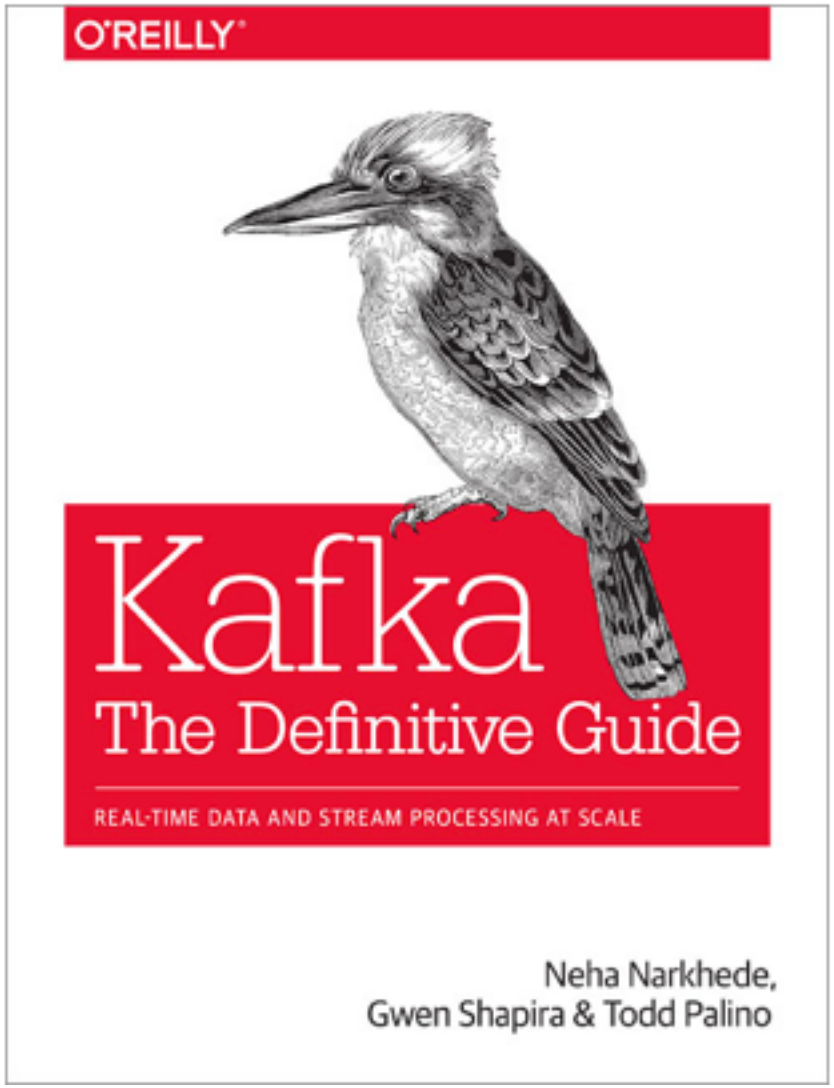
Building a **Real-Time Streaming Platform**

Gwen Shapira, Principal Data Architect

@Gwenshap

About Me

- Working @confluent
- Moving data around for ~20 years
- Was DBA, Consultant, engineer, Product Manager…
- Principal Data Architect
- Wrote a book or two
- @gwenshap
- Github.com/gwenshap



What Is Kafka?

confluent

“

Kafka is a
Distributed
Streaming Platform

confluent

“

What do “distributed streaming platforms” do?

confluent

“

We collect, store, integrate and process Streams of Events

confluent

“

Why?

confluent

This enables easier data integration, and real time processing of events.

Making our business more agile.

confluent

“ I don’t have streams of events.

I just have a bunch of tables.

confluent

Bold claim:
All your data is event streams

EXAMPLE EVENT: PRODUCT VIEW

```
{
    "time": 1447042241376,
    "user_id": 12345,
    "product_id": 5678,
    "page": "product.detail",
    ⋮
}
```

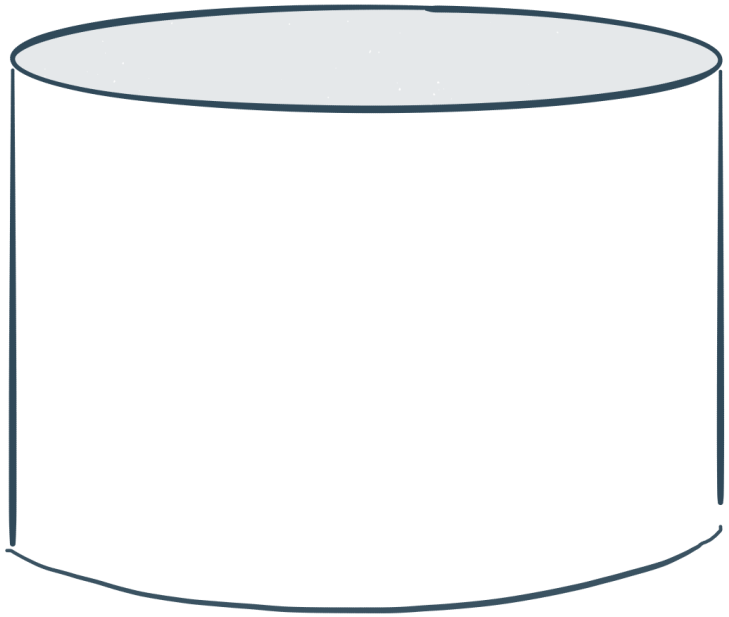
confluent

SENSORS

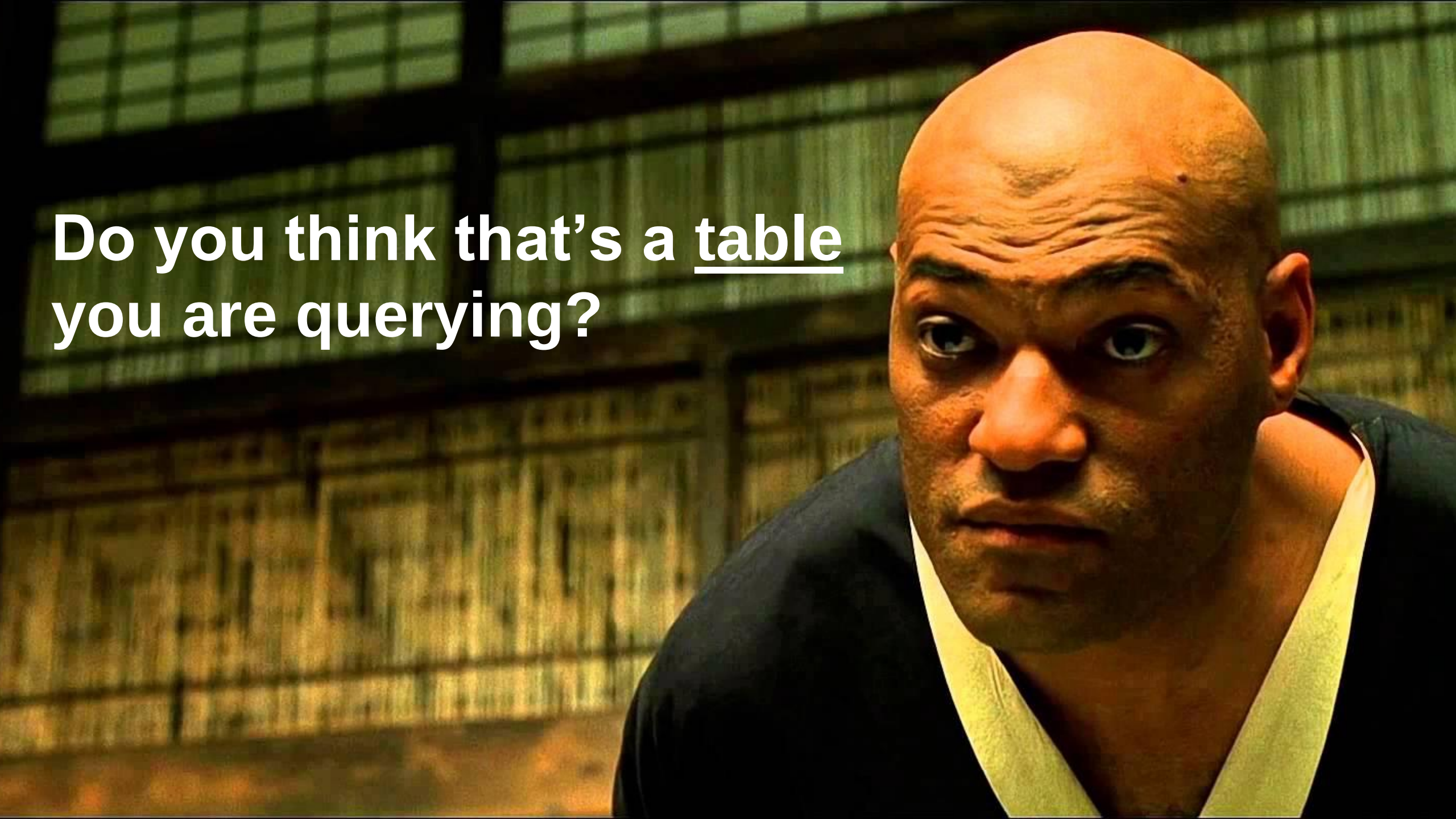
LOG FILES

```
jkreps-mn:~ jkreps$ tail -f -n 20 /var/log/apache2/access_log
::1 - - [23/Mar/2014:15:07:00 -0700] "GET /images/apache_feather.gif HTTP/1.1" 200 4128
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/producer_consumer.png HTTP/1.1" 200 86
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_anatomy.png HTTP/1.1" 200 19579
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/consumer-groups.png HTTP/1.1" 200 268
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_compaction.png HTTP/1.1" 200 41414
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /documentation.html HTTP/1.1" 200 189893
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 200
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/kafka_log.png HTTP/1.1" 200 134321
::1 - - [23/Mar/2014:15:07:04 -0700] "GET /images/mirror-maker.png HTTP/1.1" 200 17054
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /documentation.html HTTP/1.1" 200 189937
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /styles.css HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_logo.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/producer_consumer.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_anatomy.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/consumer-groups.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_cleaner_anatomy.png HTTP/1.1" 304
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/log_compaction.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/kafka_log.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:08:07 -0700] "GET /images/mirror-maker.png HTTP/1.1" 304 -
::1 - - [23/Mar/2014:15:09:55 -0700] "GET /documentation.html HTTP/1.1" 200 195264
```

