 대량신속처리 분산 메시지 시스템 **high-throughput distributed messaging system**

= 분산 커밋 로그



이미지 출처 : Putting Apache Kafka To Use: A Practical Guide to Building a Stream Data Platform (Part 1)
(<http://www.confluent.io/blog/stream-data-platform-1/>)



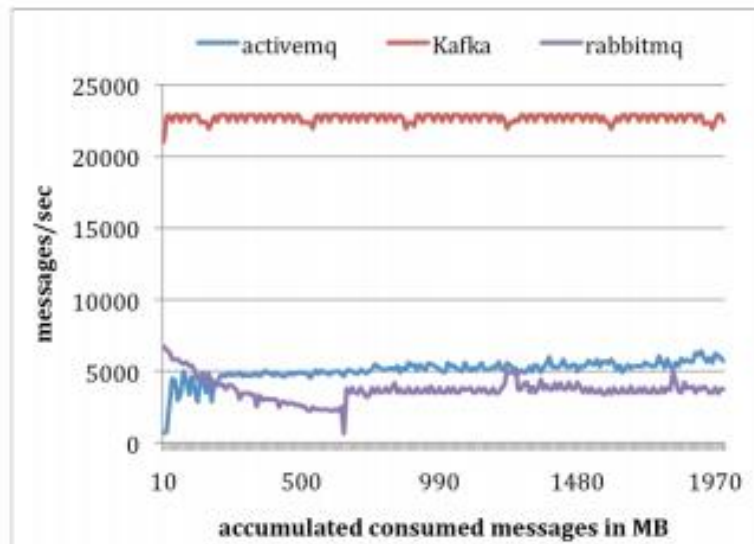
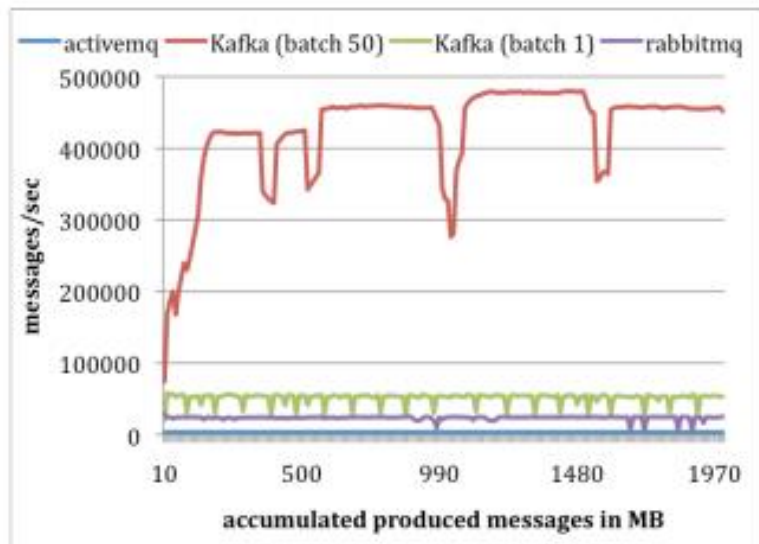
- 대량신속처리 분산 메시지 시스템
- LinkedIn SNA팀에서 개발하고 서비스에 사용 중
 - 2011년 시작
 - Scala, Java 구현
- LinkedIn에서 2015년 기준 60개 이상의 클러스터로 1100개가 넘는 브로커 운영
 - 혼잡할 때 초당 1억 3천만/2.75기가바이트 메시지 처리
- 어디어디를 포함한 xxx 개의 회사가 현재 사용 중

<https://kafka.apache.org/committers.html>

<https://github.com/apache/kafka/graphs/contributors>

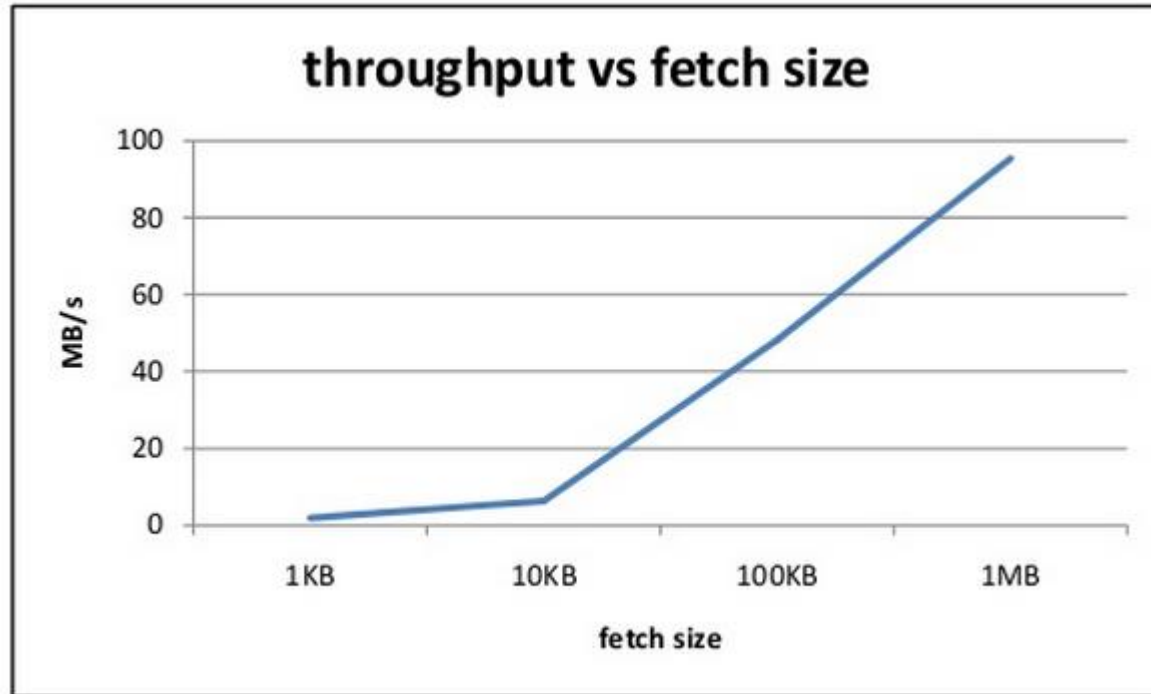
기존 메시징 시스템과의 성능 비교

(2011년 Kafka: a Distributed Messaging System for Log Processing)

