CDC(Change Data Capture) 오픈 소스
Debezium 쓸까? 말까?

카카오/카카오커머스/김용환(samuel.c)

Debezium 발음과 유래

유래 : DBs + ium (주기율표의 원소이름에 붙이는 접미사
(Dbs-ium)
=> Dee-BEE-zee-uhm
=> Debezium

발음 : /디비-지:움/

발표 내용

1. 발표 배경
2. CDC
3. Debezium
4. 데모
5. 특징과 팁

If Kakao 2019 세션 질문을 받았습니다

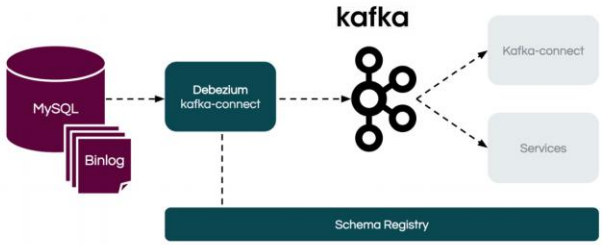
발표 배경

If Kakao 2019 – 밑바닥부터 시작하는 쇼핑 데이터 엔지니어링 고군 분투기 (@tasha.j)발표

배치 처리를 통한 데이터 변경 감지 솔루션

if(kakao)dev2019

DB 가 변경 이력을 push 해주는 방식



CDC (Change Data Capture) 는
- 다양한 기술스택의 운영 노하우가 필요
- 테이블 스키마에 대한 변경 제약
- master/slave 변경 시 bin log 핸들링 이슈

DB 의 증분 필드를 pull 하는 방식

Incremental ingest

So far we've just pulled entire tables into Kafka on a scheduled basis. This is useful
the data, but very batchy and not always so appropriate for actually integrating
systems into the streaming world of Kafka.

The JDBC connector gives you the option to stream into Kafka just the rows from
changed in the period since it was last polled. It can do this based either on an in
(e.g., incrementing primary key) and/or a timestamp (e.g., last updated timestamp
A common practice in schema design is to have one or both of these present. For exa
transaction table such as `ORDERS` may have:

- `ORDER_ID`, a unique key (maybe the primary key) that increments for each new order
- `UPDATE_TS`, a timestamp column that is updated every time the row is *updated*

To specify which option you want to use, set the `<mode` option. Let's switch to `timestamp`:

```
curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d '{
        "name": "jdbc_source_mysql_08",
        "config": {
                "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
                "connection.url": "jdbc:mysql://mysql:3306/demo",
                "connection.user": "connect_user",
                "connection.password": "asgard",
                "topic.prefix": "mysql-08-",
                "mode":"timestamp",
                "table.whitelist" : "demo.accounts",
                "timestamp.column.name": "UPDATE_TS",
                "validate.non.null": false
                }
        }'
```

CDC(Debezium) 기술이 궁금한데 더 알려주실 수 있나요?

발표 배경

DB의 증분 필드를 pull하는 방식(Incremental Ingest)

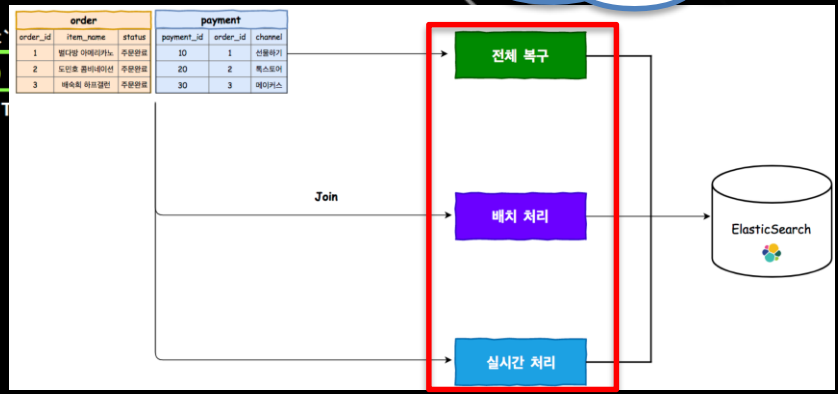
기존 스키마

```
CREATE TABLE `order` (
  `order_id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT 'PK ID',
  `item_name` varchar(255) NOT NULL COMMENT '상품명',
  `status` int(11) NOT NULL COMMENT '주문상태',
  `channel` varchar(50) NOT NULL COMMENT '채널명',
  `created_at` datetime NOT NULL COMMENT '등록일',
  `modified_at` datetime DEFAULT NULL COMMENT '수정일',
PRIMARY KEY (`order_id`),
  KEY `idx_created_at` (`created_at`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4;
```

변경된 스키마

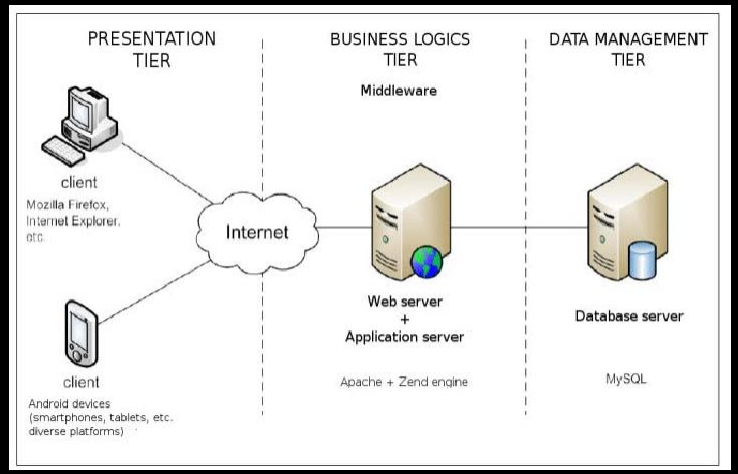
```
CREATE TABLE `order` (
  `order_id` bigint(20) unsigned NOT NULL AUTO_INCREMENT COMMENT 'PK ID',
  `item_name` varchar(255) NOT NULL COMMENT '상품명',
  `status` int(11) NOT NULL COMMENT '주문상태',
  `channel` varchar(50) NOT NULL COMMENT '채널명',
  `created_at` datetime NOT NULL COMMENT '등록일',
  `modified_at` datetime DEFAULT NULL COMMENT '수정일',
  `update_ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT 'ROW 변경 시점 마킹',
PRIMARY KEY (`order_id`),
  KEY `idx_created_at` (`created_at`
  KEY `idx_update_ts` (`update_ts`)
) ENGINE=InnoDB AUTO_INCREMENT
```

Delete?!



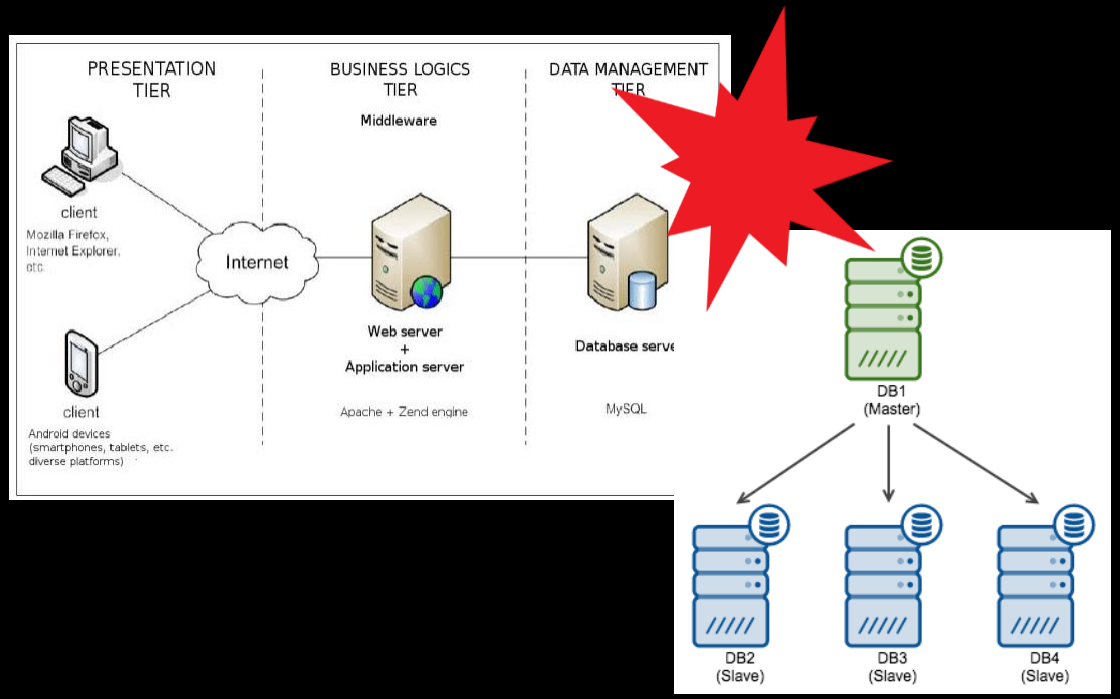
왜 Debezium에 관심을 가졌을까요?

발표 배경



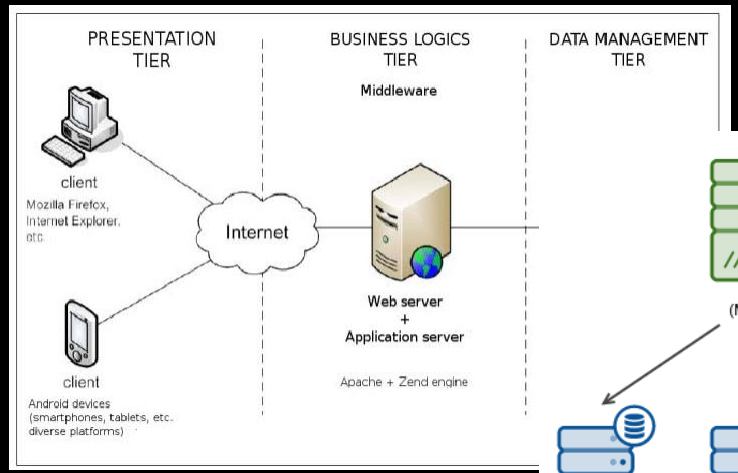
<처음 시작>
데이터와 로그(history)를 모두
DB에 저장

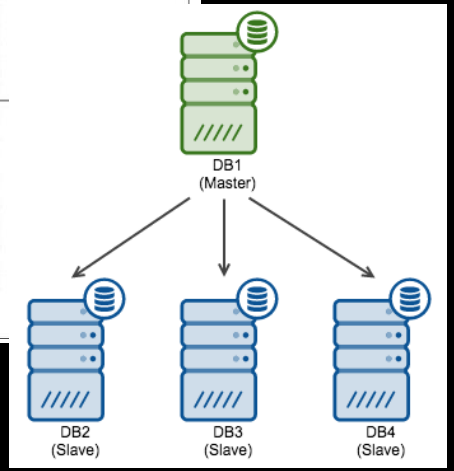
발표 배경



<Load Balancer, HA>
mysql LB.MHA 적용

발표 배경

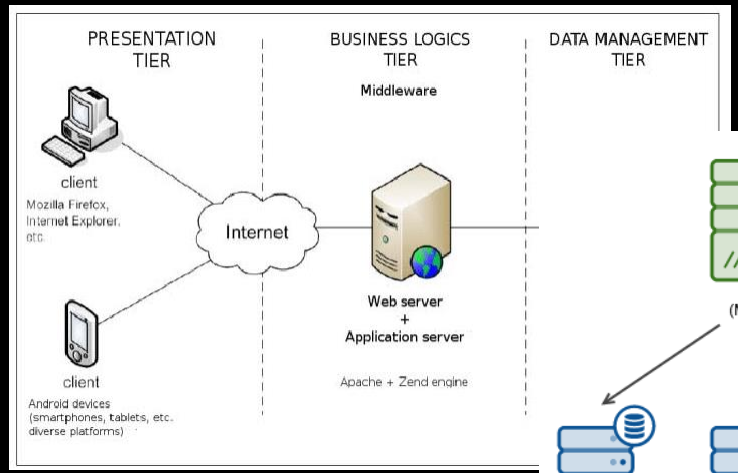


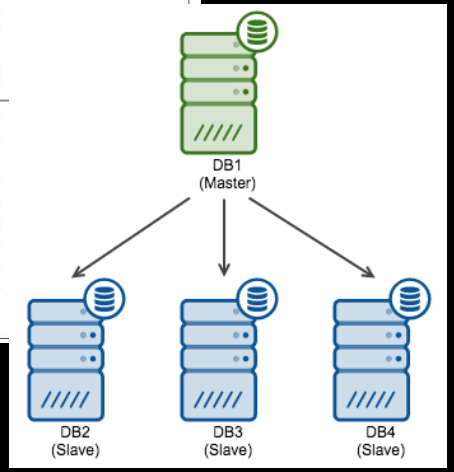


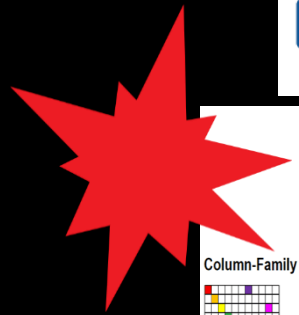
ETL(Spark, Impala, Hive, MapReduce…)