DATABASES



confluent


Do you think that's a table
you are querying?

The Stream/Table Duality

Time

Stream

Account ID	Amount
12345	+ €50
12345	+ €25
12345	-€60

Table

Account ID	Balance
12345	€50

Account ID	Balance
12345	€75

Account ID	Balance
12345	€15

http://docs.confluent.io/current/streams/concepts.html#duality-of-streams-and-tables

Now we just need to get this stream into Kafka.

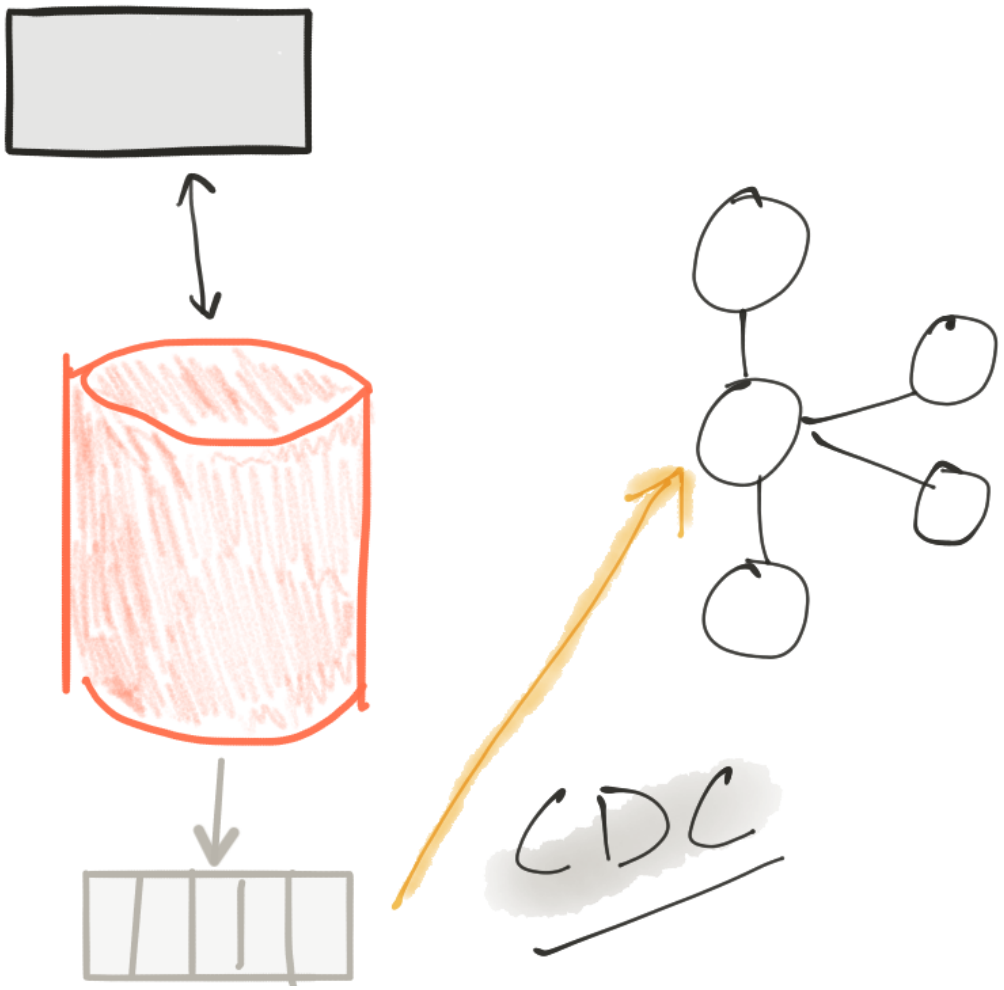
Why?

Change Data Capture is the “Gateway” to Stream Processing

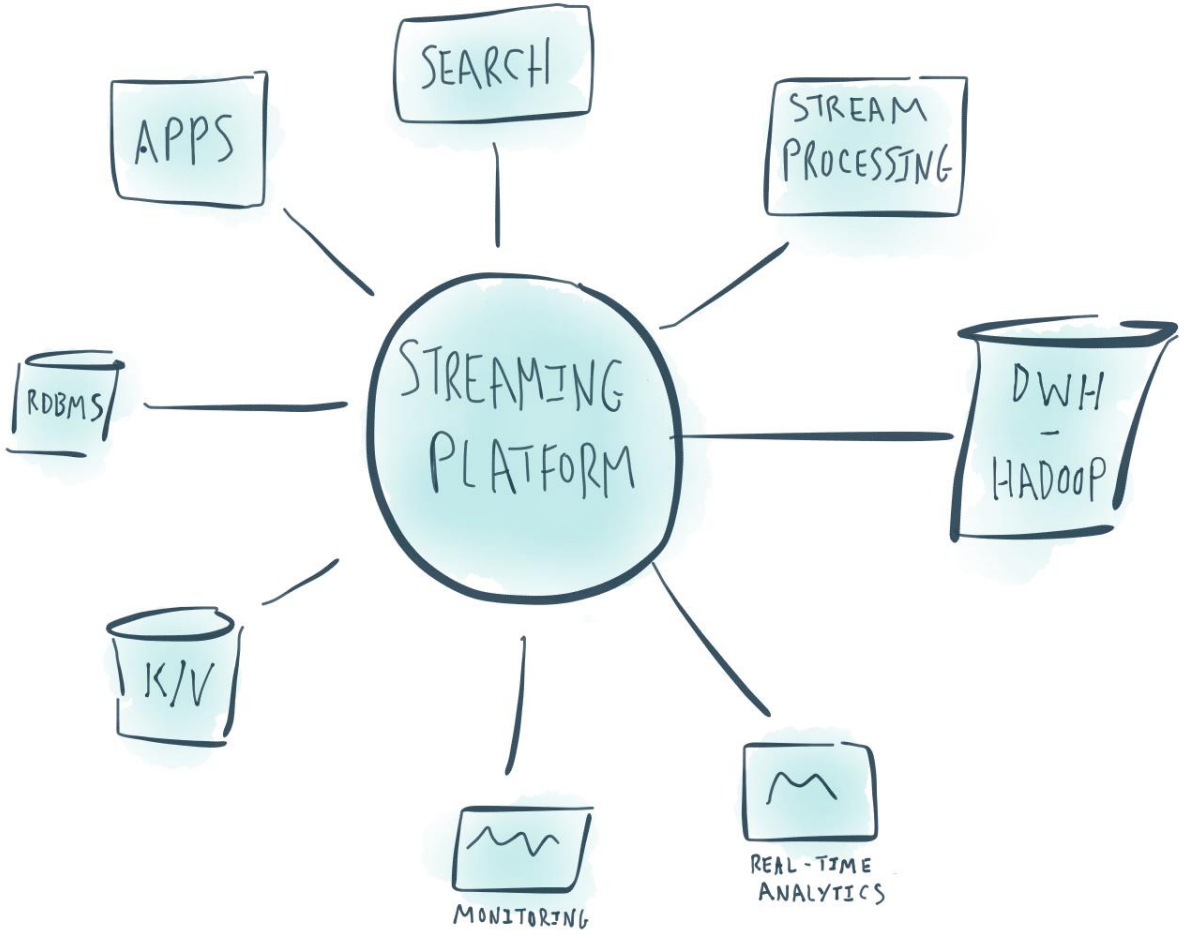
- You can’t “Stream Process” if you don’t have streams of events.
- Changing applications isn’t always easy
- Create streams of events with database change capture