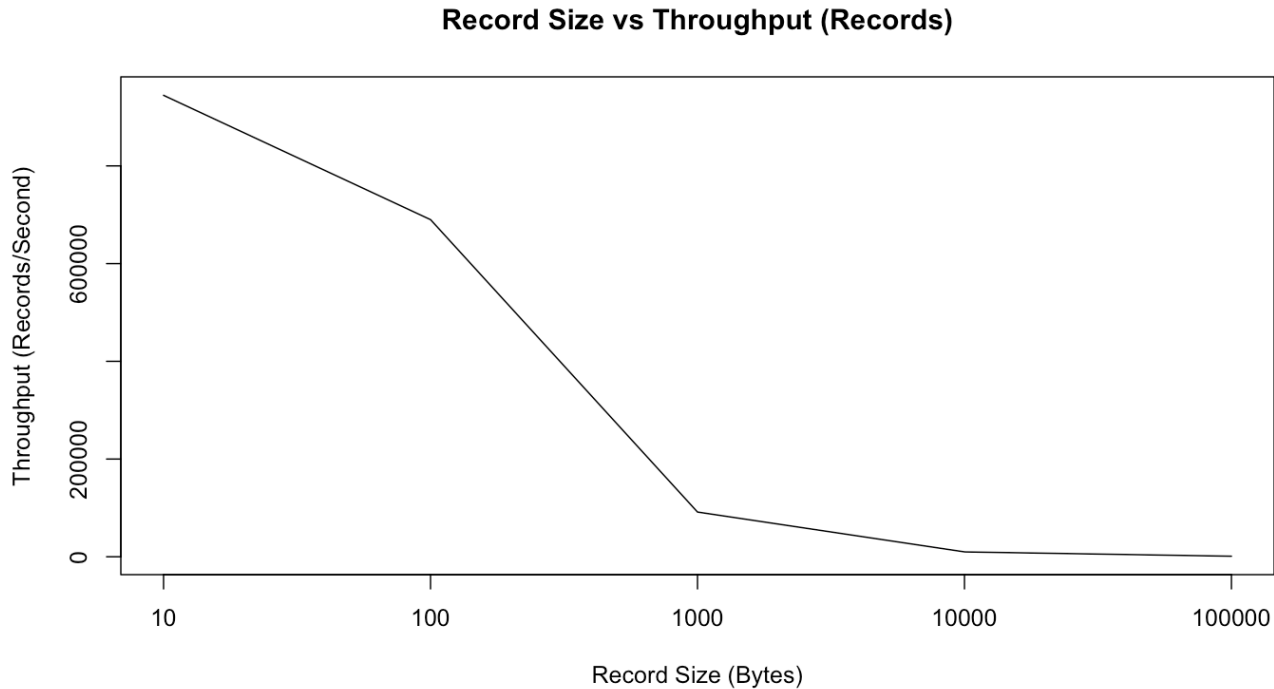


이미지 출처 : Current and Future of Apache Kafka
(<http://www.slideshare.net/charmalloc/current-and-future-of-apache-kafka>)

Consumer Throughput

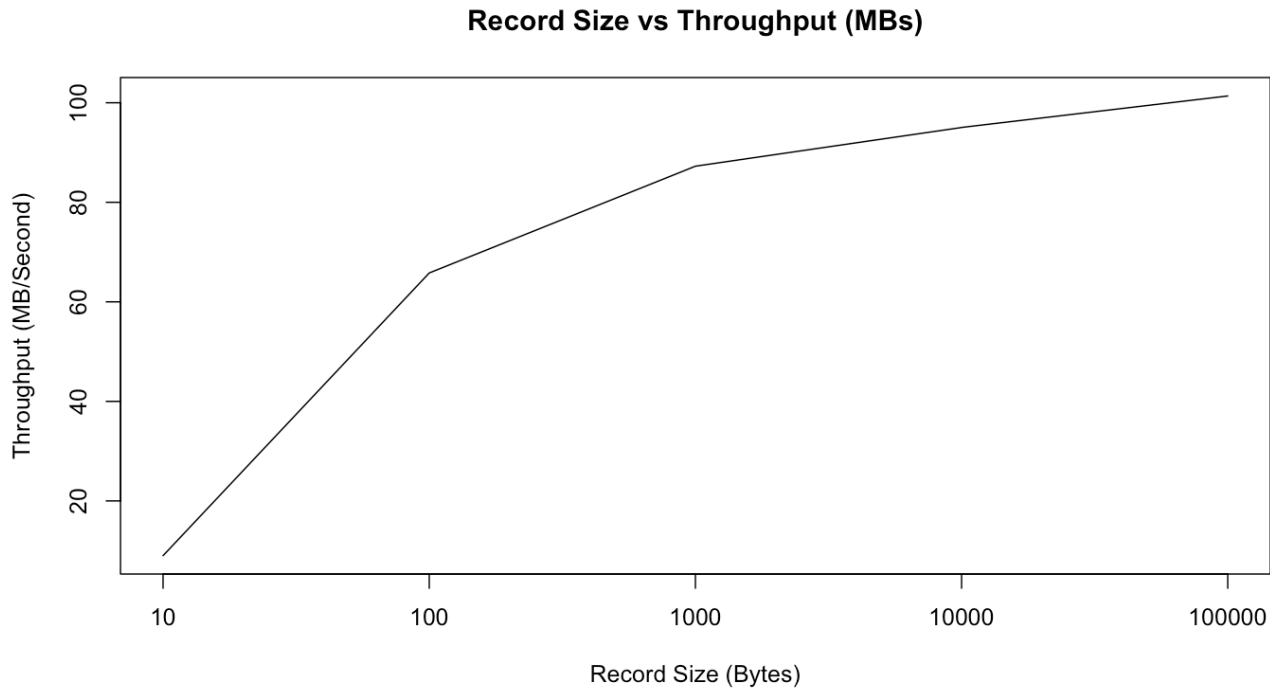


Message Size vs Throughput (count)



이미지 출처 : Current and Future of Apache Kafka
(<http://www.slideshare.net/charmalloc/current-and-future-of-apache-kafka>)

Message Size vs Throughput (MB)

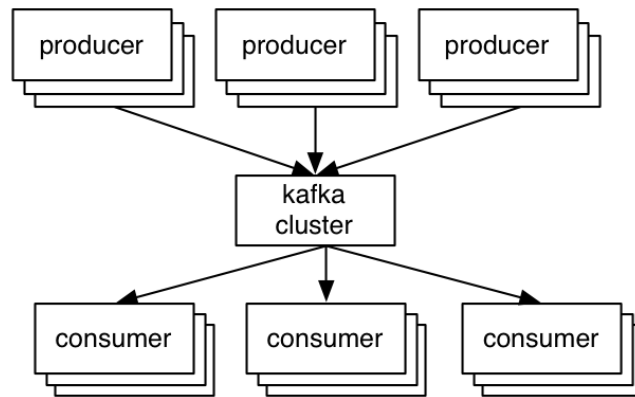


이미지 출처 : Current and Future of Apache Kafka
(<http://www.slideshare.net/charmalloc/current-and-future-of-apache-kafka>)