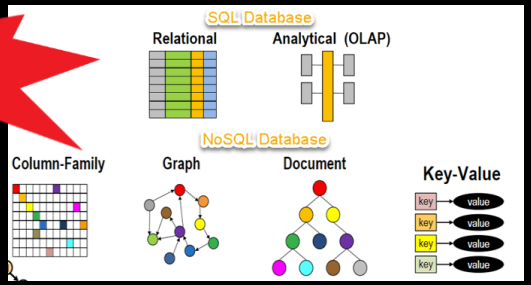
로그(history) 데이터를 HDFS에 저장 및 처리,
데이터 플랫폼 용으로 DB 데이터를
HDFS에 저장 및 처리

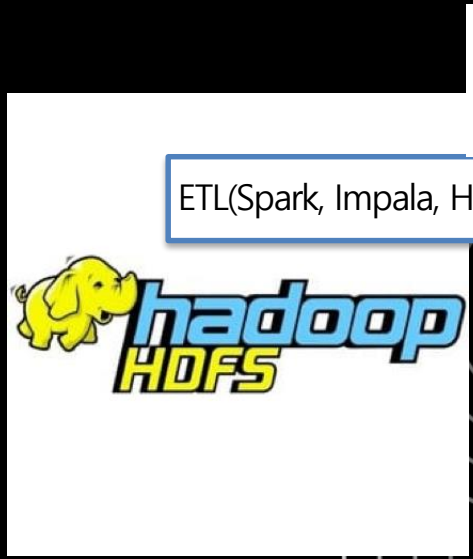
발표 배경





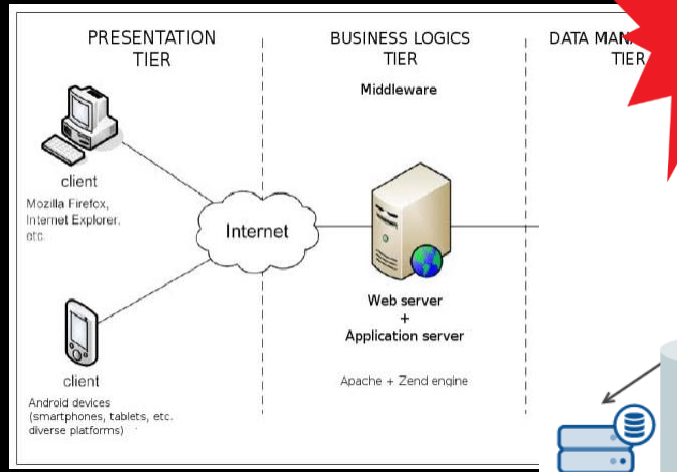


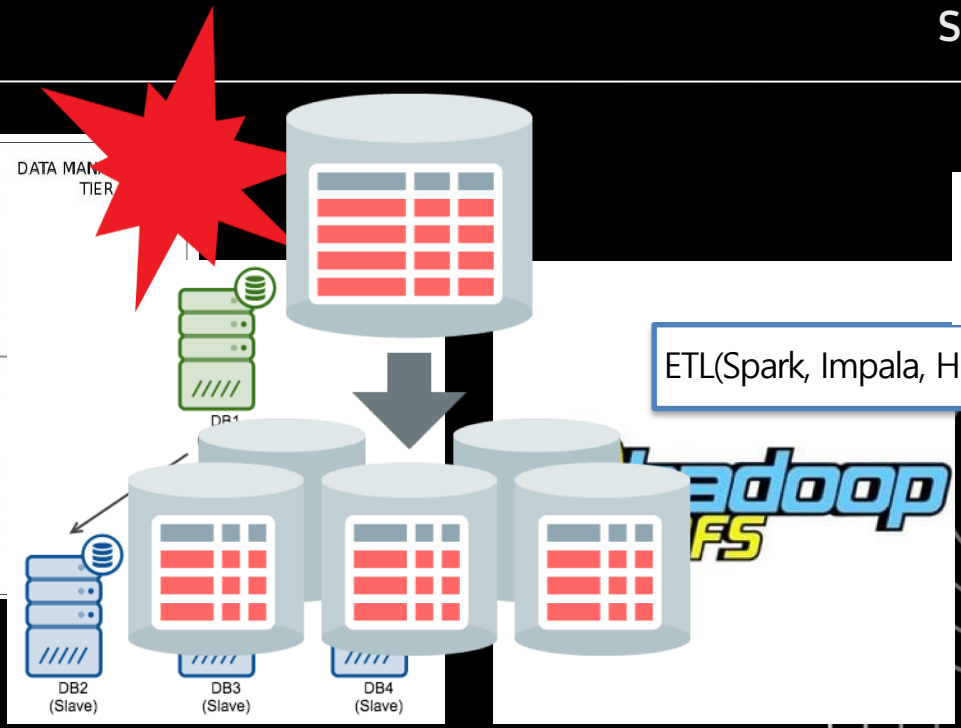


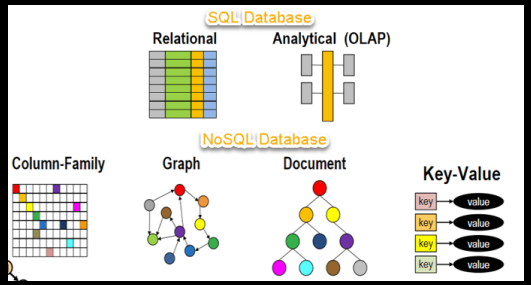


NoSQL(Cache) 이용
- Redis, Memcached, Elasticsearch, Cassandra, Mongodb, Hbase....

발표 배경

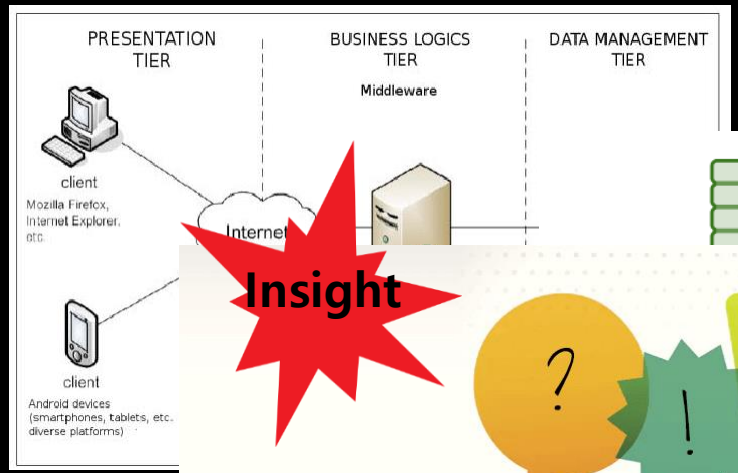




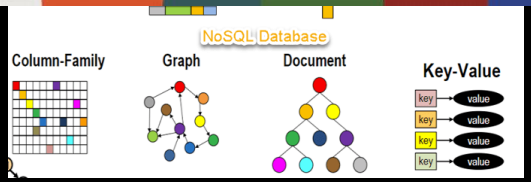


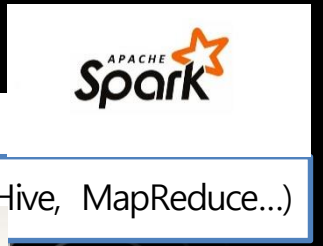
One DB -> Sharding DB

발표 배경


PRESENTATION TIER
BUSINESS LOGICS TIER
Middleware
DATA MANAGEMENT TIER
client
Mozilla Firefox, Internet Explorer, etc.
Internet
client
Android devices (smartphones, tablets, etc. diverse platforms)

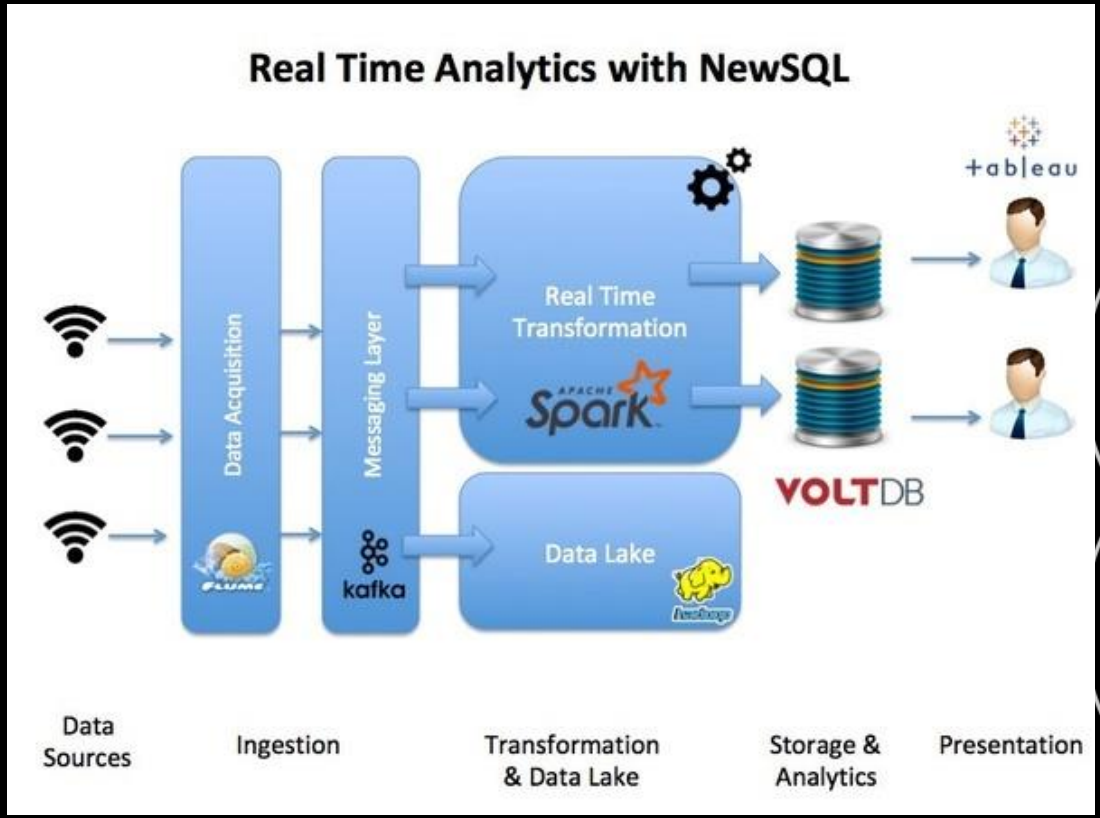
Insight
?
!
[• • •]
:-)
#_ _ _
LOL


NoSQL Database
Column-Family
Graph
Document
Key-Value
key
value


APACHE
Spark
ETL (Spark, Impala, Hive, MapReduce…)