Liberate Application Data into Kafka with CDC

- CDC tools that integrate with Kafka Connect include:
  - GoldenGate
  - Debezium
  - DBVisit
  - Attunity
  - + more

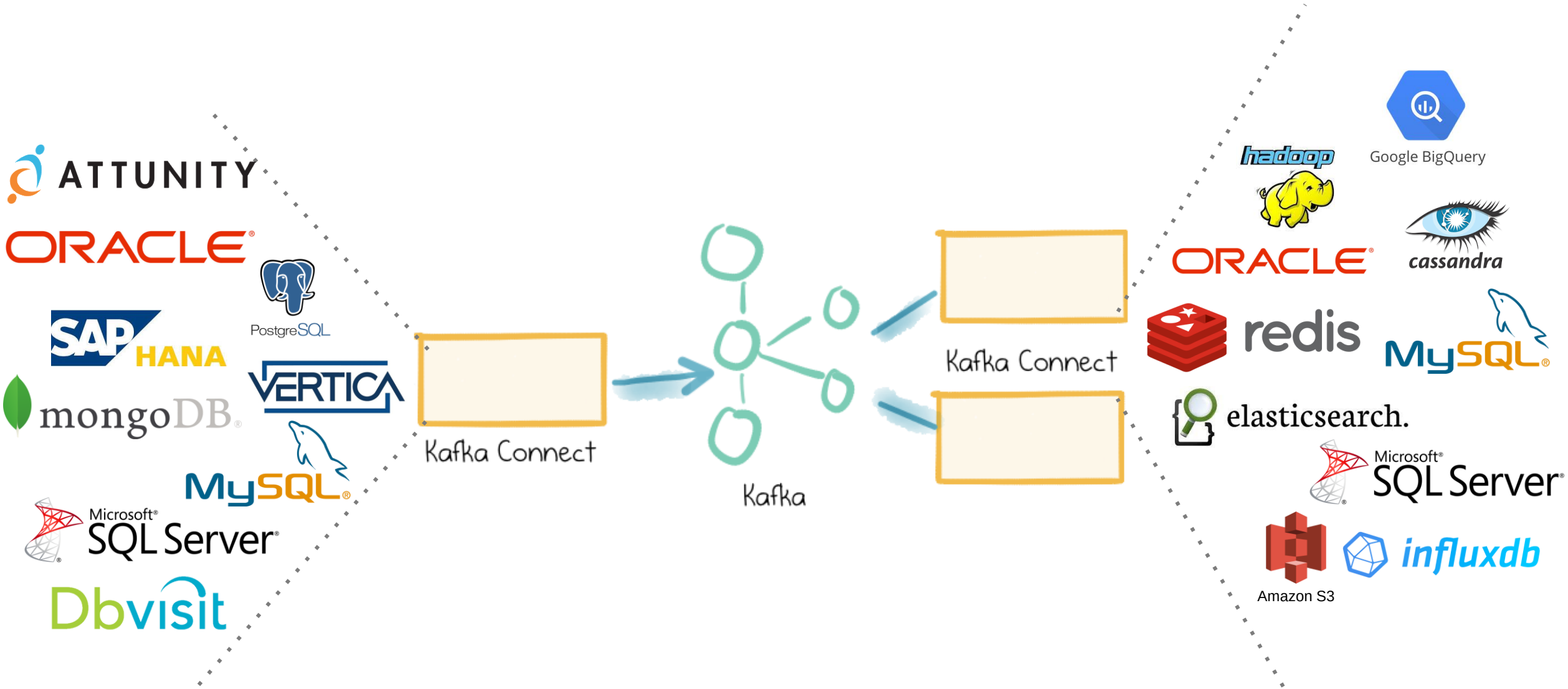


STREAMING PLATFORM

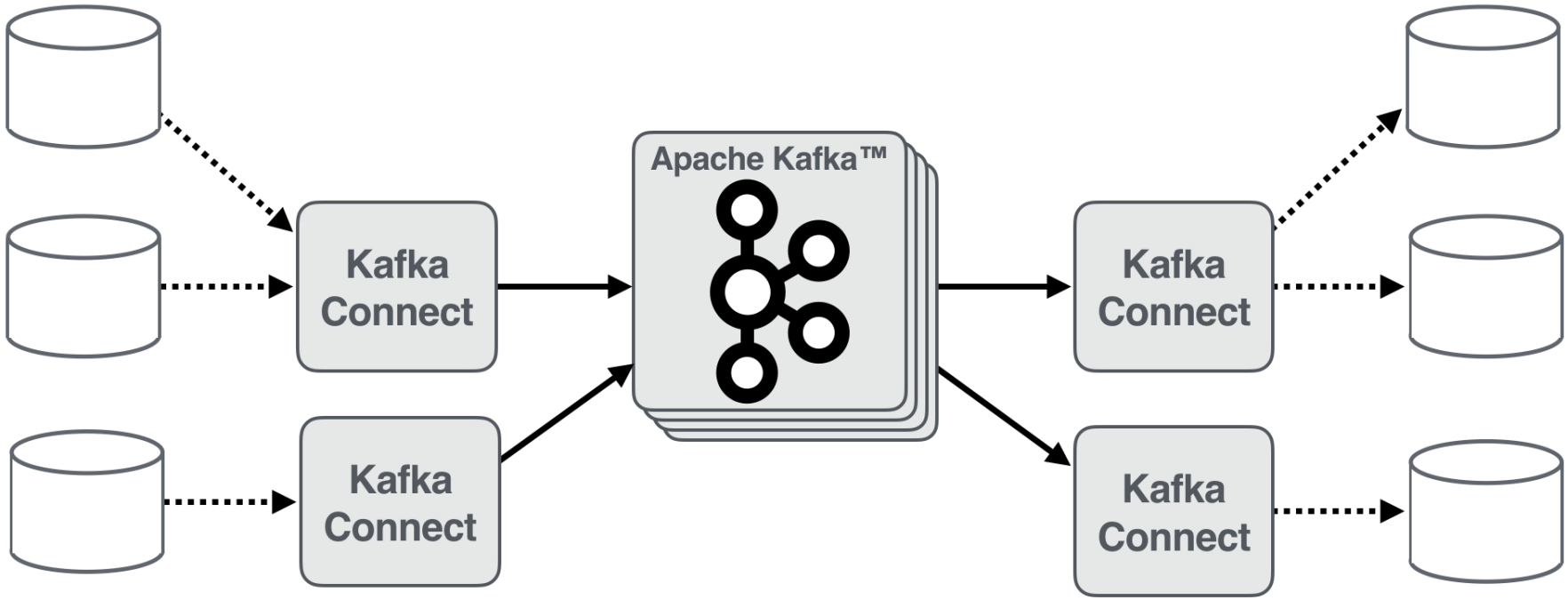


confluent

Kafka Connect : Stream data in and out of Kafka

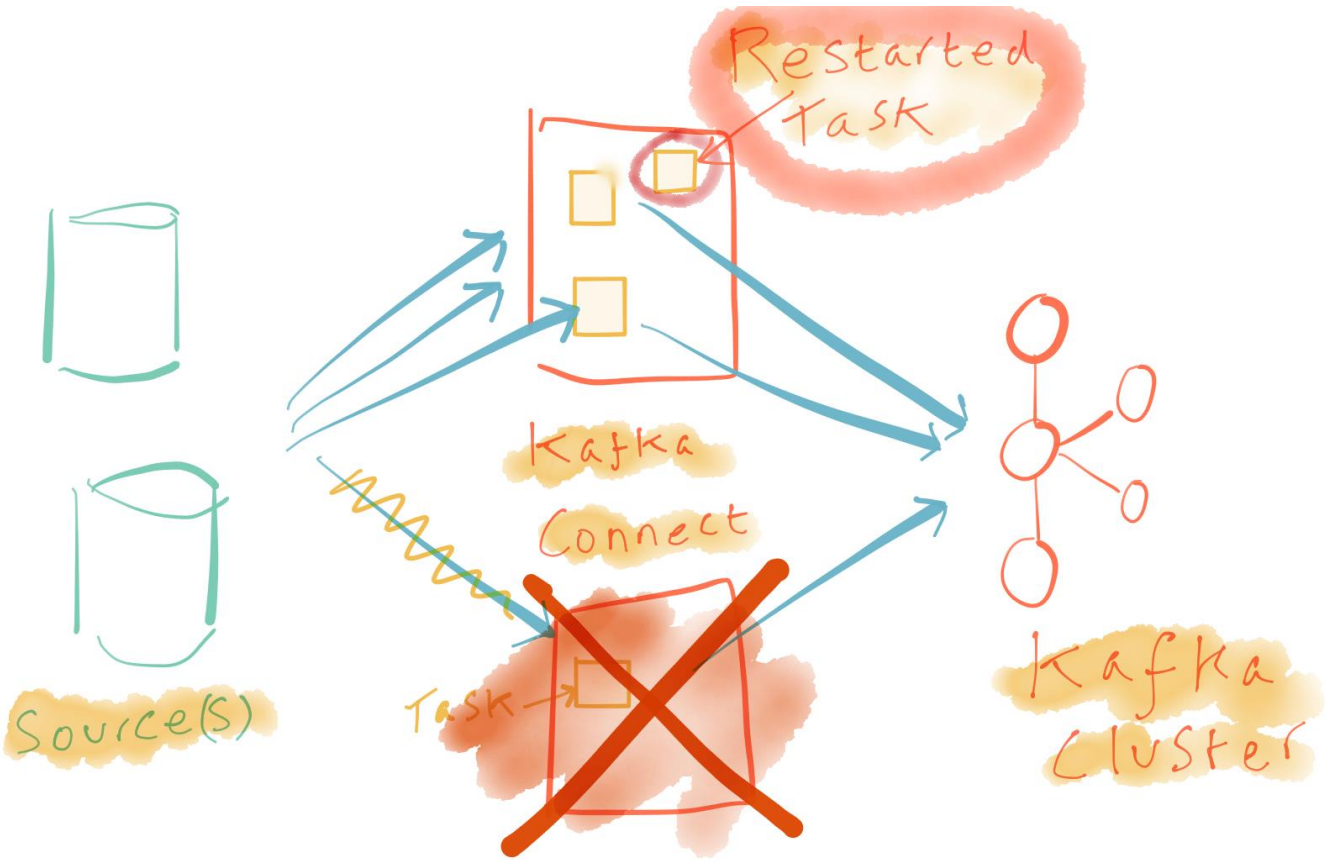