주요 컨셉

- 구성요소

- 프로듀서는 브로커에게 데이터를 기록
- 컨슈머는 브로커에게 데이터를 읽음
- 모든 것은 분산distributed



- 데이터

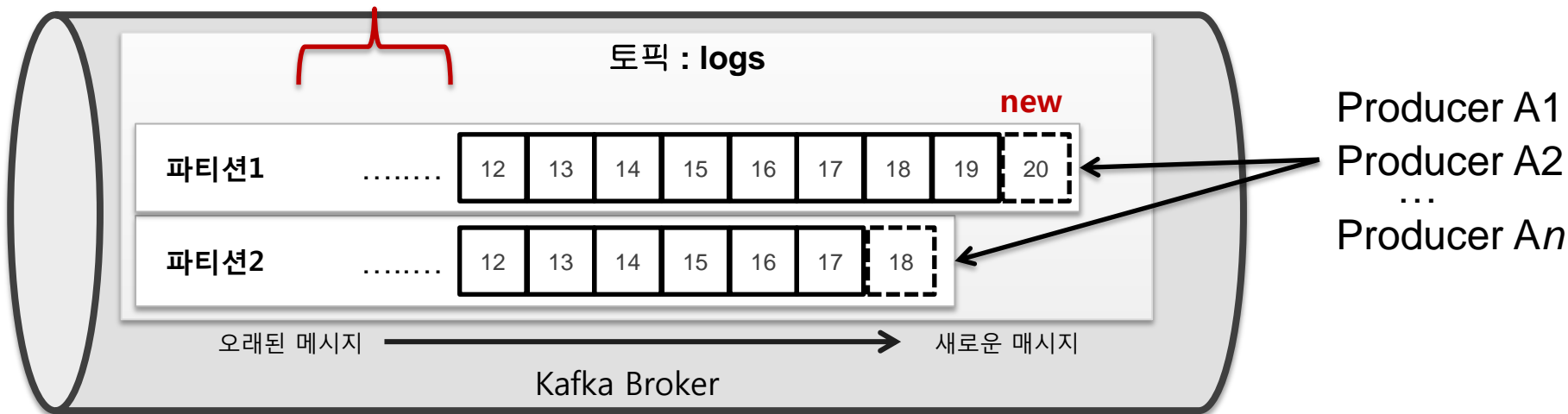
- 데이터는 **토픽**에 저장
- 토픽은 파티션에 분할partitions 그리고 복제replicated되어 있음

토픽/파티션

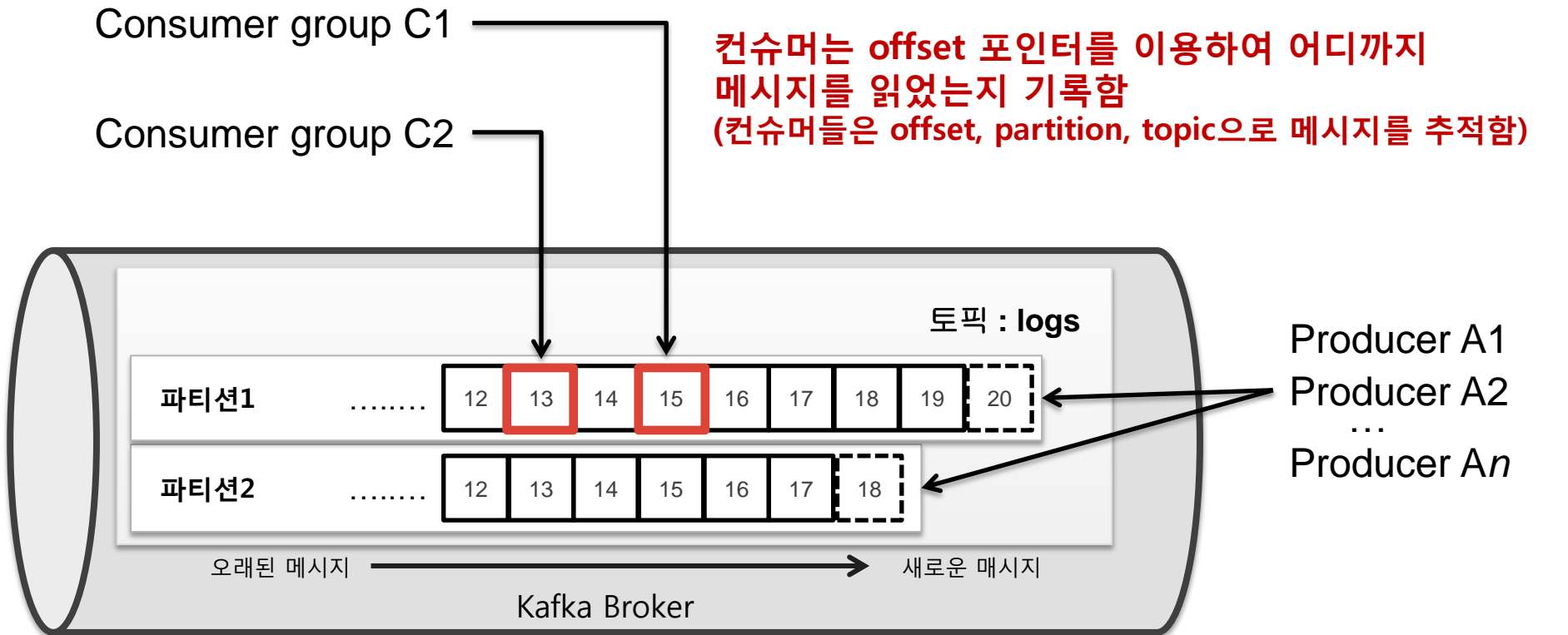
- 토픽은 메시지 파이프라인 이름; 예) logs
- 토픽은 파티션으로 구성
- 파티션은 지속적으로 추가되는 순차적이고 불변적인 일련의 메시지들
- 파티션에 의하여 분산 처리