요즘 아키텍처 트렌드

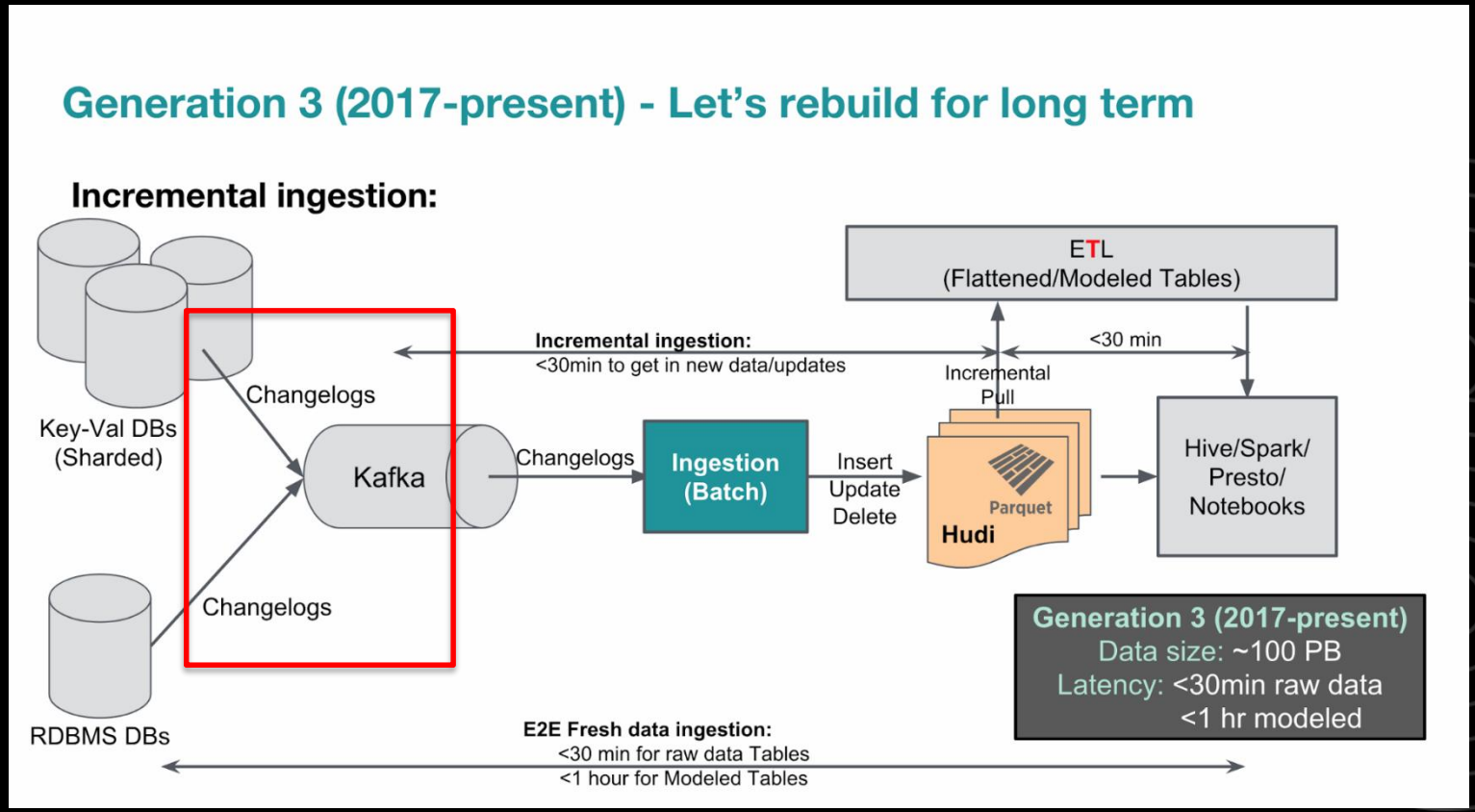
발표 배경



UBS 투자 은행

출처 :https://www.quora.com/How-Kappa-Architecture-can-be-improved

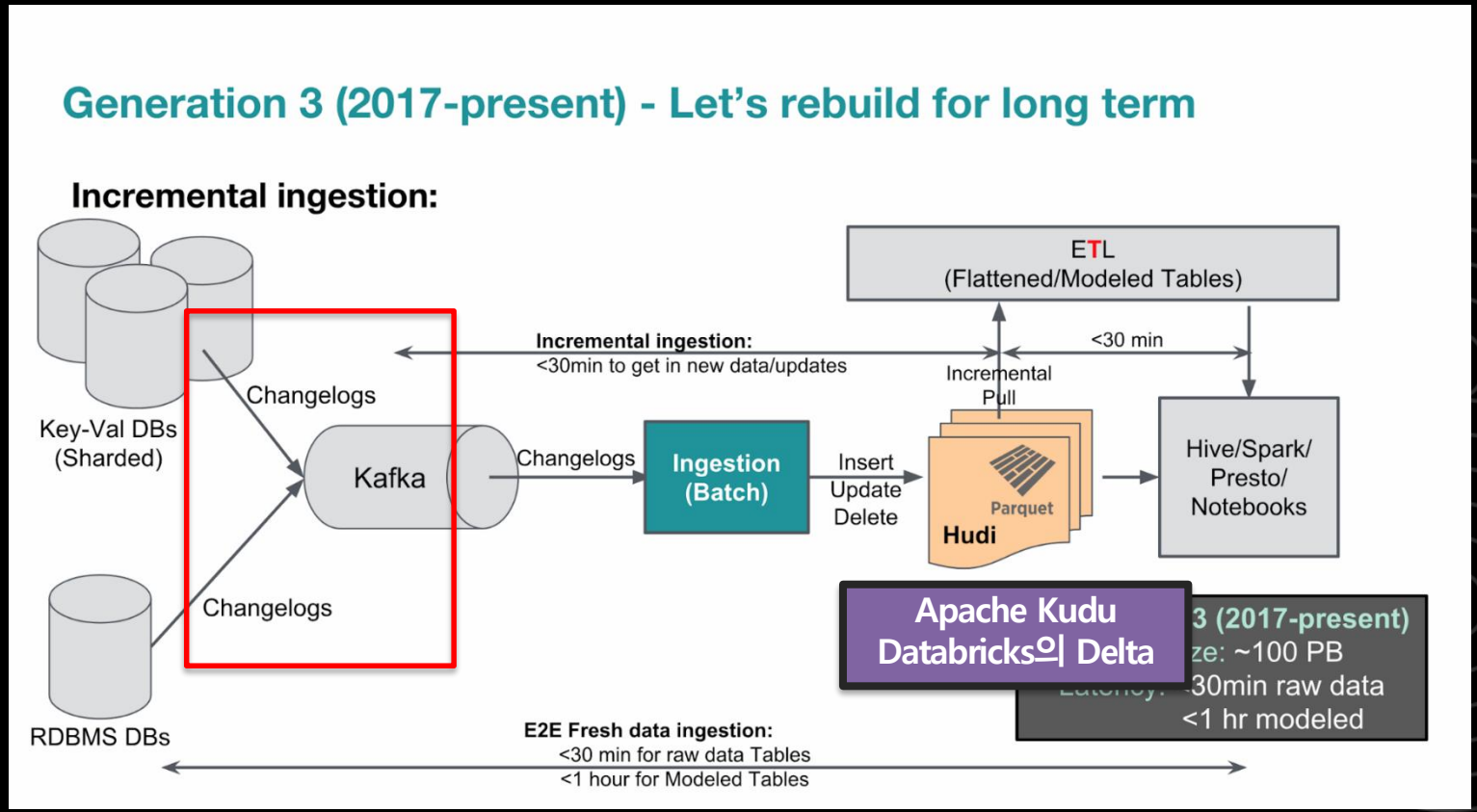
발표 배경



우버 빅 데이터 플랫폼

출처 : https://eng.uber.com/uber-big-data-platform/

발표 배경

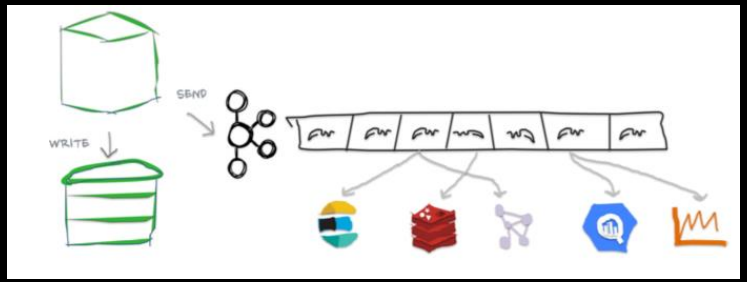


우버 빅 데이터 플랫폼

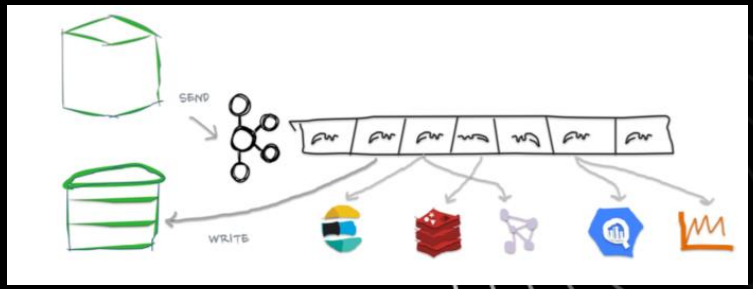
출처 : https://eng.uber.com/uber-big-data-platform/

발표 배경

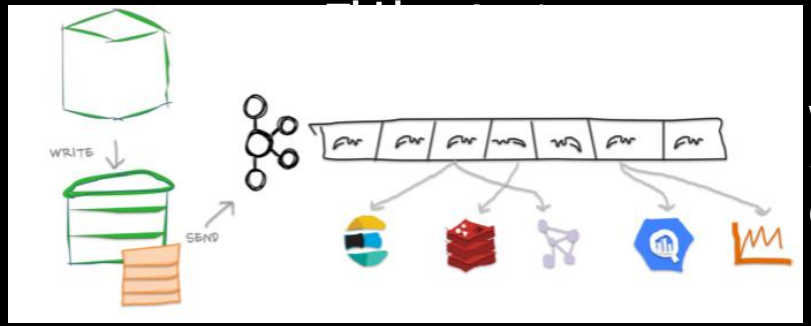
양쪽 쓰기(Double writing)



이벤트 저장소로 카프카 사용



커밋 로그



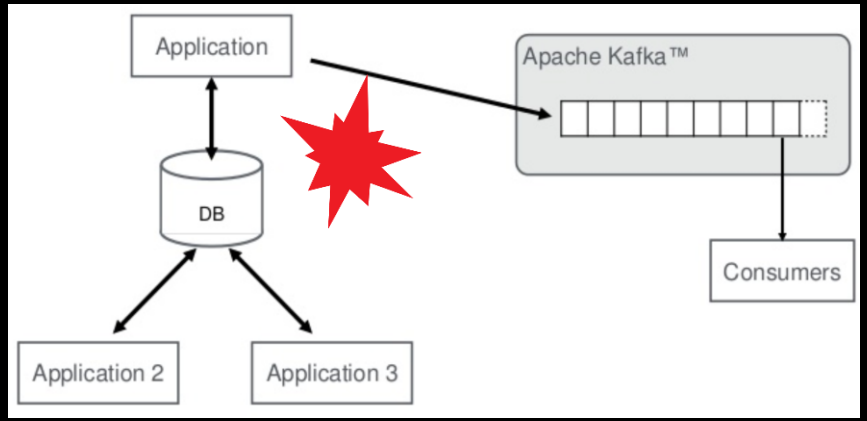
Wepay에서 debezium mysql을 적용하고 debezium Cassandra를 사용 중임

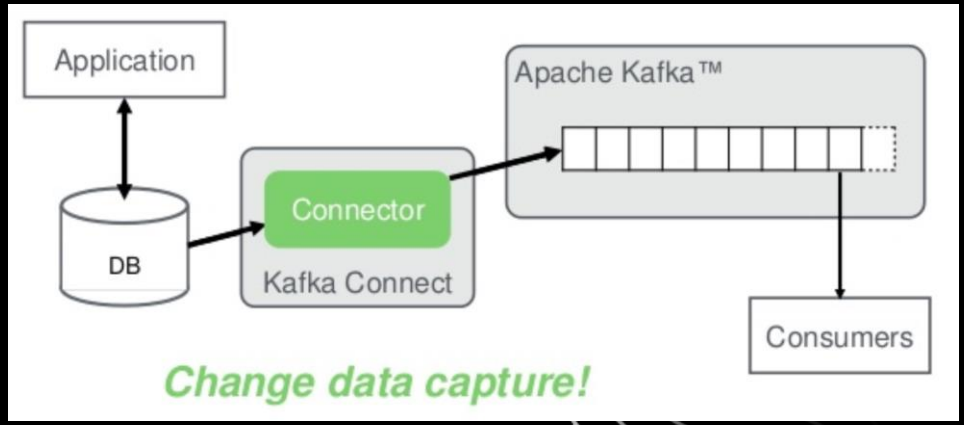
출처 : https://debezium.io/blog/2019/07/12/streaming-cassandra-at-wepay-part-1/

Debezium의 플랫폼인
카프카 커넥트(Kafka Connect)

카프카를 지원하는
Confluent의 오픈소스 전략!