Kafka Connect : Integration at Scale

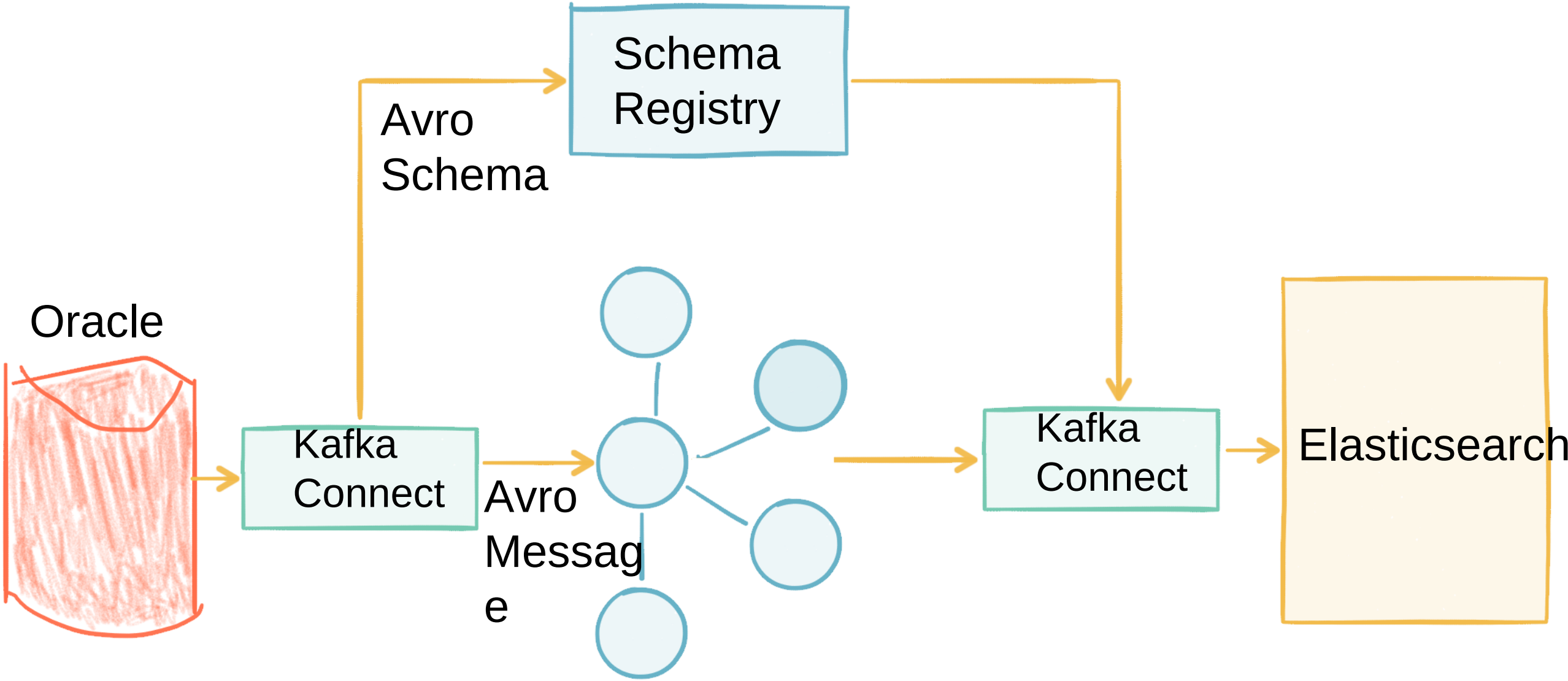


Kafka Connect – under the covers

- Each Kafka Connect node is a worker
- Each worker executes one or more tasks
- Tasks do the actual work of pulling data from sources / landing it to sinks
- Kafka Connect manages the distribution and execution of tasks
- Parallelism, fault-tolerance, load balancing all handled automatically