Age, Max Size, Key 기분으로 머리^{head}부터 메시지제거

메시지는 항상 꼬리^{tail}에 추가

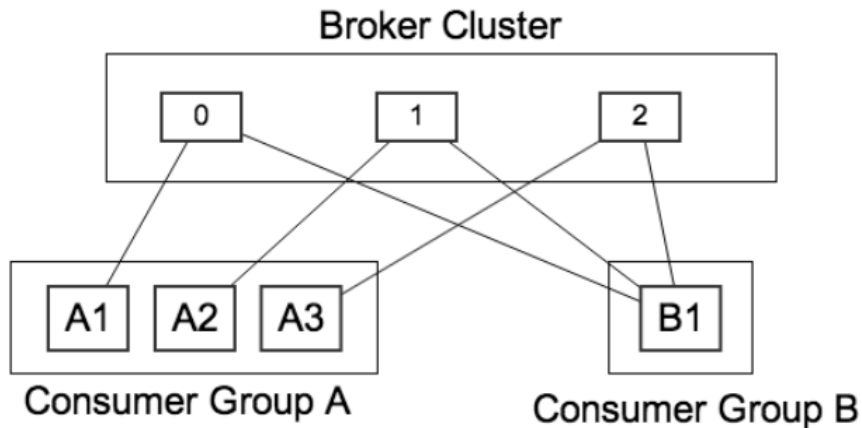


컨슈머의 Offset 포인팅



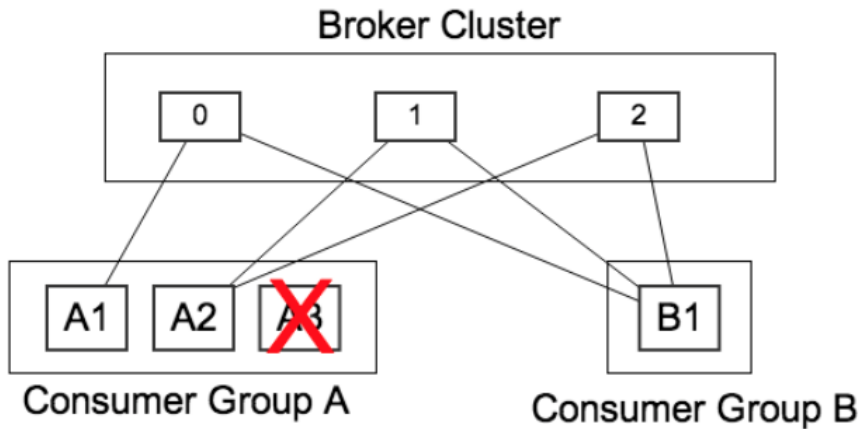
컨슈머와 컨슈머 그룹

- 컨슈머 그룹 개념을 통해 큐^{queue} 모델과 발행-구독^{publish-subscribe} 모델 모두 지원
- 파티션은 컨슈머 그룹당 오로지 하나의 컨슈머 접근만을 허용하고 해당 컨슈머를 파티션 오너라고 부름
- 파티션 내에 메시지 순서에 대하여 보장, 토픽의 다른 파티션의 순서는 보장하지 않음



컨슈머와 컨슈머 그룹

- 컨슈머 스스로 파티션 재분배^{rebalance} 처리



파티션 복제

- **Replicas: 파티션의 “백업”**

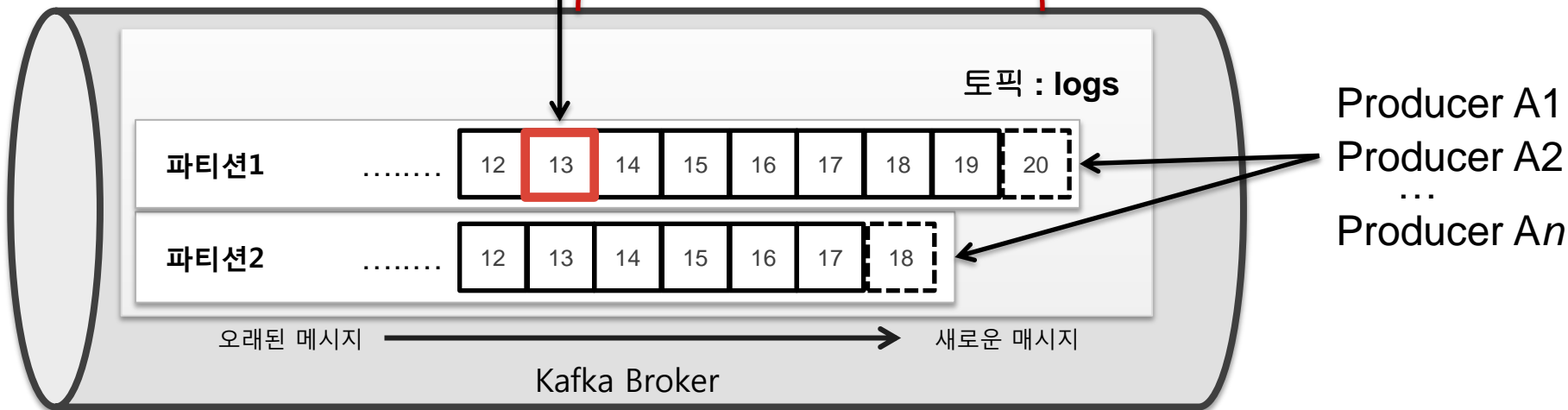
- 데이터 손실을 방지하기 위해 존재
- 복제는 절대 읽지도 쓰지도 않음
 - 프로듀서/컨슈머의 분산 처리를 증가시키지 않음
- 데이터 손실전에 *(numReplicas - 1)* 만큼 죽는 브로커 허용