발표 배경



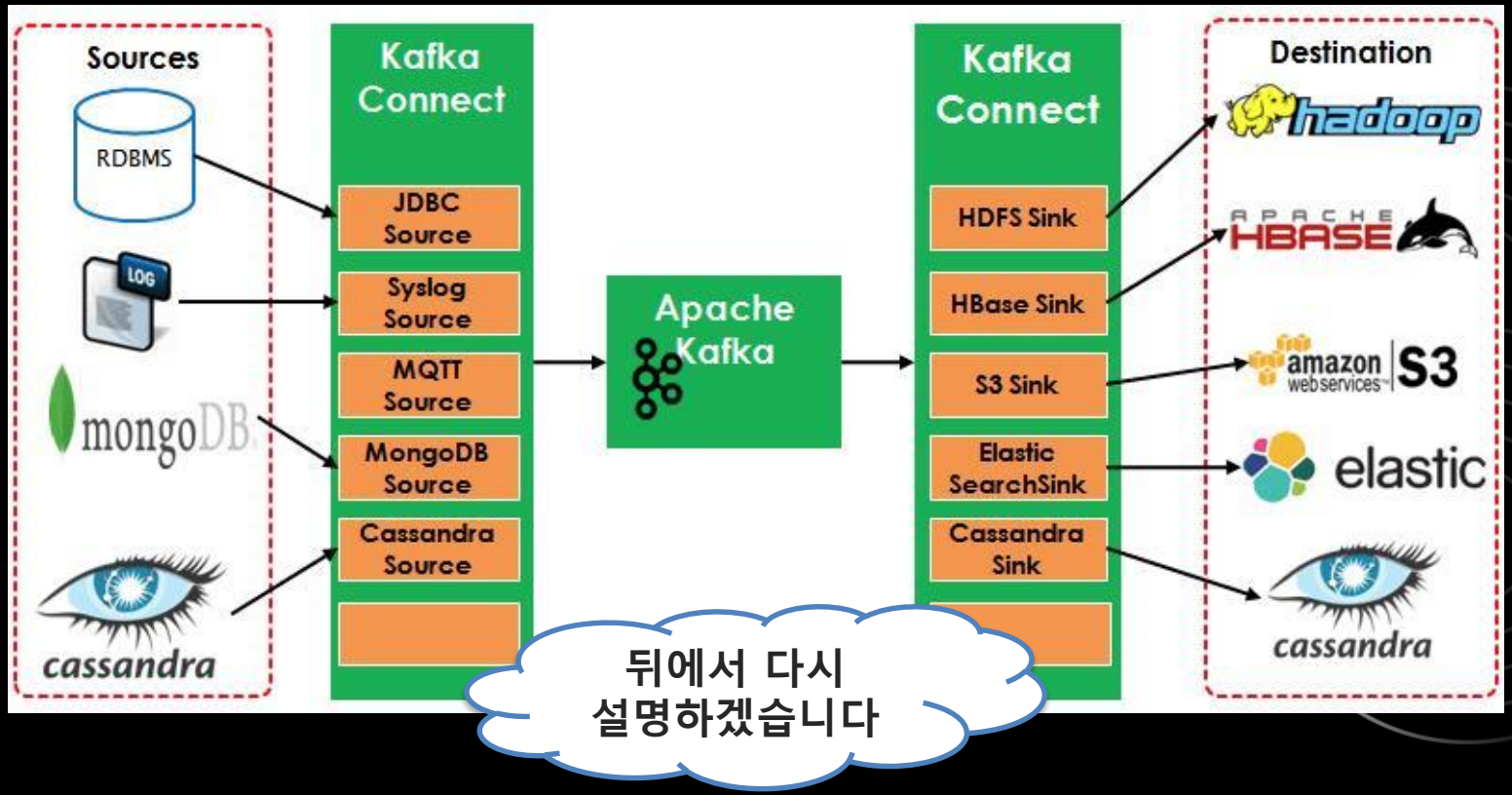


<일반적인 방식>
애플리케이션 또는 Fluentd와
같은 수집 툴을 사용해
로그를 카프카로 전달한다.
또는 DB데이터와 동일한
데이터를 카프카로 전달한다.

<CDC 방식>
CDC를 통해 데이터를
카프카에 전달한다

발표 배경

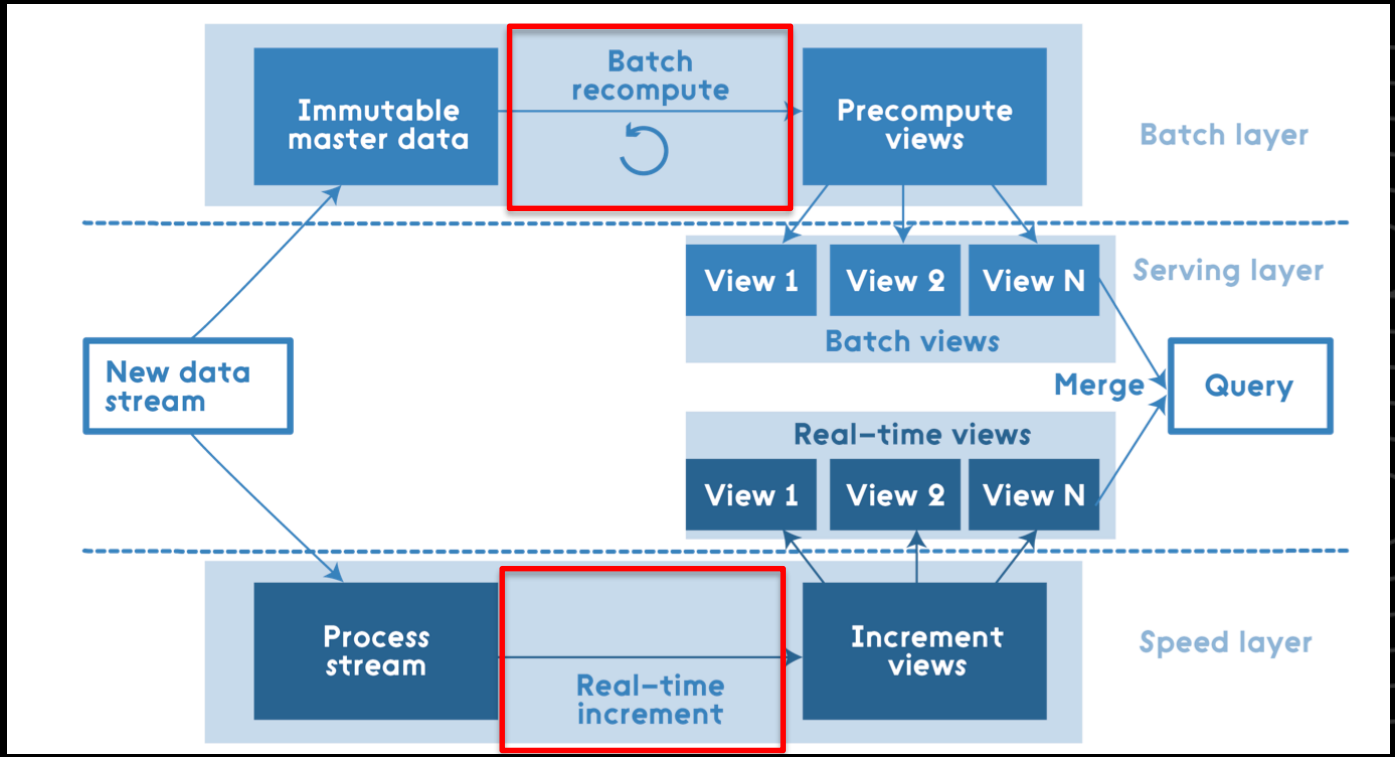
카프카 지원 벤더인 Confluent 사가 지원하는 오픈소스 –카프카 커넥트



점차 Lambda 아키텍처에서 Kappa 아키텍처로 이동 중..

발표 배경

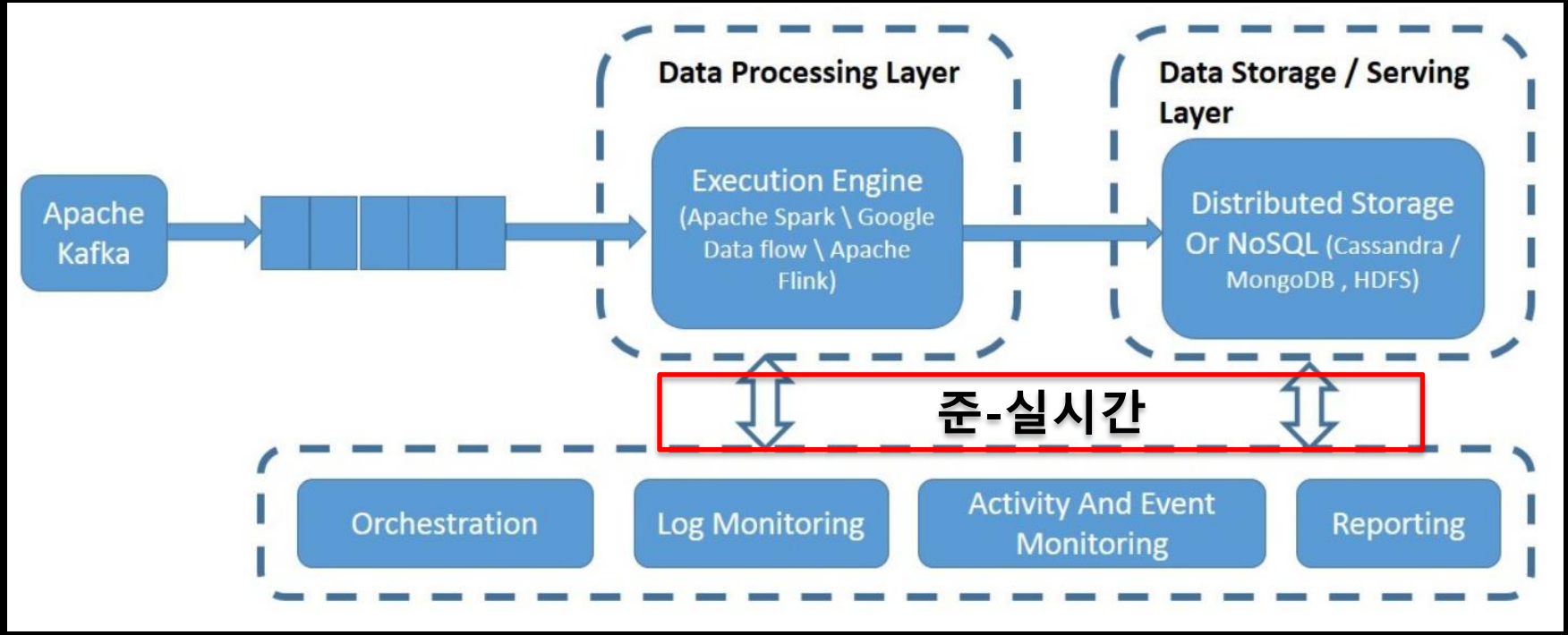
<Lamdba 아키텍처>



출처 : http://jonboulineau.me/blog/architecture/kappa-architecture

발표 배경

<Kappa 아키텍처>



스토리지 데이터 변경 내용만
쉽게 파악할 수 있다면
몸이 편할텐데..
=> CDC 알아볼까?

CDC 정의- 위키피디아

In databases, Change Data Capture (CDC) is a set of software design patterns used to determine (and track) the data that has changed so that action can be taken using the changed data.

데이터베이스의 CDC (Change Data Capture)는 변경된 데이터를 사용해 액션(Action)이 취하도록 변경된 데이터를 판별 및 추적하는 데 사용되는 소프트웨어 설계 패턴이다.

CDC– The why?