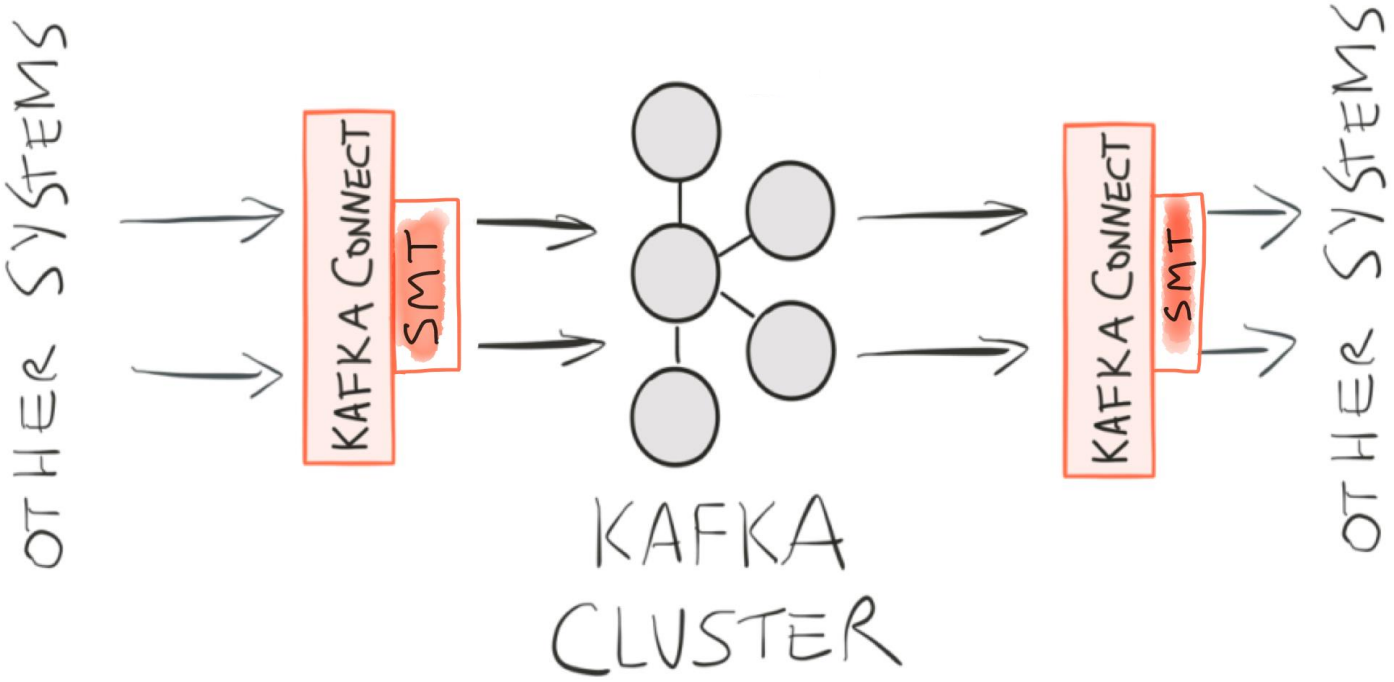


Kafka Connect + Schema Registry = WIN



Single Message Transform (SMT) -- Extract, TRANSFORM, Load…

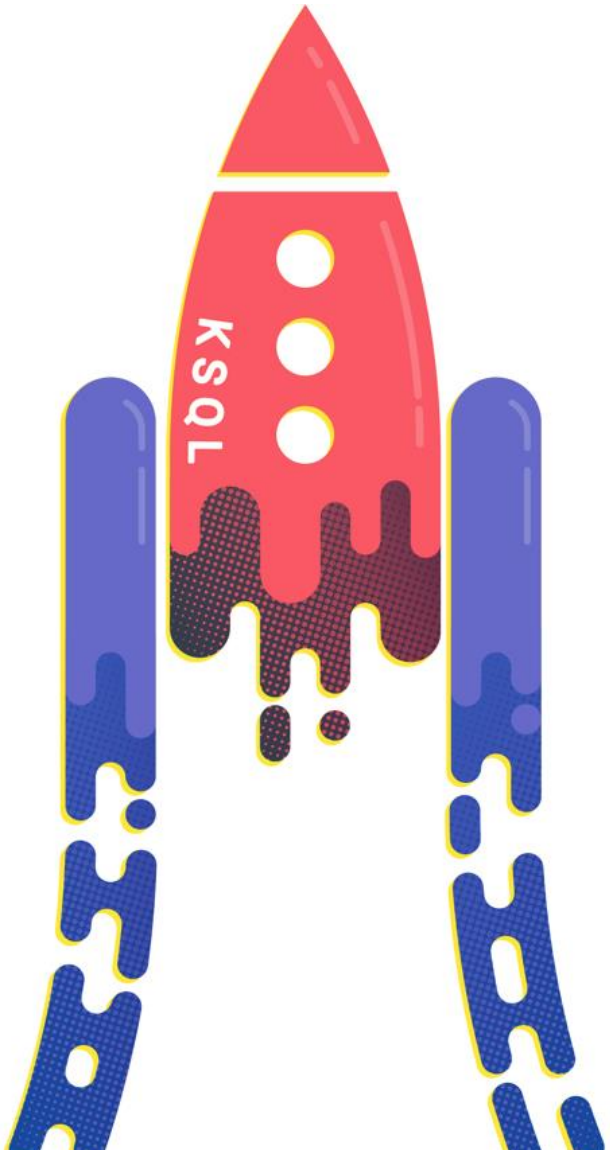
- Modify events *before storing* in Kafka:
  - Mask/drop sensitive information
  - Set partitioning key
  - Store lineage
- Modify events *going out* of Kafka:
  - Route high priority events to faster data stores
  - Direct events to different Elasticsearch indexes
  - Cast data types to match destination



But I need to join…aggregate…filter …

KSQL

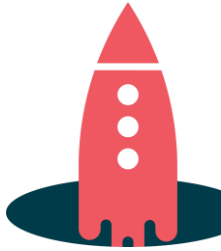
An Open Source Streaming SQL Engine for Apache Kafka™



KSQL for Data Exploration

An easy way to inspect data in a running cluster

```
SELECT status, bytes
  FROM clickstream
  WHERE user_agent =
    'Mozilla/5.0 (compatible; MSIE 6.0)';
```



KSQL for Anomaly Detection

Identifying patterns or anomalies in real-time data, surfaced in milliseconds

```
CREATE TABLE possible_fraud AS
  SELECT card_number, count(*)
    FROM authorization_attempts
    WINDOW TUMBLING (SIZE 5 SECONDS)
    GROUP BY card_number
    HAVING count(*) > 3;
```

KSQL for Real-Time Monitoring

- Log data monitoring, tracking and alerting
- Sensor / IoT data

```
CREATE TABLE error_counts AS
  SELECT error_code, count(*)
    FROM monitoring_stream
    WINDOW TUMBLING (SIZE 1 MINUTE)
    WHERE type = 'ERROR'
    GROUP BY error_code;
```

KSQL for Streaming ETL

- Kafka is popular for data pipelines.
- KSQL enables easy transformations of data within the pipe.
- Transforming data while moving from Kafka to another system.

```
CREATE STREAM vip_actions AS
  SELECT userid, page, action FROM clickstream c
    LEFT JOIN users u ON c.userid = u.user_id
    WHERE u.level = 'Platinum';
```

Where is KSQL not such a great fit?