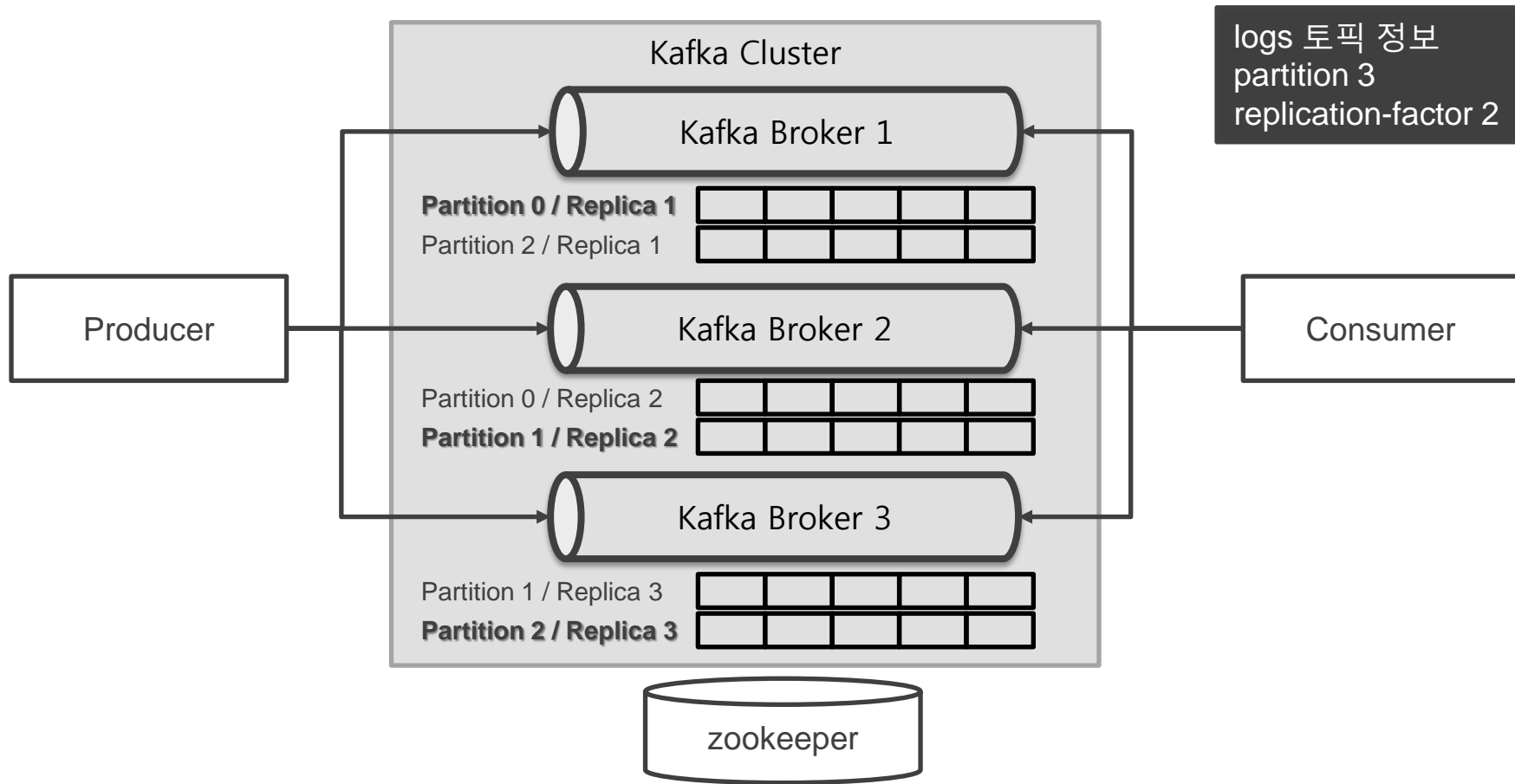
렉 lag

- 렉은 컨슈머 문제에 의하여 발생
- 메시지 소비 속도가 너무 느리거나 주키퍼 또는 카프카 연결이 끊키거나 기타 등등

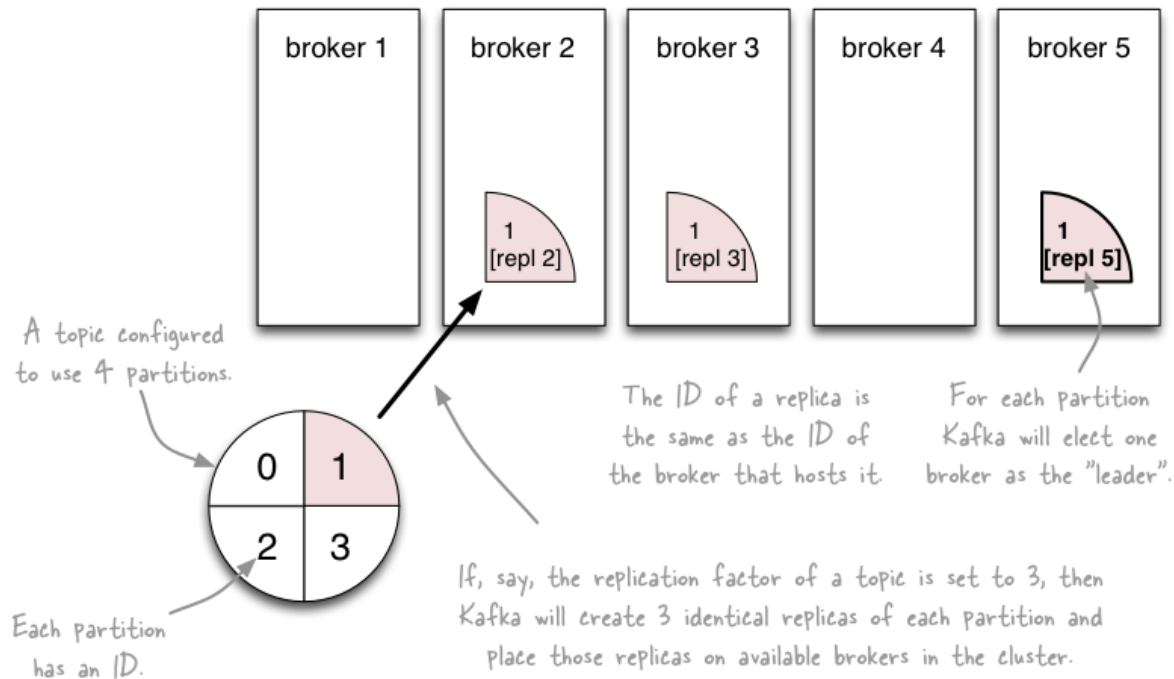
Lag = 프로듀서의 마지막 로그와의 얼마나 떨어진 거리

Consumer group C1





토픽 vs 파티션 vs 복제replicas



이미지 출처 : Running a Multi-Broker Apache Kafka 0.8 Cluster on a Single Node
(<http://www.michael-noll.com/blog/2013/03/13/running-a-multi-broker-apache-kafka-cluster-on-a-single-node/>)