- 일반적으로 서비스 초기에 데이터는 데이터 베이스에 저장된다
- 실시간 처리가 가능하다
- 새벽마다 통계 및 분석 를 위한 대량 배치 작업을 줄일 수 있다
- 데이터 변경분만 전송되기에 훨씬 효율적이고 필요한 자원이 줄어 든다

CDC 오픈소스

- Mysql의 경우 엄청 많다 - Maxwell, NIFI, Alibaba의 canal 등등
- Mysql bin 로그 파서는 shyiko bin로그 파서가 https://github.com/shyiko/mysql-binlog-connector-java가 유명하다
- Debezium(mysql), NIFI 내부에서 shyiko/mysql-binlog-connector-java 파서를 사용한다

Debezium

- 카프카 커넥트 기반의 오픈 소스 플러그인(http://debezium.io/)
- 아파치 2.0 라이선스
- Red Hat에서 지원
- Trivago, Wepay, Yotpo, BlaBlaCar에서 적용되어 있다

Debezium

- 논리적 복제를 사용해 변경 스트림을 아파치 카프카 토픽에 복제한다
- 데이터 덤프 + CDC 기능
- PostgreSQL, MongoDB, MySQL, Oracle, Amazon RDS, SQL Server, Cassandra를 지원한다.

  Cassandra와 Oracle은 incubate-project이다.

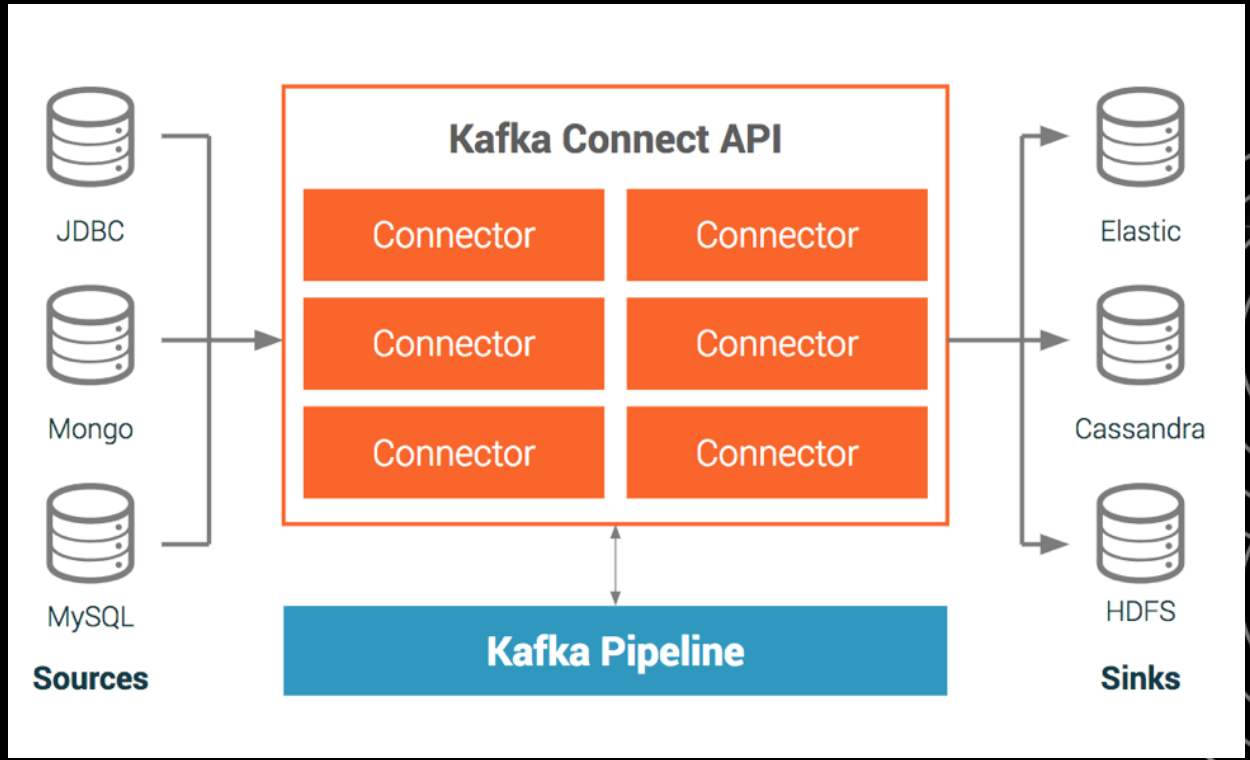
Debezium-카프카 커넥트(Kafka Connect)

- Debezium은 카프카 커넥트(Kafka Connect) 기반의 플러그인
- 카프카 커넥트는 카프카 관련 생태계 오픈 소스 중 하나
- 카프카 커넥트는 아파치 라이선스 2.0

카프카 커넥터(Kafka Connector)

- 카프카 커넥트는 카프카 클러스터를 통해 데이터 베이스, 하둡 HDFS, 검색 같은 외부 시스템 및 파일 시스템에 연결하고 데이터를 가져오는/내보내는 프레임워크를 제공한다.
- 데이터 베이스, 키-값 저장소, 검색 엔진, 파일 시스템에 대한 연결 인터페이스를 커넥터(Connector)라 부른다.

카프카 커넥터(Kafka Connector)



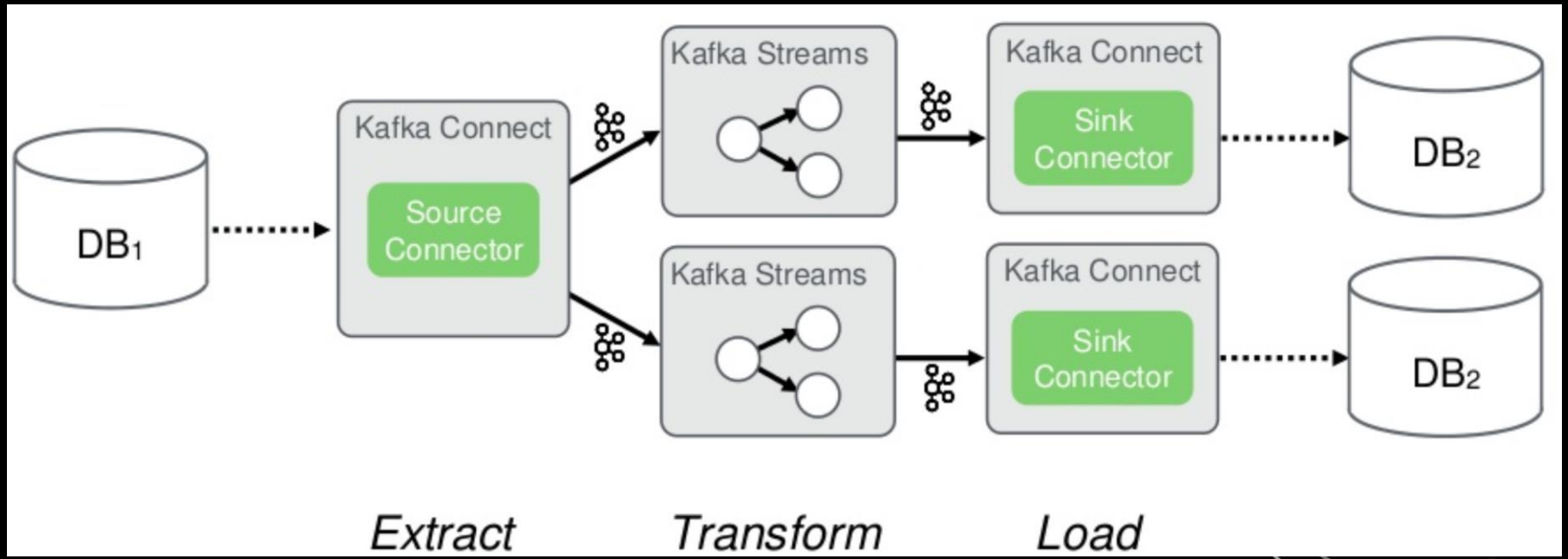
아파치 카프카(Apache Kafka)

- 아파치 카프카는 스트리밍 데이터를 처리한다
- 데이터 스트림을 처리 할 수있는 방대한 생태계를 갖는다
- 신뢰성과 확장성을 갖는다
- 데이터베이스에 데이터를 전달할 수 있는 자 "메시지 버스(message bus)"

카프카 커넥터(Kafka Connector)

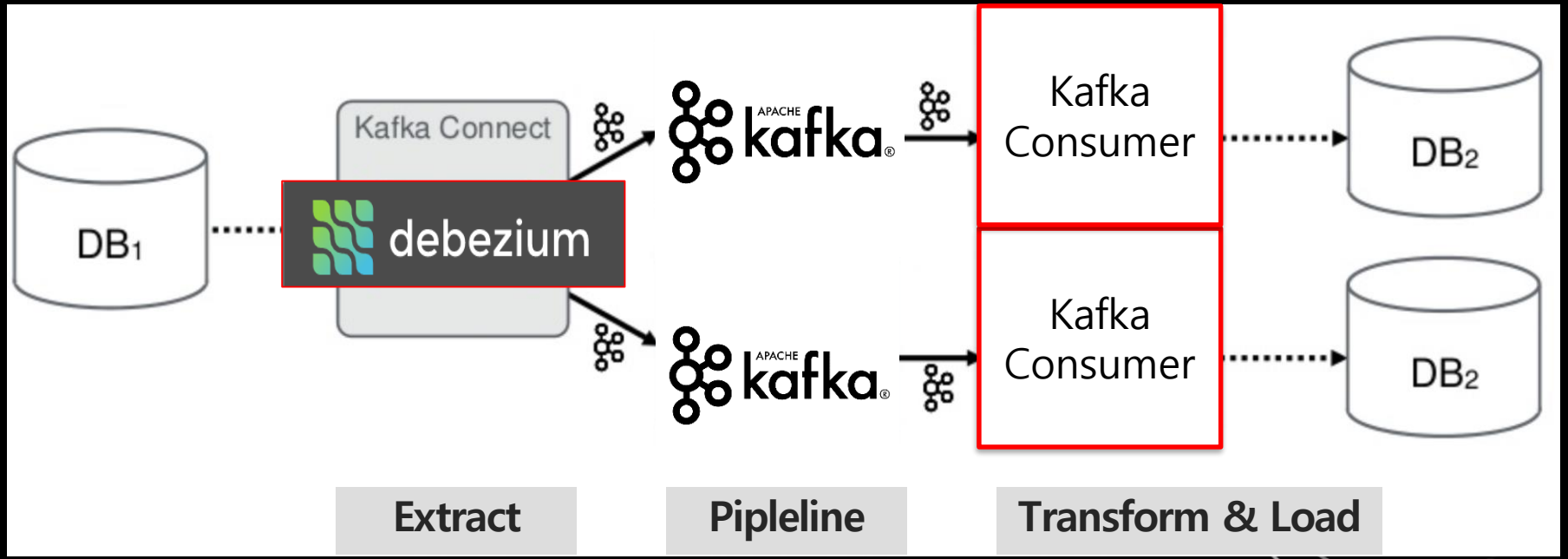
카프카 커넥트 아키텍처

Extract – Transform – Load 패턴

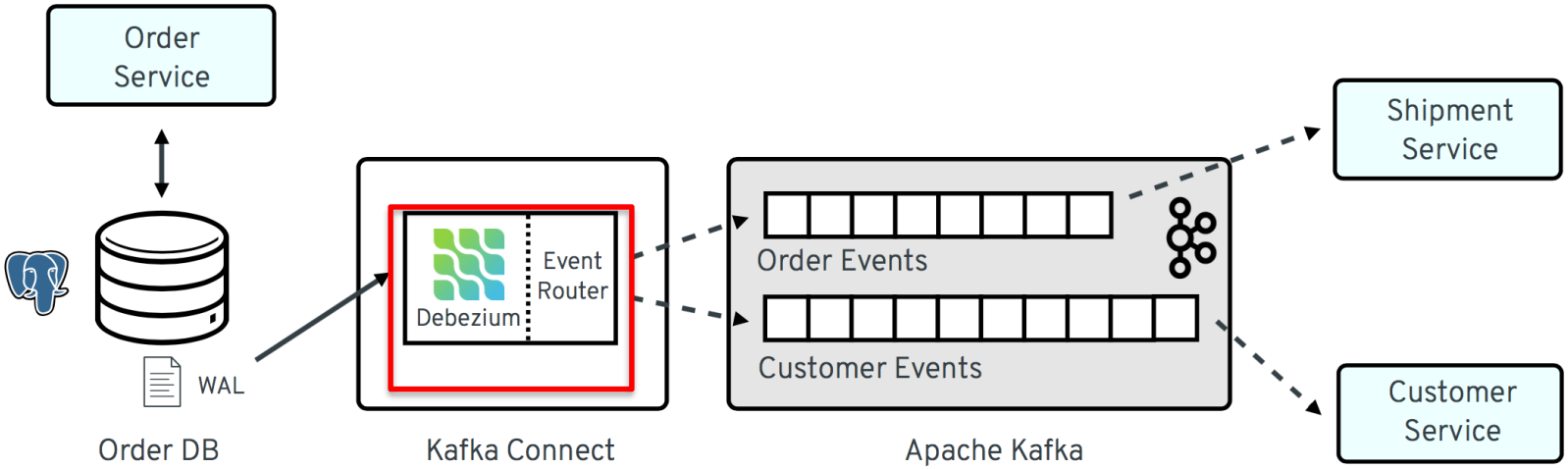


Debezium-카프카 커넥트 아키텍처

Debezium, 아키텍처



Debezium-카프카 커넥트 아키텍처



Id	AggregateType	AggregateId	Type	Payload
ec6e	Order	123	OrderCreated	{ "id" : 123, ... }
8af8	Order	456	OrderDetailCanceled	{ "id" : 456, ... }
890b	Customer	789	InvoiceCreated	{ "id" : 789, ... }