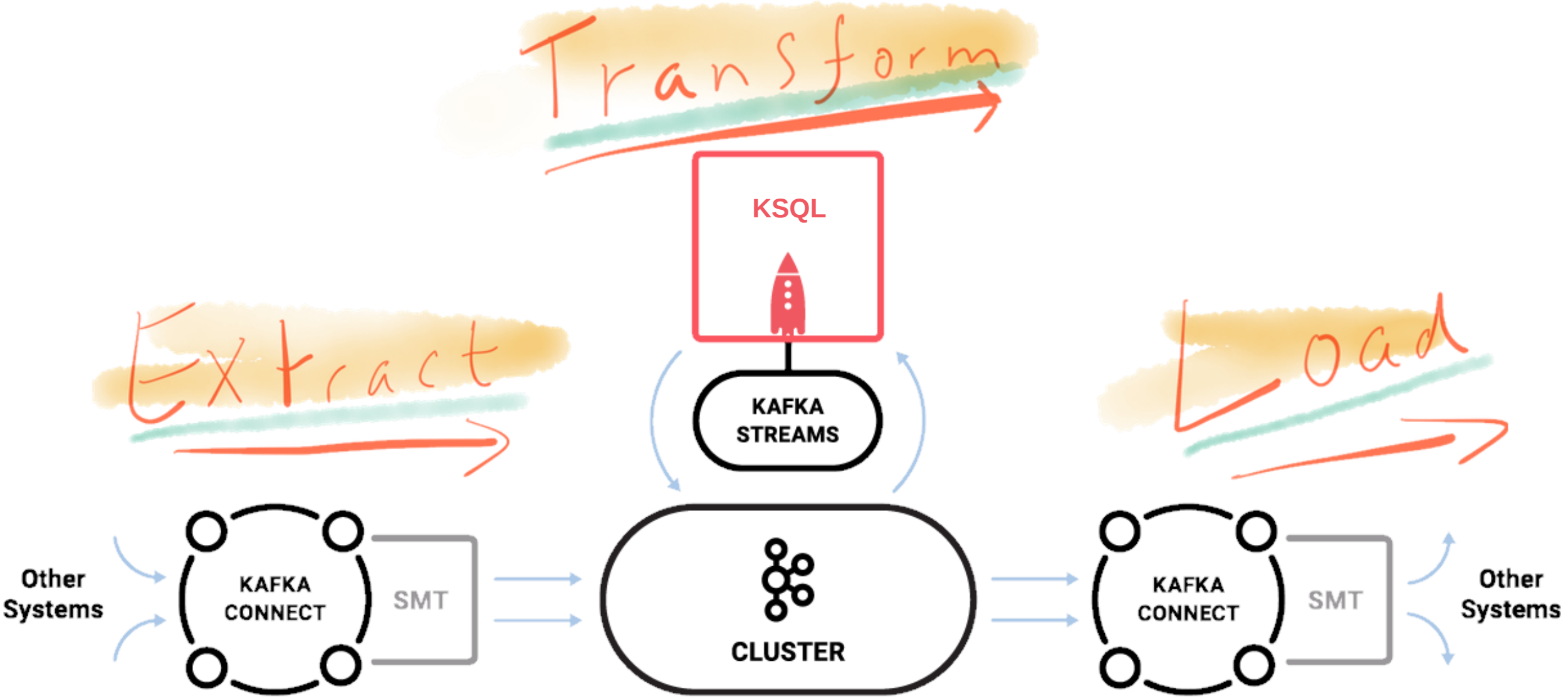
Ad-hoc queries

- Limited span of time usually retained in Kafka
- No indexes

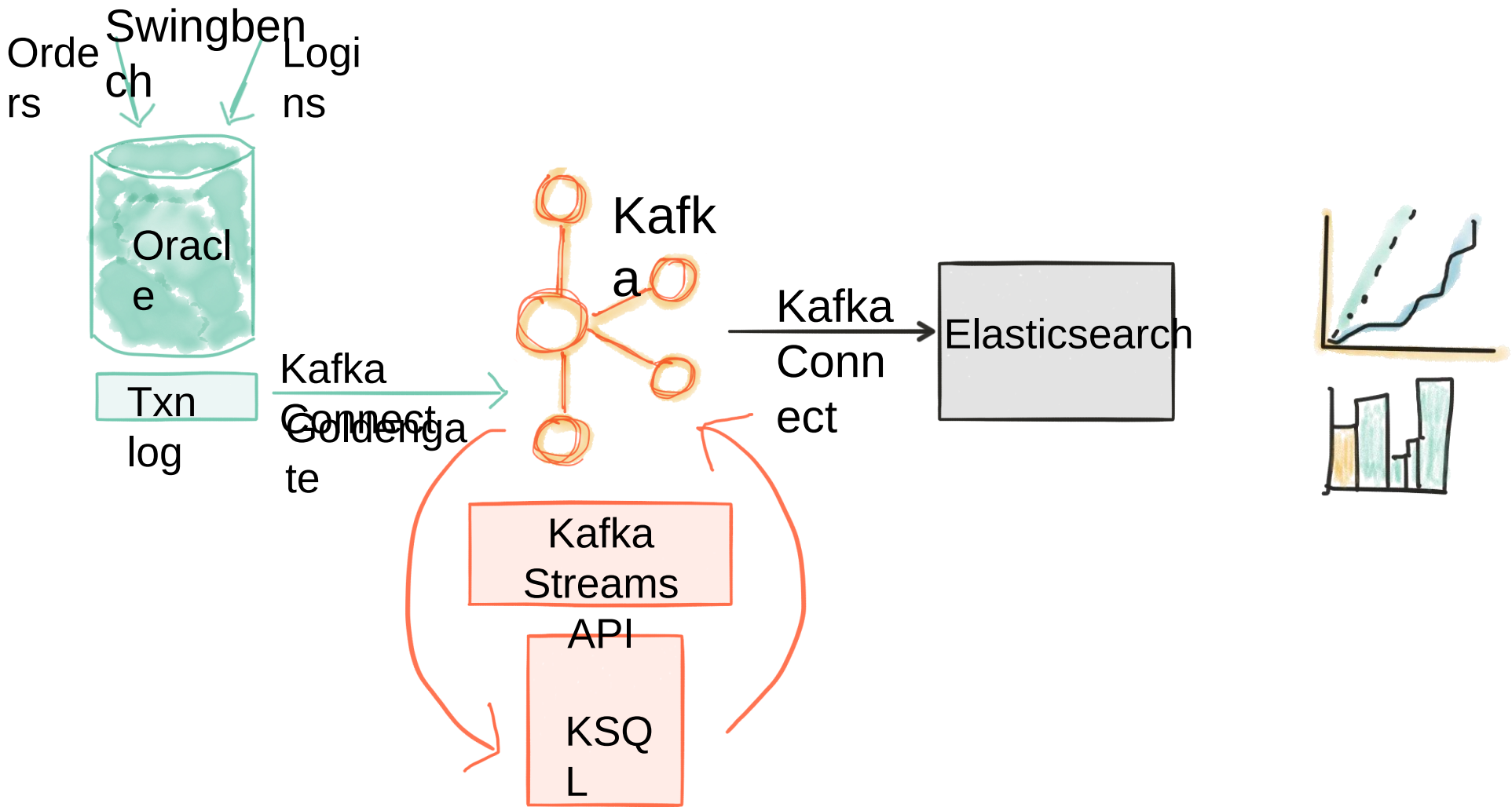
BI reports (Tableau etc.)

- No indexes
- No JDBC (most BI tools are not good with continuous results!)