카프카를 사용하는 기업

Companies

- [LinkedIn](#) - Apache Kafka is used at LinkedIn for activity stream data and operational metrics. This powers various products like LinkedIn Newsfeed, LinkedIn Today in addition to our offline analytics systems like Hadoop.
- [Yahoo](#) - See [this](#).
- [Twitter](#) - As part of their Storm stream processing infrastructure, e.g. [this](#) and [this](#).
- [Netflix](#) - Real-time monitoring and event-processing [pipeline](#).
- [Square](#) - We use Kafka as a bus to move all systems events through our various datacenters. This includes metrics, logs, custom events etc. On the consumer side, we output into Splunk, Graphite, Esper-like real-time alerting.
- [Spotify](#) - Kafka is used at Spotify as part of their [log delivery system](#).
- [Pinterest](#) - Kafka is used with [Secor](#) as part of their [log collection pipeline](#).
- [Uber](#)
- [Goldman Sachs](#)
- [Tumblr](#) - See [this](#)
- [PayPal](#) - See [this](#).
- [Box](#) - At Box, Kafka is used for the production analytics pipeline & real time monitoring infrastructure. We are planning to use Kafka for some of the new products & features
- [Airbnb](#) - Used in our event pipeline, exception tracking & more to come.
- [Mozilla](#) - Kafka will soon be replacing part of our current production system to collect performance and usage data from the end-users browser for projects like Telemetry, Test Pilot, etc. Downstream consumers usually persist to either HDFS or HBase.
- [Cisco](#) - Cisco is using Kafka as part of their OpenSOC (Security Operations Center). More detail [here](#).
- [Etsy](#) - See [this article](#).
- [Tagged](#) - Apache Kafka drives our new pub sub system which delivers real-time events for users in our latest game - Deckadence. It will soon be used in a host of new use cases including group chat and back end stats and log collection.
- [Foursquare](#) - Kafka powers online to online messaging, and online to offline messaging at Foursquare. We integrate with monitoring, production systems, and our offline infrastructure, including hadoop.
- [StumbleUpon](#) - Data collection platform for analytics.
- [Coursera](#) - At Coursera, Kafka powers education at scale, serving as the data pipeline for realtime learning analytics/dashboards.
- [Shopify](#) - Access logs, A/B testing events, domain events ("a checkout happened", etc.), metrics, delivery to HDFS, and customer reporting. We are now focusing on consumers: analytics, support tools, and fraud analysis.
- [Cerner](#) - Kafka is used with HBase and Storm as described [here](#).
- [Oracle](#) - Oracle provides native connectivity to Kafka from its Enterprise Service Bus product called OSB (Oracle Service Bus) which allows developers to leverage OSB built-in mediation capabilities to implement staged data pipelines.
- [CloudFlare](#) - CloudFlare uses Kafka for our log processing and analytics pipeline, collecting hundreds of billions of events/day data from a thousands of servers.
- [Mate1.com Inc.](#) - Apache kafka is used at Mate1 as our main event bus that powers our news and activity feeds, automated review systems, and will soon power real time notifications and log distribution.
- [Boundary](#) - Apache Kafka aggregates high-flow message streams into a unified distributed pubsub service, brokering the data for other internal systems as part of Boundary's real-time network analytics infrastructure.

Quick Start

다운로드

https://www.apache.org/dyn/closer.cgi?path=/kafka/0.8.2.0/kafka_2.10-0.8.2.0.tgz

```
> tar -xzf kafka_2.10-0.8.2.0.tgz  
> cd kafka_2.10-0.8.2.0
```

주키퍼 서버 구동

```
> bin/zookeeper-server-start.sh config/zookeeper.properties
```

카프카 서버 구동

```
> bin/kafka-server-start.sh config/server.properties
```

토픽 생성

```
> bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2  
--partitions 3 --topic logs
```

토픽 목록 조회

```
> bin/kafka-topics.sh --list --zookeeper localhost:2181
```

메시지 보내기

```
> bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test
```

메시지 받기

```
> bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning
```


아파치 카프카 기반 메시지 전송/수신

메시지 전송/수신 하기

- 카프카 프로듀서 SDK를 이용 (공식적으로 Java, Scala)
- 별도 오픈소스 SDK 존재
(<https://cwiki.apache.org/confluence/display/KAFKA/Clients#Clients-HowTheKafkaProjectHandlesClients>)
- For 0.8.x
 - Producer Daemon
 - Python
 - Go (AKA golang)
 - C/C++
 - .net
 - Clojure
 - Ruby
 - Node.js
 - Alternative Java Clients
 - Storm
 - Scala DSL
 - HTTP REST
 - JRuby
 - Perl
 - stdin/stdout
 - Erlang
 - PHP
 - Rust

Java SDK 사용

Maven POM에 dependency 추가

```
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka_2.10</artifactId>
  <version>0.8.2.0</version>
  <exclusions>
    <exclusion>
      <artifactId>slf4j-log4j12</artifactId>
      <groupId>org.slf4j</groupId>
    </exclusion>
    <exclusion>
      <artifactId>log4j</artifactId>
      <groupId>log4j</groupId>
    </exclusion>
  </exclusions>
</dependency>
```



스프링 slf4j와 logback 사용시 추가

Java 프로듀서 API

- 신규 Producer API와 기존 Scala API 존재
- 기존 Scala API

```
1 class kafka.javaapi.producer.Producer<K,V>
2 {
3     public Producer(ProducerConfig config);
4
5     /**
6      * Sends the data to a single topic, partitioned by key, using either the
7      * synchronous or the asynchronous producer.
8      */
9     public void send(KeyedMessage<K,V> message);
10
11     /**
12      * Use this API to send data to multiple topics.
13      */
14     public void send(List<KeyedMessage<K,V>> messages);
15
16     /**
17      * Close API to close the producer pool connections to all Kafka brokers.
18      */
19     public void close();
20 }
```

Java 프로듀서 API

- 신규 Producer API

Constructor Summary

[KafkaProducer](#)(java.util.Map<java.lang.String,java.lang.Object> configs)

A producer is instantiated by providing a set of key-value pairs as configuration.

[KafkaProducer](#)(java.util.Map<java.lang.String,java.lang.Object> configs, [Serializer](#)<K> keySerializer, [Serializer](#)<V> valueSerializer)

A producer is instantiated by providing a set of key-value pairs as configuration, a key and a value [Serializer](#).

[KafkaProducer](#)(java.util.Properties properties)

A producer is instantiated by providing a set of key-value pairs as configuration.

[KafkaProducer](#)(java.util.Properties properties, [Serializer](#)<K> keySerializer, [Serializer](#)<V> valueSerializer)

A producer is instantiated by providing a set of key-value pairs as configuration, a key and a value [Serializer](#).