Outbox Table

Debezium 아키텍처 Demo

- Debezium 0.10.0.Beta4
- Kafka Connect 2.3.0
- Mysql 5.7
- Zookeeper

Debezium 아키텍처 Demo


```
// 주키퍼(zookeeper)
$ docker run -it --rm --name zookeeper -p 2181:2181 -p 2888:2888 -p 3888:3888 debezium/zookeeper:0.10

// 카프카
$ docker run -it --rm --name kafka -p 9092:9092 --link zookeeper:zookeeper debezium/kafka:0.10

// mysql
$ docker run -it --rm --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=debezium -e MYSQL_USER=mysqluser -e
MYSQL_PASSWORD=mysqlpw debezium/example-mysql:0.10

// 카프카 커넥트(Debezium)
$ docker run -it --rm --name connect -p 8083:8083 -e GROUP_ID=1 -e CONFIG_STORAGE_TOPIC=my_connect_configs -e
OFFSET_STORAGE_TOPIC=my_connect_offsets -e STATUS_STORAGE_TOPIC=my_connect_statuses --link zookeeper:zookeeper --link
kafka:kafka --link mysql:mysql debezium/connect:0.10
```

Debezium 아키텍처 Demo


```
[~] docker ps -a
CONTAINER ID        IMAGE                           COMMAND
24eaeb468b8d        debezium/connect:0.10           "/docker-entrypoint.…"
954c15adbfb8        debezium/example-mysql:0.10     "docker-entrypoint.s…"
e5bf4e134a07        debezium/kafka:0.10             "/docker-entrypoint.…"
81d2180556ea        debezium/zookeeper:0.10         "/docker-entrypoint.…"
```

Debezium 아키텍처 Demo


```
$ curl -H "Accept:application/json" localhost:8083/
{"version":"2.3.0","commit":"cb8625948210849f"}

//카프카 커넥터 조회
$ curl -H "Accept:application/json" localhost:8083/connectors/
[]
```

Debezium 아키텍처 Demo


```
//카프카 커넥트에 Debezium 설정 추가하기
$ curl -i -X POST -H "Accept:application/json" -H "Content-Type:application/json" localhost:8083/connectors/ -d '{
  "name": "inventory-connector",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "tasks.max": "1",
    "database.hostname": "mysql",
    "database.port": "3306",
    "database.user": "debezium",
    "database.password": "dbz",
    "database.server.id": "184054",
    "database.server.name": "dbserver1",
    "database.whitelist": "inventory",
    "database.history.kafka.bootstrap.servers": "kafka:9092",
    "database.history.kafka.topic": "schema-changes.inventory"
  }
}'
```

Debezium 아키텍처 Demo


```
$ curl -H "Accept:application/json" localhost:8083/connectors/
["inventory-connector"]
```

Debezium 아키텍처 Demo


```
//카프카 커넥터 조회
$ curl -H "Accept:application/json" localhost:8083/connectors/invento
{
 "name": "inventory-connector",
 "config": {
  "connector.class": "io.debezium.connector.mysql.MySqlConnector",
  "tasks.max": "1",
  "database.hostname": "mysql",
  "database.port": "3306",
  "database.user": "debezium",
  "database.password": "dbz",
  "database.server.id": "184054",
  "database.server.name": "dbserver1",
  "database.whitelist": "inventory",
  "database.history.kafka.bootstrap.servers": "kafka:9092",
  "database.history.kafka.topic": "schema-changes.inventory"
 }
}
```

tasks.max : 정확히 하나의 작업이 한 번에 작동해야 한다. MySQL 서버의 binlog를 읽고 올바른 순서를 보장하기 위한 유일한 방법database.hostname: 데이터베이스 호스트는 mysql로 지정되는데, 이는 MySQL 서버를 실행하는 Docker 컨테이너 이름

database.port: MySQL 서버의 포트 지정

database.user , database.password : 사용자 이름과 패스워드

database.server.id :.고유 한 서버 ID와 이름을 제공

database.server.name: MySQL 서버 또는 서버 클러스터의 논리적 식별자이며 모든 카프카 토픽의 접두사로 사용된다.

database.whitelist: inventory 데이터베이스의 변경 사항만 감지

database.history.kafka.bootstrap.servers : Debezium에서는 명명 된 브로커 (이벤트를 보내는 동일한 브로커)와 토픽 이름을 사용해 데이터베이스 스키마 레코드를 카프카에 저장한다

database.history.kafka.topic: 카프카 커넥트가 다시 시작되면 카프카커넥터가 읽기를 시작하기 위해 binlog의 특정 시점에 존재했던 데이터베이스의 스키마를 정의된 카프카 토픽을 통해 복구한다.

Debezium 아키텍처 Demo


```
// 카프카 커넥트 로그
2019-09-28 13:26:02,596 INFO   ||  Successfully tested connection for
jdbc:mysql://mysql:3306/?useInformationSchema=true&nullCatalogMeansCurrent=false&useSSL=false&useUnicode=true&characterEncoding=UTF-8&characterSetResults=UTF-
8&zeroDateTimeBehavior=CONVERT_TO_NULL with user 'debezium'   [io.debezium.connector.mysql.MySqlConnector]
2019-09-28 13:26:02,618 INFO   ||  AbstractConfig values:   [org.apache.kafka.common.config.AbstractConfig]
2019-09-28 13:26:02,691 INFO   ||  [Worker clientId=connect-1, groupId=1] Connector inventory-connector config updated
[org.apache.kafka.connect.runtime.distributed.DistributedHerder]
2019-09-28 13:26:03,198 INFO   ||  [Worker clientId=connect-1, groupId=1] Rebalance started   [org.apache.kafka.connect.runtime.distributed.WorkerCoordinator]
2019-09-28 13:26:03,198 INFO   ||  [Worker clientId=connect-1, groupId=1] (Re-)joining group   [org.apache.kafka.clients.consumer.internals.AbstractCoordinator]
2019-09-28 13:26:03,222 INFO   ||  [Worker clientId=connect-1, groupId=1] Successfully joined group with generation 2
[org.apache.kafka.clients.consumer.internals.AbstractCoordinator]
2019-09-28 13:26:03,223 INFO   ||  [Worker clientId=connect-1, groupId=1] Joined group at generation 2 and got assignment: Assignment{error=0, leader='connect-1-
49541565-2b8a-46df-95d3-1964805561bf', leaderUrl='http://172.17.0.6:8083/', offset=1, connectorIds=[inventory-connector], taskIds=[], revokedConnectorIds=[],
revokedTaskIds=[], delay=0}   [org.apache.kafka.connect.runtime.distributed.DistributedHerder]
2019-09-28 13:26:03,223 INFO   ||  [Worker clientId=connect-1, groupId=1] Starting connectors and tasks using config offset 1
[org.apache.kafka.connect.runtime.distributed.DistributedHerder]
2019-09-28 13:26:03,225 INFO   ||  [Worker clientId=connect-1, groupId=1] Starting connector inventory-connector
[org.apache.kafka.connect.runtime.distributed.DistributedHerder]
```

Debezium 아키텍처 Demo


```
// 카프카 커넥트 로그
2019-09-28 13:26:05,230 INrh ⊏ FO  MySQL|dbserver1|task  Database history topic '(name=schema-changes.inventory, numPartitions=1, replicationFactor=1,
replicasAssignments=null, configs={cleanup.policy=delete, retention.ms=9223372036854775807, retention.bytes=-1})' created
[io.debezium.relational.history.KafkaDatabaseHistory]
2019-09-28 13:26:05,268 INFO  MySQL|dbserver1|task  Found no existing offset, so preparing to perform a snapshot  [io.debezium.connector.mysql.MySqlConnectorTask]
2019-09-28 13:26:05,269 INFO  MySQL|dbserver1|task  [Producer clientId=inventory-connector-dbhistory] Cluster ID: qSAu9VWeR2qdsnexs12z-Q
[org.apache.kafka.clients.Metadata]
2019-09-28 13:26:05,317 INFO  MySQL|dbserver1|task  Requested thread factory for connector MySqlConnector, id = dbserver1 named = binlog-client
[io.debezium.util.Threads]
2019-09-28 13:26:05,330 INFO  MySQL|dbserver1|task  Requested thread factory for connector MySqlConnector, id = dbserver1 named = snapshot  [io.debezium.util.Threads]

2019-09-28 13:26:05,332 INFO  MySQL|dbserver1|task  Creating thread debezium-mysqlconnector-dbserver1-snapshot  [io.debezium.util.Threads]
2019-09-28 13:26:05,333 INFO  ||  WorkerSourceTask{id=inventory-connector-0} Source task finished initialization and start
[org.apache.kafka.connect.runtime.WorkerSourceTask]
2019-09-28 13:26:05,334 INFO  MySQL|dbserver1|snapshot  Starting snapshot for
jdbc:mysql://mysql:3306/?useInformationSchema=true&nullCatalogMeansCurrent=false&useSSL=false&useUnicode=true&characterEncoding=UTF-8&characterSetResults=UTF-
8&zeroDateTimeBehavior=CONVERT_TO_NULL with user 'debezium' with locking mode 'minimal'  [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,339 INFO  MySQL|dbserver1|snapshot  Snapshot is using user 'debezium' with these MySQL grants:  [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,340 INFO  MySQL|dbserver1|snapshot  GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO
'debezium'@'%'  [io.debezium.connector.mysql.SnapshotReader]
```

Debezium 아키텍처 Demo


```
// 카프카 커넥트 로그
2019-09-28 13:26:05,357 INFO   MySQL|dbserver1|snapshot  Step 0: disabling autocommit and enabling repeatable read transactions
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,361 INFO   MySQL|dbserver1|snapshot  Step 1: flush and obtain global read lock to prevent writes to database
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,362 INFO   MySQL|dbserver1|snapshot  Step 2: start transaction with consistent snapshot   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,363 INFO   MySQL|dbserver1|snapshot  Step 3: read binlog position of MySQL master   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,366 INFO   MySQL|dbserver1|snapshot                  using binlog 'mysql-bin.000003' at position '154' and gtid ''
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,367 INFO   MySQL|dbserver1|snapshot  Step 4: read list of available databases   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,368 INFO   MySQL|dbserver1|snapshot                  list of available databases is: [information_schema, inventory, mysql, performance_schema, sys]
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,368 INFO   MySQL|dbserver1|snapshot  Step 5: read list of available tables in each database   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:05,426 INFO   MySQL|dbserver1|snapshot  Step 6: generating DROP and CREATE statements to reflect current database schemas:
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:06,513 INFO   MySQL|dbserver1|snapshot                  SET character_set_server=latin1, collation_server=latin1_swedish_ci;SET character_set_server=latin1,
collation_server=latin1_swedish_ci;   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:06,579 INFO   MySQL|dbserver1|snapshot                  DROP TABLE IF EXISTS addresses;   [io.debezium.connector.mysql.SnapshotReader]
..
2019-09-28 13:26:08,044 INFO   MySQL|dbserver1|snapshot                  DROP DATABASE IF EXISTS `inventory`   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,048 INFO   MySQL|dbserver1|snapshot                  CREATE DATABASE `inventory`   [io.debezium.connector.mysql.SnapshotReader]
```