Streaming ETL, powered by Apache Kafka and Confluent Platform

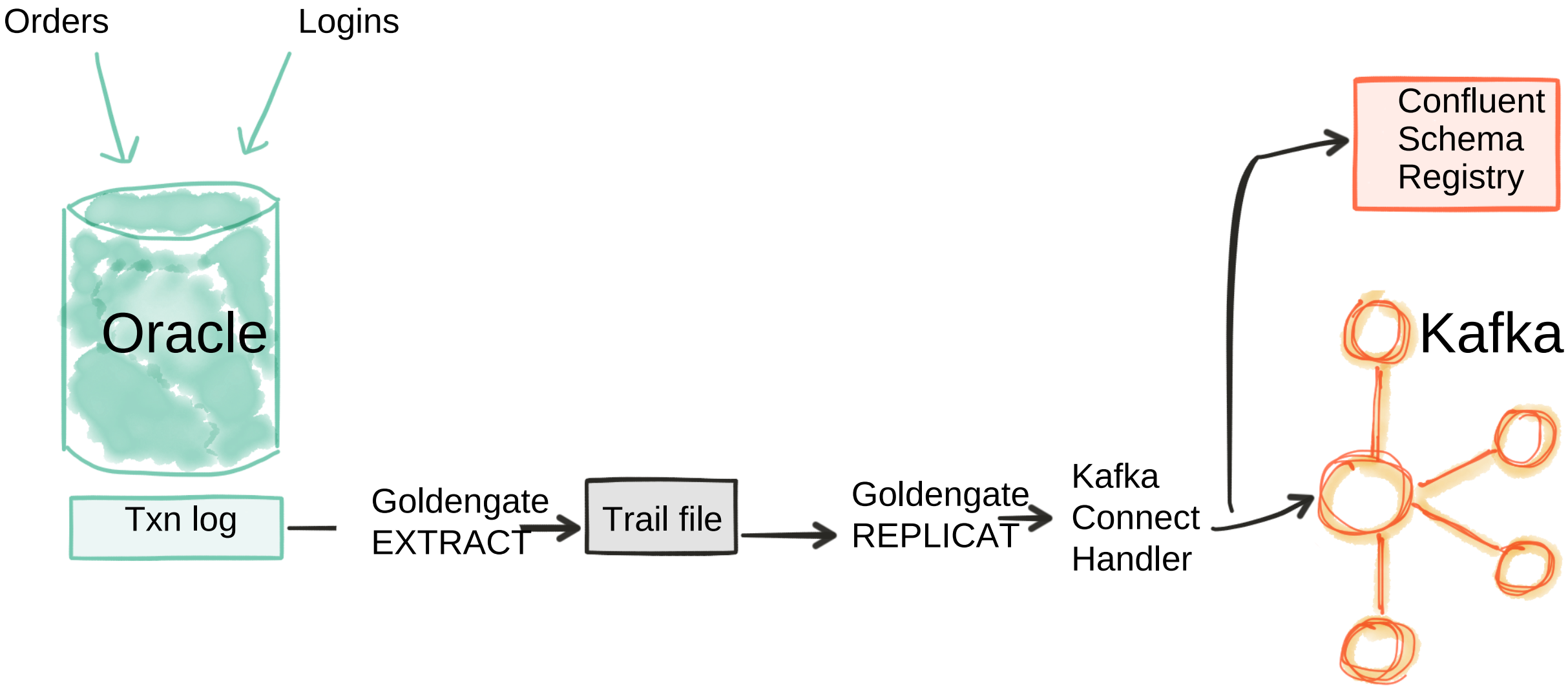


Realtime Analytics with Kafka and KSQL



https://www.confluent.io/blog/ksql-in-action-real-time-streaming-etl-from-oracle-transactional-data

GoldenGate to Kafka



https://rmoff.net/2017/11/21/installing-oracle-goldengate-for-big-data-12-3-1-with-kafka-connect-and-confluent-platform/

Smoke Test - Create row in Oracle

```
[oracle@localhost ~]$ rlwrap sqlplus SYS/oracle@orcl as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Mon Sep 11 10:14:28 2017
Copyright (c) 1982, 2016, Oracle.  All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL> INSERT INTO soe.logon VALUES(42,42,SYSDATE);

1 row created.

SQL> COMMIT;

Commit complete.
```

Smoke Test - Verify Kafka Topic

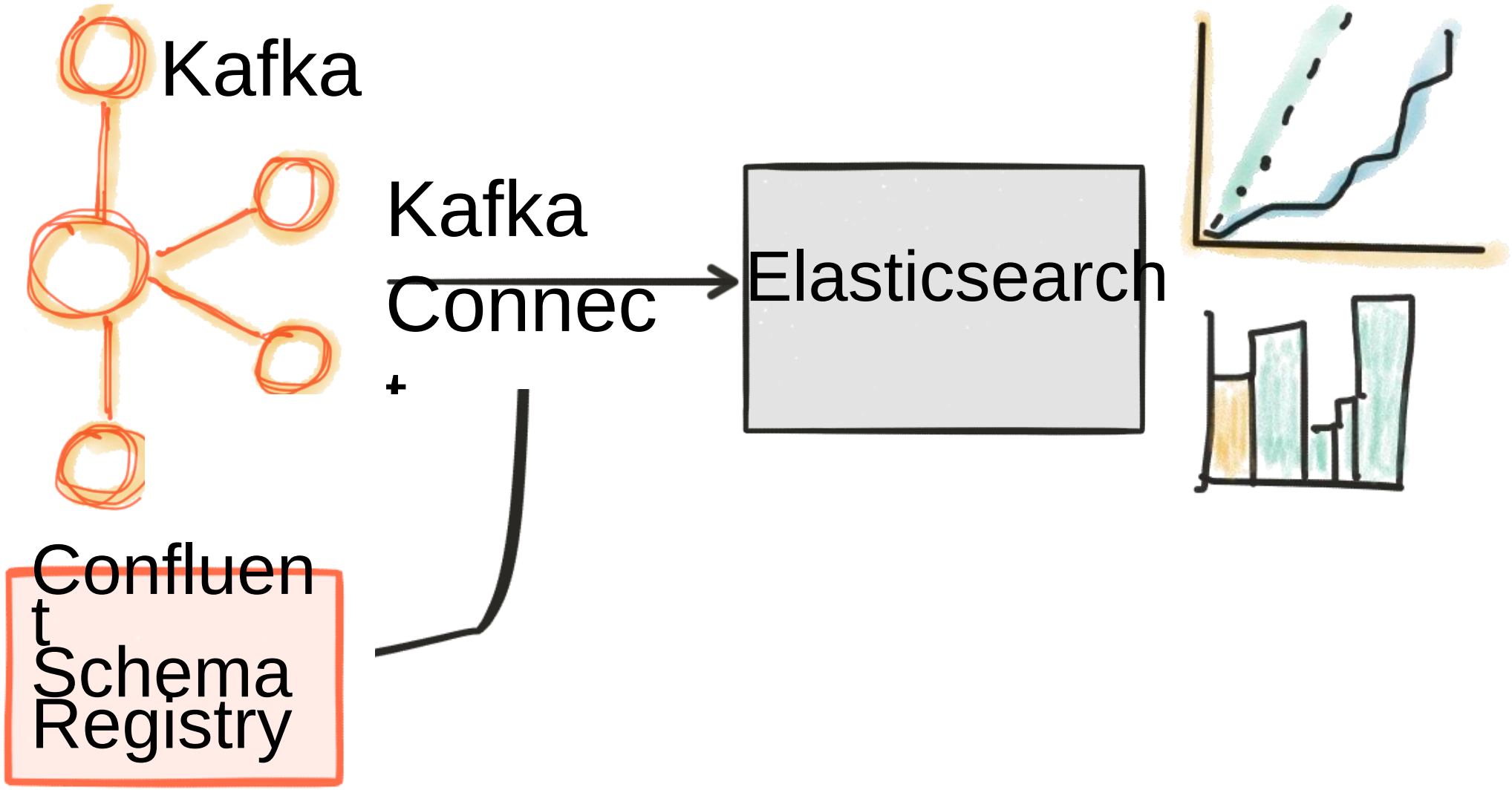
```
$ kafka-topics --zookeeper localhost:2181 --list
__confluent.support.metrics
__consumer_offsets
_schemas
connect-configs
connect-offsets
connect-statuses
ora-ogg-SOE-LOGON-avro
```

Smoke Test - Verify Kafka Message

```
$ kafka-avro-console-consumer
--bootstrap-server localhost:9092 \
--property schema.registry.url=http
--topic ora-ogg-SOE-LOGON-avr
```


```
"table": {
  "string": "ORCL.SOE.LOGON"
},
"op_type": {
  "string": "I"
},
"op_ts": {
  "string": "2017-09-11 14:14:39.000000"
},
"LOGON_ID": {
  "double": 42
},
"CUSTOMER_ID": {
  "double": 42
},
"LOGON_DATE": {
  "string": "2017-09-11 10:14:30"
```

Realtime Analytics with Kafka and KSQL



Configuring Kafka Connect

```
"connector.class": "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
"connection.url": "http://localhost:9200",
"topics": "ora-ogg-SOE-ORDERS-avro",
"topic.index.map": "ora-ogg-SOE-ORDERS-avro:soe.orders",
"transforms": "convert_op_ts",
"transforms.convert_op_ts.type":
"org.apache.kafka.connect.transforms.TimestampConverter$Value",
"transforms.convert_op_ts.target.type": "Timestamp",
"transforms.convert_op_ts.field": "op_ts",
"transforms.convert_op_ts.format": "yyyy-MM-dd HH:mm:ss.SSSSSS",
```