Method Summary

void	close () Close this producer.
java.util.Map< MetricName ,? extends Metric >	metrics () Return a map of metrics maintained by the producer
java.util.List< PartitionInfo >	partitionsFor (java.lang.String topic) Get a list of partitions for the given topic for custom partition assignment.
java.util.concurrent.Future< RecordMetadata >	send (ProducerRecord <K,V> record) Asynchronously send a record to a topic.
java.util.concurrent.Future< RecordMetadata >	send (ProducerRecord <K,V> record, Callback callback) Asynchronously send a record to a topic and invoke the provided callback when the send has been acknowledged.

Java 프로듀서 개발

- 기존 Producer API 사용

```
14 @Component
15 public class DemoProducer {
16
17     private Producer<String, String> producer;
18
19     public DemoProducer() {
20         Properties props = new Properties();
21         props.put("metadata.broker.list",
22             "localhost:9092,localhost:9093,localhost:9094");
23         props.put("serializer.class", "kafka.serializer.StringEncoder");
24         ProducerConfig producerConfig = new ProducerConfig(props);
25         producer = new Producer<String, String>(producerConfig);
26     }
27
28     public void send() {
29         KeyedMessage<String, String> message = new KeyedMessage<String, String>("sample", "Hello, World!");
30         producer.send(message);
31         producer.close();
32     }
33
34 }
```

상세 설정 : <http://kafka.apache.org/documentation.html#producerconfigs>

Java 프로듀서 개발

주요 설정

- `metadata.broker.list`
메타데이터를 받아올 카프카 브로커 리스트 (호스트:포트,...)
- `request.required.acks`
Ack 여부 (0, 1, -1) 기본값 1(리더 파티션이 받음 확인)
- `producer.type`
동기, 비동기 프로듀서 설정 (sync, async) 기본값 sync
- `serializer.class`
메시지 직렬화 인코더 설정(기본 byte[] 그대로 전달)
- `partitioner.class`
메시지를 어떤 파티션에 전송할지 결정하는 클래스 기본은 메시지 키의 해시 코드를 기반

** 신규 `Producer API`는 설정 키가 변경 되었음*

상세 설정 : <http://kafka.apache.org/documentation.html#producerconfigs>

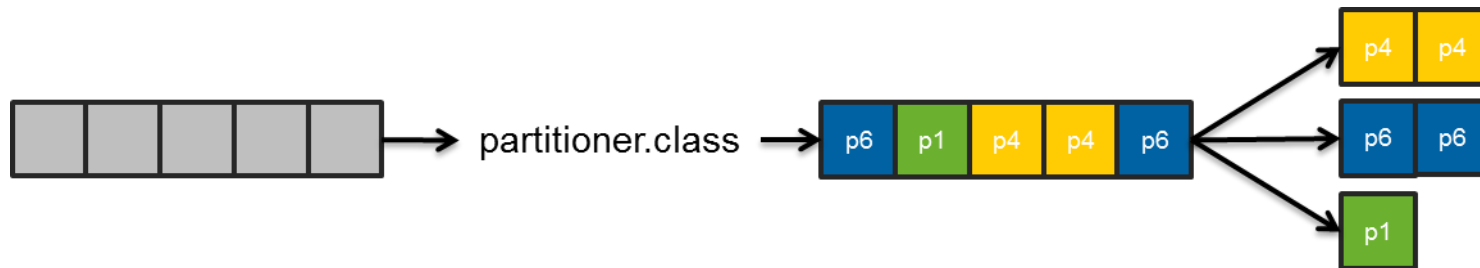
Java 프로듀서 개발 - 메시지 벌크 처리

1. send(listOfMessages)

```
1 producer.send(List<KeyedMessage<K,V>> messages);
```

동기 프로듀서 : 메시지 리스트를 동기로 보냄; 블록

비동기 프로듀서 : 백그라운드에서 메시지를 비동기로 처리; 블록 안됨

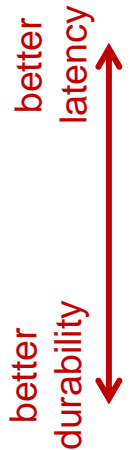


2. send(singleMessage) 비동기 프로듀서

```
1 producer.send(KeyedMessage<K,V> message);
```

비동기 프로듀서 경우 : send(listOfMessages) 와 같음

Java 프로듀서 개발 - 메시지 ack 처리

- 
- request.required.acks 설정
 - **0**: 프로듀서는 브로커의 ack를 기다리지 않음
 - **1**: 리더 브로커가 데이터를 받으면 프로듀서 ack 받음
 - **-1**: 프로듀서는 모든 메시지가 복제되면 ack를 받음

- request.timeout.ms

프로듀서 클라이언트에게 에러를 보내기 전까지 'request.required.acks'를 만족하기 위해 브로커가 기다리는 시간

Java 컨슈머 API

- High-level 컨슈머 API와 Simple Consumer API 존재
- 컨슈머는 브로커로부터 메시지를 땡겨옴^{pull} 받지않음^{not push}
- 컨슈머는 스스로 어디까지 읽었는지 관리해야함 (offset 관리)
- High-level 컨슈머는 offset을 Zookeeper를 통하여 관리함
- Simple 컨슈머는 스스로 관리하는 것을 구현해야 함

주요 설정

group.id

zookeeper.connect

간단한 데모 보기

References

- <http://kafka.apache.org/documentation.html>
- <http://www.slideshare.net/charmalloc/developingwithapachekafka-29910685>
- <https://cwiki.apache.org/confluence/display/KAFKA/Index>
- <http://www.michael-noll.com/blog/2013/03/13/running-a-multi-broker-apache-kafka-cluster-on-a-single-node/>
- <http://www.confluent.io/blog/stream-data-platform-1/>
- <http://www.slideshare.net/charmalloc/developing-with-the-go-client-for-apache-kafka>
- <http://www.slideshare.net/miguno/apache-kafka-08-basic-training-verisign>
- <http://www.slideshare.net/charmalloc/current-and-future-of-apache-kafka>
- <http://epicdevs.com/17>
- <http://epicdevs.com/21>