Debezium 아키텍처 Demo


```
// 카프카 커넥트 로그
…
2019-09-28 13:26:08,081 INFO   MySQL|dbserver1|snapshot                    CREATE TABLE `customers` (  `id` int(11) NOT NULL AUTO_INCREMENT,  `first_name` varchar(255)
NOT NULL,  `last_name` varchar(255) NOT NULL,  `email` varchar(255) NOT NULL,  PRIMARY KEY (`id`),  UNIQUE KEY `email` (`email`)
..
2019-09-28 13:26:08,112 INFO   MySQL|dbserver1|snapshot  Step 7: releasing global read lock to enable MySQL writes   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,116 INFO   MySQL|dbserver1|snapshot  Step 7: blocked writes to MySQL for a total of 00:00:02.752   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,118 INFO   MySQL|dbserver1|snapshot  Step 8: scanning contents of 6 tables while still in transaction   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,122 INFO   MySQL|dbserver1|snapshot  Step 8: - scanning table 'inventory.addresses' (1 of 6 tables)   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,122 INFO   MySQL|dbserver1|snapshot  For table 'inventory.addresses' using select statement: 'SELECT * FROM `inventory`.`addresses`'
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,130 INFO   MySQL|dbserver1|snapshot  Step 8: - Completed scanning a total of 7 rows from table 'inventory.addresses' after 00:00:00.008
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,133 INFO   MySQL|dbserver1|snapshot  Step 8: - scanning table 'inventory.customers' (2 of 6 tables)   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,133 INFO   MySQL|dbserver1|snapshot  For table 'inventory.customers' using select statement: 'SELECT * FROM `inventory`.`customers`'
[io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,135 INFO   MySQL|dbserver1|snapshot  Step 8: - Completed scanning a total of 4 rows from table 'inventory.customers' after 00:00:00.002
[io.debezium.connector.mysql.SnapshotReader]
```

Debezium 아키텍처 Demo


```
// 카프카 커넥트 로그
…
2019-09-28 13:26:08,160 INFO   MySQL|dbserver1|snapshot  Step 9: committing transaction   [io.debezium.connector.mysql.SnapshotReader]
2019-09-28 13:26:08,161 INFO   MySQL|dbserver1|snapshot  Completed snapshot in 00:00:02.828   [io.debezium.connector.mysql.SnapshotReader]

// 스냅샷 완료
```

Debezium 아키텍처 Demo


```
// 생성된 카프카 토픽


dbserver1
schema-changes
dbserver1.inventory.products
dbserver1.inventory.products_on_hand
dbserver1.inventory.customers
dbserver1.inventory.orders

// 2개의 논리 스키마 정보(DDL) + 테이블 정보(데이터)
```

Debezium 아키텍처 Demo

// 생성된 카프카 토픽 : dbserver1.inventory.customers

$ docker run -it --name watcher --rm --link zookeeper:zookeeper --link kafka:kafka debezium/kafka:0.10 watch-topic -a -k dbserver1.inventory.customers

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1001}} {"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"before"},{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"field":"version"},{"type":"string","optional":false,"field":"connector"},{"type":"string","optional":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":{"allowed":"true,last,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"string","optional":true,"field":"table"},{"type":"int64","optional":false,"field":"server_id"},{"type":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optional":false,"field":"pos"},{"type":"int32","optional":false,"field":"row"},{"type":"int64","optional":true,"field":"thread"},{"type":"string","optional":true,"field":"query"}],"optional":false,"name":"io.debezium.connector.mysql.Source","field":"source"},{"type":"string","optional":false,"field":"op"},{"type":"int64","optional":true,"field":"ts_ms"}],"optional":false,"name":"dbserver1.inventory.customers.Envelope"},"payload":{"before":null,"after":{"id":1001,"first_name":"Sally","last_name":"Thomas","email":"sally.thomas@acme.com"},"source":{"version":"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":0,"snapshot":"true","db":"inventory","table":"customers","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"op":"c","ts_ms":1569907292059}}

Debezium 아키텍처 Demo

// 생성된 카프카 토픽 : dbserver1.inventory.customers

$ docker run -it --name watcher --rm --link zookeeper:zookeeper --link kafka:kafka debezium/kafka:0.10 watch-topic -a -k dbserver1.inventory.customers

Key

Value


```
"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1001}}	{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"before"},{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"field":"version"},{"type":"string","optional":false,"field":"connector"},{"type":"string","optional":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":{"allowed":"true,last,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"string","optional":true,"field":"table"},{"type":"int64","optional":false,"field":"server_id"},{"type":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optional":false,"field":"pos"},{"type":"int32","optional":false,"field":"row"},{"type":"int64","optional":true,"field":"thread"},{"type":"string","optional":true,"field":"query"}],"optional":false,"name":"io.debezium.connector.mysql.Source","field":"source"},{"type":"string","optional":false,"field":"op"},{"type":"int64","optional":true,"field":"ts_ms"}],"optional":false,"name":"dbserver1.inventory.customers.Envelope"},"payload":{"before":null,"after":{"id":1001,"first_name":"Sally","last_name":"Thomas","email":"sally.thomas@acme.com"},"source":{"version":"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":0,"snapshot":"true","db":"inventory","table":"customers","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"op":"c","ts_ms":1569907292059}}
```

Debezium 아키텍처 Demo

// dbserver1 토픽에 논리 스키마(DDL 정보)가 저장됨

{"schema":{"type":"struct","fields":.... "gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"databaseName":"","ddl":"SET character_set_server=latin1, collation_server=latin1_swedish_ci;SET character_set_server=latin1, collation_server=latin1_swedish_ci;"}}

..

{"schema":{"type":"struct","fields":.... "gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"databaseName":"inventory","ddl":"DROP TABLE IF EXISTS customers;"}}

{"schema":{"type":"struct","fields":.... "gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"databaseName":"inventory","ddl":"CREATE DATABASE `inventory`"}}

{"schema":{"type":"struct","fields":.... "gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"databaseName":"inventory","ddl":"USE `inventory`"}}

{"schema":{"type":"struct","fields":.... "gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"databaseName":"inventory","ddl":"CREATE TABLE `customers` (₩n `id` int(11) NOT NULL AUTO_INCREMENT,₩n `first_name` varchar(255) NOT NULL,₩n `last_name` varchar(255) NOT NULL,₩n `email` varchar(255) NOT NULL,₩n PRIMARY KEY (`id`),₩n UNIQUE KEY `email` (`email`)₩n) ENGINE=InnoDB AUTO_INCREMENT=1005 DEFAULT CHARSET=latin1"}}

...

Debezium 아키텍처 Demo

SOSCON2019

// mysql db의 customers 테이블

```
mysql> select * from customers;
+------+------------+-----------+-----------------------+
| id   | first_name | last_name | email                 |
+------+------------+-----------+-----------------------+
| 1001 | Sally      | Thomas    | sally.thomas@acme.com |
| 1002 | George     | Bailey    | gbailey@foobar.com    |
| 1003 | Edward     | Walker    | ed@walker.com         |
| 1004 | Anne       | Kretchmar | annek@noanswer.org    |
+------+------------+-----------+-----------------------+
4 rows in set (0.00 sec)
```

// 카프카 토픽 (dbserver1.inventory.customers) 확인하기

Debezium 아키텍처 Demo

SOSCON2019

// 카프카 토픽 (dbserver1.inventory.customers) 확인하기

```
[~] docker run -it --name watcher --rm --link zookeeper:zookeeper --link kafka:kafka debezium/kafka:0.10 watch-topic -a -k dbserver1.inventory.customers
WARNING: Using default BROKER_ID=1, which is valid only for non-clustered installations.
Using ZOOKEEPER_CONNECT=172.17.0.2:2181
Using KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://172.17.0.7:9092
Using KAFKA_BROKER=172.17.0.3:9092
Contents of topic dbserver1.inventory.customers:
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1001}}  {"schema":{"type":"struct
,"fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{
"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"before"},{"type":"struct","fields":[{"type":"int32","optional":false,"field
:"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"db
erver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"field":"version"},{"type":"string","optional":false,"field":"connector"},{"type":"st
ring","optional":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":{"allowed":"true,lo
st,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"string","optional":true,"field":"table"},{"type":"int64","optional":false,"field":"server_i
d"},{"type":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optional":false,"field":"pos"},{"type":"int32","optional":false,"field":"row
"},{"type":"int64","optional":true,"field":"thread"},{"type":"string","optional":true,"field":"query"}],"optional":false,"name":"io.debezium.connector.mysql.Source","field":"source"},{"type":"string
","optional":false,"field":"op"},{"type":"int64","optional":true,"field":"ts_ms"}],"optional":false,"name":"dbserver1.inventory.customers.Envelope"},"payload":{"before":null,"after":{"id":1001,"fir
st_name":"Sally","last_name":"Thomas","email":"sally.thomas@acme.com"},"source":{"version":"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":0,"snapshot":"true","db":"inventory","table":"
customers","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"op":"c","ts_ms":1569907292059}}
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1002}}  {"schema":{"type":"struct
,"fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{
"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"before"},{"type":"struct","fields":[{"type":"int32","optional":false,"field
:"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"db
erver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"field":"version"},{"type":"string","optional":false,"field":"connector"},{"type":"st
ring","optional":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":{"allowed":"true,lo
st,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"string","optional":true,"field":"table"},{"type":"int64","optional":false,"field":"server_i
d"},{"type":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optional":false,"field":"pos"},{"type":"int32","optional":false,"field":"row
"},{"type":"int64","optional":true,"field":"thread"},{"type":"string","optional":true,"field":"query"}],"optional":false,"name":"io.debezium.connector.mysql.Source","field":"source"},{"type":"string
","optional":false,"field":"op"},{"type":"int64","optional":true,"field":"ts_ms"}],"optional":false,"name":"dbserver1.inventory.customers.Envelope"},"payload":{"before":null,"after":{"id":1002,"fir
st_name":"George","last_name":"Bailey","email":"gbailey@foobar.com"},"source":{"version":"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":0,"snapshot":"true","db":"inventory","table":"cu
stomers","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"op":"c","ts_ms":1
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inven
,"fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field
"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"befo
:"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type"
erver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"field":"vers
ring","optional":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"na
st,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"string","optional":
d"},{"type":"string","optional":true,"field":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optio
},{"type":"int64","optional":true,"field":"thread"},{"type":"string","optional":true,"field":"query"}],"optional":false,"name
","optional":false,"field":"op"},{"type":"int64","optional":true,"field":"ts_ms"}],"optional":false,"name":"dbserver1.invento
st_name":"George","last_name":"Bailey","email":"gbailey@foobar.com"},"source":{"version":"0.10.0.CR2","connector":"mysql","na
stomers","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"op":"c","ts_ms":1
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inven
,"fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field
"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"befo
:"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_name"},{"type"
erver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"field":"vers
ring","optional":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"na
st,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"string","optional":
```

```
// mysql db의 customers 테이블
mysql> select * from customers;
+------+------------+-----------+-----------------------+
| id   | first_name | last_name | email                 |
+------+------------+-----------+-----------------------+
| 1001 | Sally      | Thomas    | sally.thomas@acme.com |
| 1002 | George     | Bailey    | gbailey@foobar.com    |
| 1003 | Edward     | Walker    | ed@walker.com         |
| 1004 | Anne       | Kretchmar | annek@noanswer.org    |
+------+------------+-----------+-----------------------+
4 rows in set (0.00 sec)
```

op:커맨드(c: 생성, u:변경, d:삭제)

before: op커맨드 이전 값

after: op 커맨드 이후 값

file: bin로그 파일

pos: bin 로그 파일 위치

ts_ms: Debezium이 이벤트를 처리한 시간

…"payload":{"before":null,"after":{"id":1004,"first_name":"Anne","last_name":"Kretchmar","email":"annek@noanswer.org"},"source":{"version":"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":0,"snapshot":"true","db":"inventory","table":"customers","server_id":0,"gtid":null,"file":"mysql-bin.000003","pos":154,"row":0,"thread":null,"query":null},"op":"c","ts_ms":1569907292059}}

Debezium 아키텍처 Demo


```
mysql> DELETE FROM customers WHERE id=1004;Query OK, 1 row affected (0.00 sec)
```

// 카프카 토픽 (dbserver1.inventory.customers) 확인하기


```
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory
.customers.Key"},"payload":{"id":1004}} {"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional
":false,"field":"id"},{"type":"string","optional":false,"field":"first_name"},{"type":"string","optional":false,"field":"last_nam
e"},{"type":"string","optional":false,"field":"email"}],"optional":true,"name":"dbserver1.inventory.customers.Value","field":"bef
ore"},{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"},{"type":"string","optional":false,"field":"first_
name"},{"type":"string","optional":false,"field":"last_name"},{"type":"string","optional":false,"field":"email"}],"optional":true
,"name":"dbserver1.inventory.customers.Value","field":"after"},{"type":"struct","fields":[{"type":"string","optional":false,"fiel
d":"version"},{"type":"string","optional":false,"field":"connector"},{"type":"string","optional":false,"field":"name"},{"type":"i
nt64","optional":false,"field":"ts_ms"},{"type":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters":
{"allowed":"true,last,false"},"default":"false","field":"snapshot"},{"type":"string","optional":false,"field":"db"},{"type":"stri
ng","optional":true,"field":"table"},{"type":"int64","optional":false,"field":"server_id"},{"type":"string","optional":true,"fiel
d":"gtid"},{"type":"string","optional":false,"field":"file"},{"type":"int64","optional":false,"field":"pos"},{"type":"int32","opt
ional":false,"field":"row"},{"type":"int64","optional":true,"field":"thread"},{"type":"string","optional":true,"field":"query"}],
"optional":false,"name":"io.debezium.connector.mysql.Source","field":"source"},{"type":"string","optional":false,"field":"op"},{"
type":"int64","optional":true,"field":"ts_ms"}],"optional":false,"name":"dbserver1.inventory.customers.Envelope"},"payload":{"bef
ore":{"id":1004,"first_name":"Anne Marie","last_name":"Kretchmar","email":"annek@noanswer.org"},"after":null,"source":{"version":
"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":1569936375000,"snapshot":"false","db":"inventory","table":"customers"
,"server_id":223344,"gtid":null,"file":"mysql-bin.000003","pos":1066,"row":0,"thread":7,"query":null},"op":"d","ts_ms":1569936375
513}}
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory
.customers.Key"},"payload":{"id":1004}} null
```


// 2개의 이벤트를 받음

Debezium 아키텍처 Demo


```
mysql> DELETE FROM customers WHERE id=1004;Query OK, 1 row affected (0.00 sec)
```

op:커맨드(c: 생성, u:변경, d:삭제)

file: bin로그 파일

pos: bin 로그 파일 위치

ts_ms: Debezium이 이벤트를 처리한 시간

// 카프카 토픽 (dbserver1.inventory.customers) 확인하기

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory
customers.Key"},"payload":{"id":1004}} {"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional

// 2개의 이벤트를 받음

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,
"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1004}}
…."gtid":null,"file":"mysql-
bin.000003","pos":1066,"row":0,"thread":7,"query":null},"op":"d","ts_ms":1569936375513}}

{{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false
,"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1004}}

null

ore":{"id":1004,"first_name":"Anne Marie","last_name":"Kretchmar","email":"annek@noanswer.org"},"after":null,"source":{"version":
"0.10.0.CR2","connector":"mysql","name":"dbserver1","ts_ms":1569936375000,"snapshot":"false","db":"inventory","table":"customers"
,"server_id":223344,"gtid":null,"file":"mysql-bin.000003","pos":1066,"row":0,"thread":7,"query":null},"op":"d","ts_ms":1569936375
513}}
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,"name":"dbserver1.inventory
.customers.Key"},"payload":{"id":1004}} null

Debezium 아키텍처 Demo

mysql> DELETE FROM customers WHERE id=1004;Query OK, 1 row affected (0.00 sec)

op:커맨드(c: 생성, u:변경, d:삭제)
file: bin로그 파일
pos: bin 로그 파일 위치
ts_ms: Debezium이 이벤트를 처리한 시간

// 카프카 토픽 (dbserver1.inventory.customers) 확인하기

{"schema":{"type":"struct"

첫 번째는 실제 삭제에 대한 내용(before, after)을 출력한다

// 2개의 이벤트를 받

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false,
"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1004}}
…."gtid":null,"file":"mysql-
bin.000003","pos":1066,"row":0,"thread":7,"query":null},"op":"d","ts_ms":1569936375513}}

{{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"id"}],"optional":false
,"name":"dbserver1.inventory.customers.Key"},"payload":{"id":1004}} null

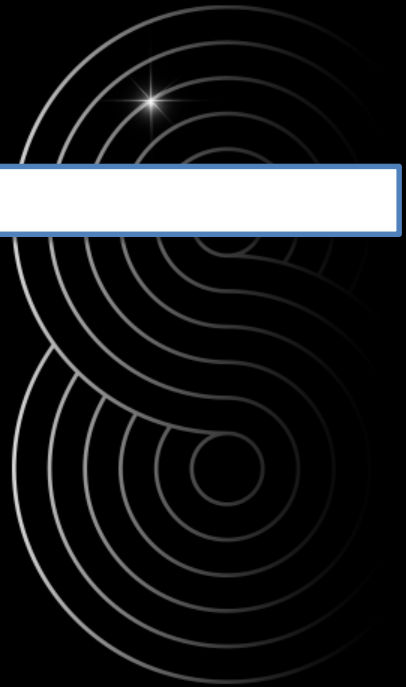
두 번째는 Debezium이 삭제 표시 이벤트를 호출한다

"tombstones.on.delete": "false"을 사용하면 null이 나오지 않는다.

Debezium 아키텍처 Demo

// 종료하기

```
$ docker stop mysqlterm watcher connect mysql kafka zookeeper
```



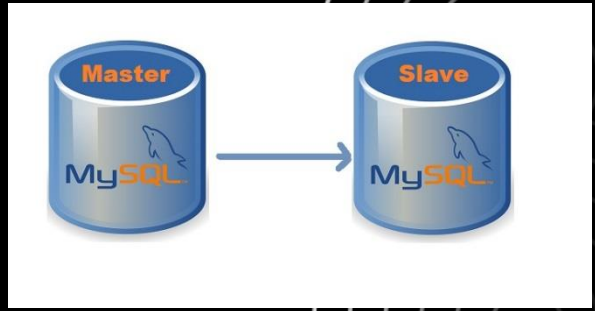
Debezium 실제 환경

- MHA(또는 Master-Slave) 환경 기반이라면 GTID를 설정한다

```
gtid_mode                = on
enforce_gtid_consistency = on
```

- Bin 로그 설정

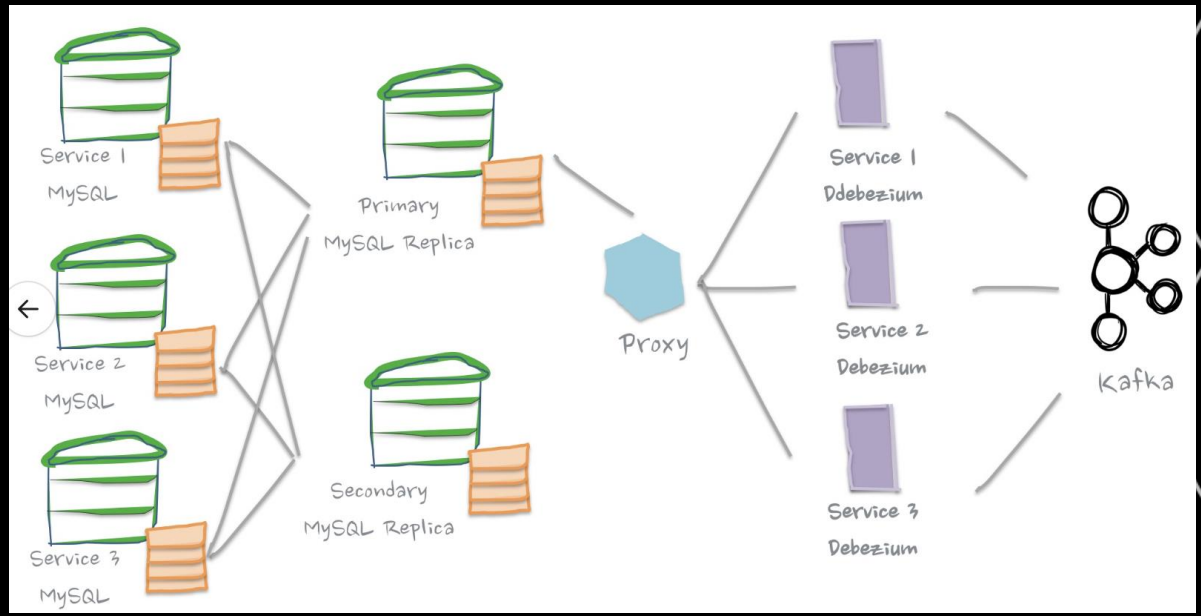
```
server-id          = 223344
log_bin            = mysql-bin
binlog_format      = row
binlog_row_image   = full
expire_logs_days   = 10
```



- Bin 로그를 사용할 Mysql 계정 생성 (Grant 권한 계정)

Debezium 실제 환경

- 대용량 환경에서는 아키텍처를 잘 잡아야 한다 ->
  훌륭한 Mysql DBA, 개발자, Devops가 있어야 한다.



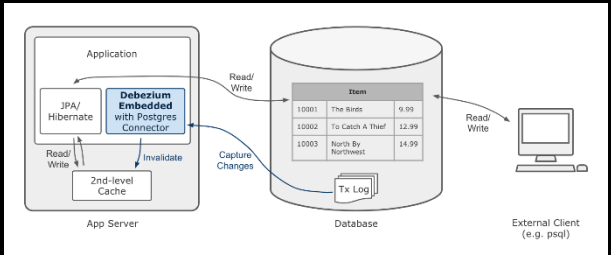
Wepay 모델

Debezium 특징

- 스키마 레지스트리(Schema Registry)와 잘 연동한다
- DML과 DDL이 카프카 토픽에 저장한다.
  Mysql bin log position(offset) 정보 및 Schema 정보를 저장한다.
  따라서 복제 인자(replication factor)가 중요하다. 복제 인자가 1이면 데이터가 깨져서 고생할 수 있다.
- Schema 토픽 데이터가 깨지면 Debezium이 정상적으로 동작하지 않는다.
  debezium 설정에 "snapshot.mode":"schema_only_recovery "로 변경해 테이블 스키마와 데이터를 처음부터 카프카 토픽에 다시 저장하게 한다.

Debezium 특징

- Debezium이 카프카 토픽에 저장하는 데이터의 Mysql GTid, Mysql bin log는 이벤트마다 저장한다.
- Debezium/카프카 커넥트가 중단되었다가 재시작되면 Debezium은 마지막에 저장된 Mysql binlog position(offset) 부터 다시 읽어 카프카 토픽에 저장한다.
- Debezium이 토픽에 저장하는 before, after 레코드 필드는 Mongodb의 Oplog를 연상한다.
- 참고로 Debezium PostgresSQL 버전에는 JPA - Local 캐시 연동이 가능하다.



Debezium 팁

- 순서 보장을 위해(데이터 무결성이 깨어지지 않도록) Debezium 설정의 tasks.max의 값을 1로 설정해야 할 때가 있다. 초당 수만건을 가져가기에 조금 걸리지만 DB, 카프카 커텍트에서는 CPU를 적게 사용한다.
- Debezium 설정 변경시 카프카 커넥트에서 기존 Debezium 설정을 삭제하고 다시 추가한다.
- Mysql 스위치 오버/페일 오버가 발생할 때 커넥션 이슈로 카프카 커넥트 재시작이 필요할 때가 있다.
- Mysql 스위치 오버/ 페일오버에 대한 binlog 손실에 대한 대비를 진행해야 한다. Debezium 만 운영해서 현실의 문제를 해결할 수 없다!! (운영툴이 필요)

Debezium 팁

- decimal.handling.mode의 기본 모드는 precise로서 Bas64인코딩 정보를 kafka에 전달한다. 정밀도 이슈가 발생할 수 있다.
  Debezium 설정에 "decimal.handling.mode": "string " 을 추가해 BigDecimal 이슈를 해결할 수 있다.
- 토픽 이름을 Mysql 스키마 형태가 아니라 다른 이름(아래 예, 테이블 이름)으로 변경을 할 수 있다

```
"transforms": "route",
"transforms.route.type": "org.apache.kafka.connect.transforms.RegexRouter",
"transforms.route.regex": "([^.]+)\\.([^.]+)\\.([^.]+)",
"transforms.route.replacement": "$3"
```

THANK YOU

SOSCON2019

SAMSUNG OPEN SOURCE CONFERENCE 2019