GiGA IoT Makers 소개

GiGA IoTMakers란

IoT 서비스를 스스로 만들 수 있는 공간으로 창조적인 Makers를 지원하는 개방형 IoT플랫폼



GiGA

kt의 고성능의 네트워크



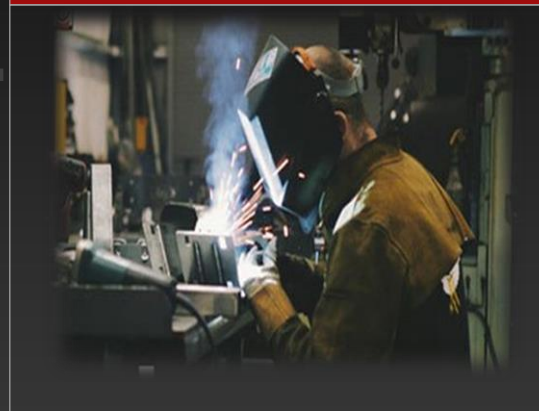
IoT

인터넷에 연결된 다양한 사물



Makers

IoT의 창조적인 제작자



IoTMakers 특징점



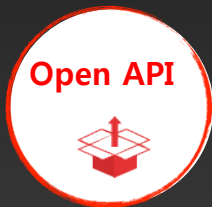
손쉬운 디바이스 연결

KT 표준 인터페이스, Java, C SDK*, 글로벌 oneM2M 가이드
비표준 어댑터 개발 GUI, 템플릿 제공, 대시보드를 통한 수집/제어 실시간 확인



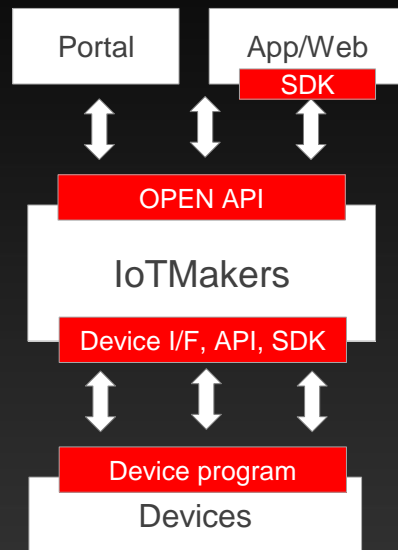
실시간 IoT 데이터 및 지능형을 처리

실시간 메시지 처리 및 분산 아키텍처로 안정적인 IoT 데이터 처리
소스 프로그램 없이 포털에서 지능형 룰 편집 및 동적 적용



IoT 서비스 개발 지원

OPEN API를 통하여 모바일 앱, 웹 등의 개발 지원
공개 디바이스 및 공개 앱 소스를 활용한 IoT 서비스 개발



IoT Makers 구성

Device connection
& management

KT 표준, 글로벌 프로토콜 적용

Intelligent Rule Engine &
Complex Event processing

실시간 대규모 데이터,
지능형 Rule Engine

OPEN API based
IoT Service creation

OPEN API 기반의 IoT 서비스 구축

Any devices

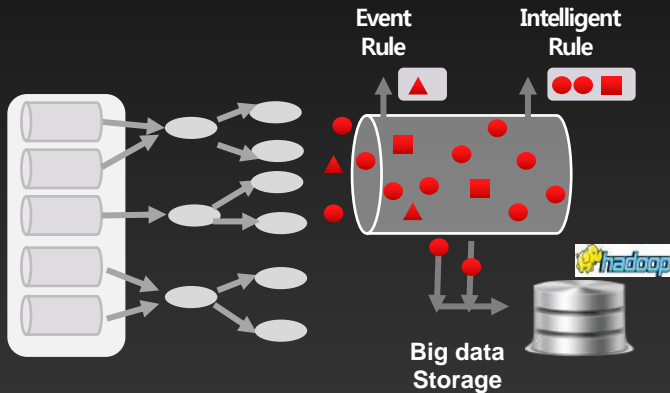


수집

Connection
Layer

(KT 표준,
글로벌
프로토콜
수용)

제어



OPEN APIs



GiGA IoT Makers 시연

GiGA IoTMakers (<http://iotmakers.olleh.com>)

GiGA IoTMakers | IoTMakers 소개 | IoT 개발 | OPEN API | 공개 디바이스 | 공개 앱 | IoTMakers 가이드 | 고객센터

초보자라도 누구나 IoTMakers를 통해 IoT 세상을 만날 수 있습니다.

IoTMakers 가이드

쉬운 디바이스 연동

IoTMakers는 모든 코드 없이도 인터넷 디바이스를 손쉽게 연동할 수 있도록 지원합니다.

- 표준 통신 프로토콜 지원
- 개인 디바이스 사용자의 자유
- 기존 서비스와의 연동 지원

IoT 데이터 및 이벤트 관리

통합된 디바이스에서 수집된 데이터와 이벤트를 한 장에서 관리합니다. 또한 다양한 이벤트를 설정하여 디바이스를 제어할 수 있습니다.

- 온디 가변 이벤트 및 비동기화 지원
- 분석도구를 통한 분석

어플리케이션 개발 지원

IoT 서비스 개발을 지원하여 IoT 애플리케이션을 쉽고 쉽게 개발할 수 있는 환경을 제공합니다.

- Open API를 통한 앱 개발 지원
- Markup IoT 개발 지원

GiGA IoTMakers™ | IoTMakers 소개 | IoT 개발 | OPEN API | 공개 디바이스 | 공개 앱 | IoTMakers 가이드 | 고객센터

IoTMakers® Showcase

IoT(Internet of Things) 란?

IoT (Internet of Things)란, 사물들의 인터넷으로 실 세계와 가상 세계에 존재하는 사물들을 네트워크로 연결하여, 사람과 사람, 사람과 사물간에 언제 어디서나 서로 소통할 수 있는 지능적 환경입니다.

인터넷의 탄생으로 수많은 새로운 비즈니스가 생겨나고 산업이 변화하였듯이 모든 사물이 연결된 IoT시대에는 지금은 상상하지 못하는 새로운 기술들이 다가가고 있습니다.

olleh GiGA IoTMakers 이란?

IoTMakers는 IoT Player들이 쉽게 IoT생태계에 참여할 수 있도록 돕기 위한 IoT 개발 플랫폼입니다.

IoTMakers를 통해서 손쉽게 IoT 디바이스를 연결하여 테스트 할 수 있고, 수집된 데이터를 관리할 수 있고, 제공되는 OPEN API를 통하여 IoT 서비스를 만들 수 있습니다.

KT는 IoTMakers를 통해 다양한 B2C, B2B, B2G 서비스를 제공할 예정입니다. 이제는 이러한 속성된 노하우를 바탕으로 다양한 개발자, 스타트업, 중소기업들이 플랫폼에 투자하지 않고도 신속하게 서비스 개발을 실현하여 IoT의 창조적인 Makers가 되도록 지원하겠습니다.

<thank-you>



+jead0.ko
haibane84@gmail.com