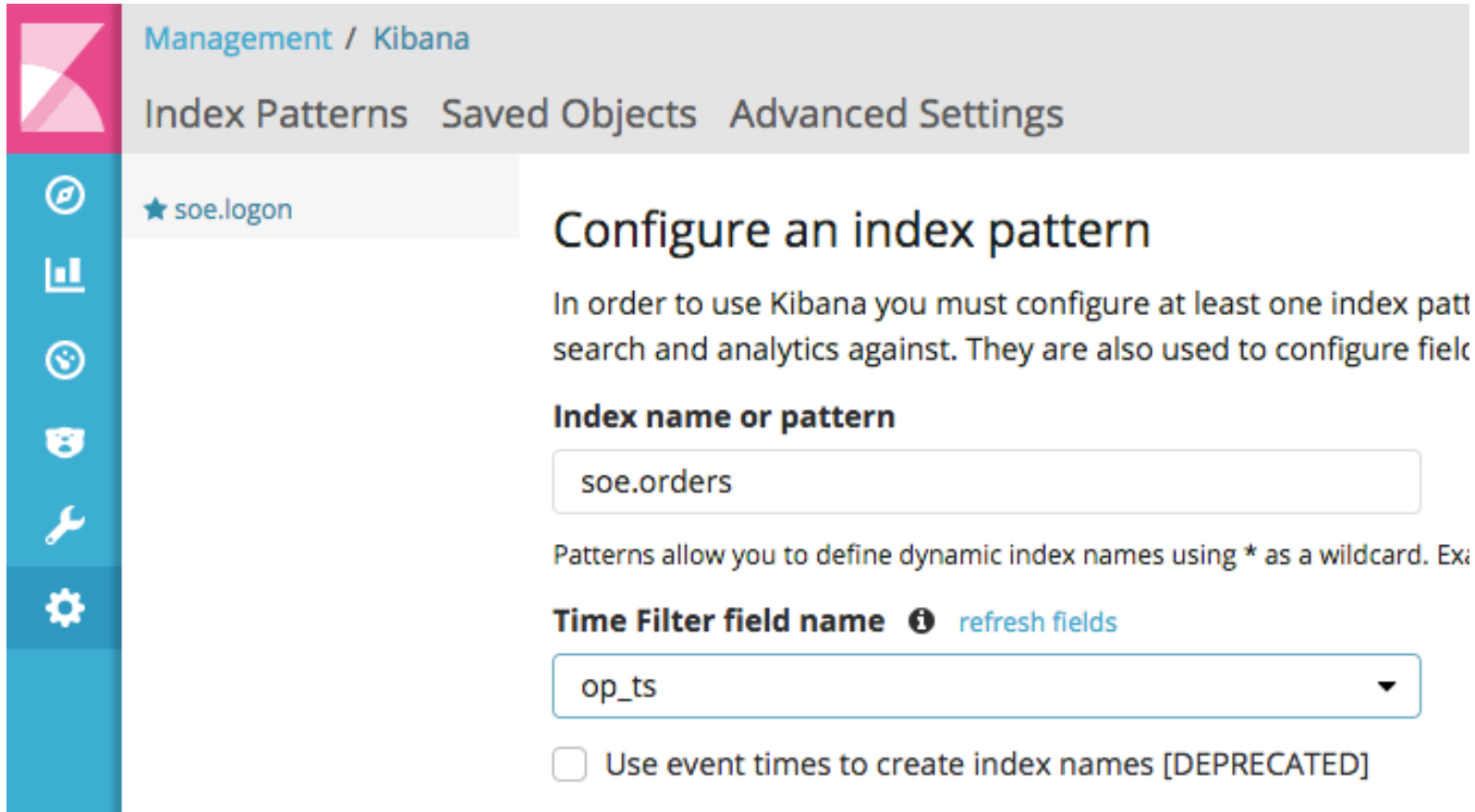
Load the Kafka Connect Elasticsearch Sink Configuration

```
$ confluent load es-sink-soe -d es-sink-soe.json
```

Checking the data in Elasticsearch

```
$ curl -s "http://localhost:9200/soe.orders/_search" | jq
'.hits.total'
```

1000



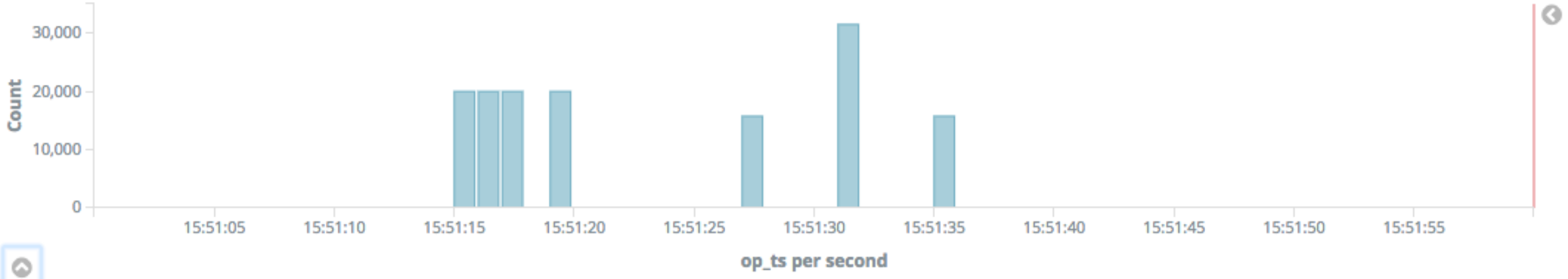
Realtime stream of data from Oracle to Elasticsearch via Kafka

142,979 hits New Save Open Share September 13th 2017, 15:51:00.000 to September 13th 2017, 15:52:00.000

Search... (e.g. status:200 AND extension:PHP) Uses lucene query syntax

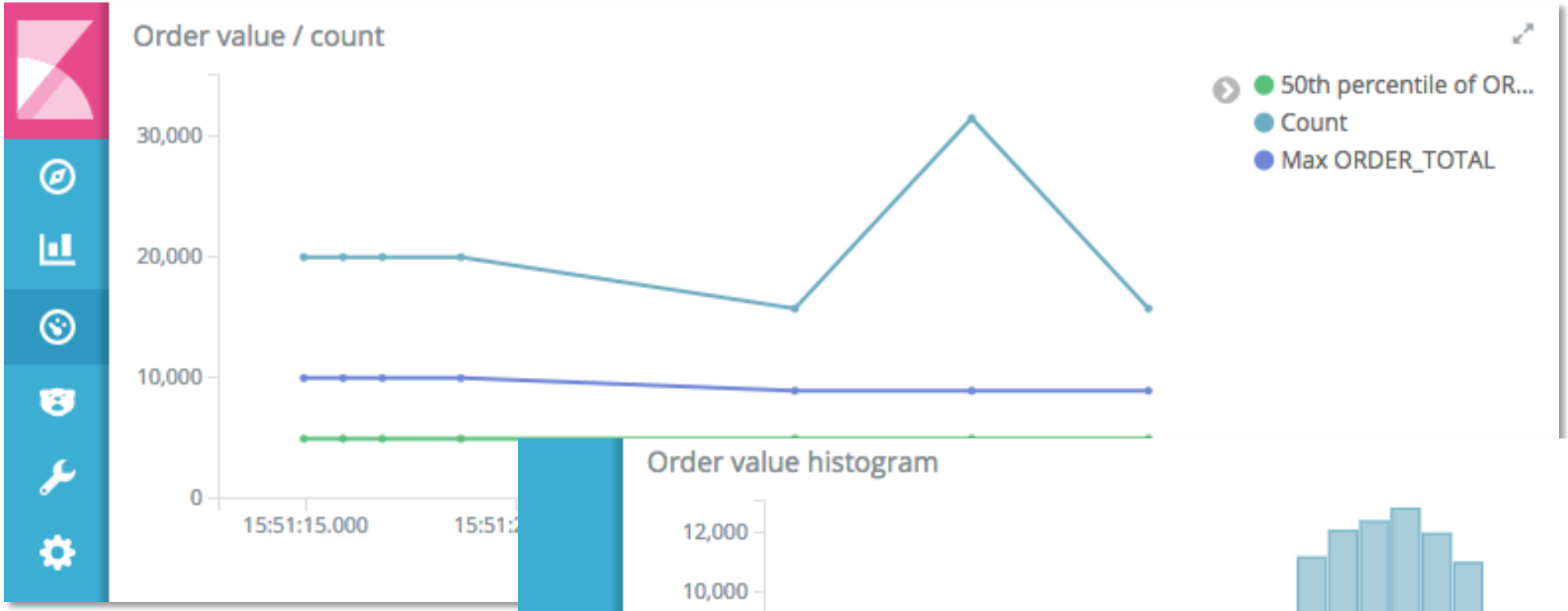
Add a filter +

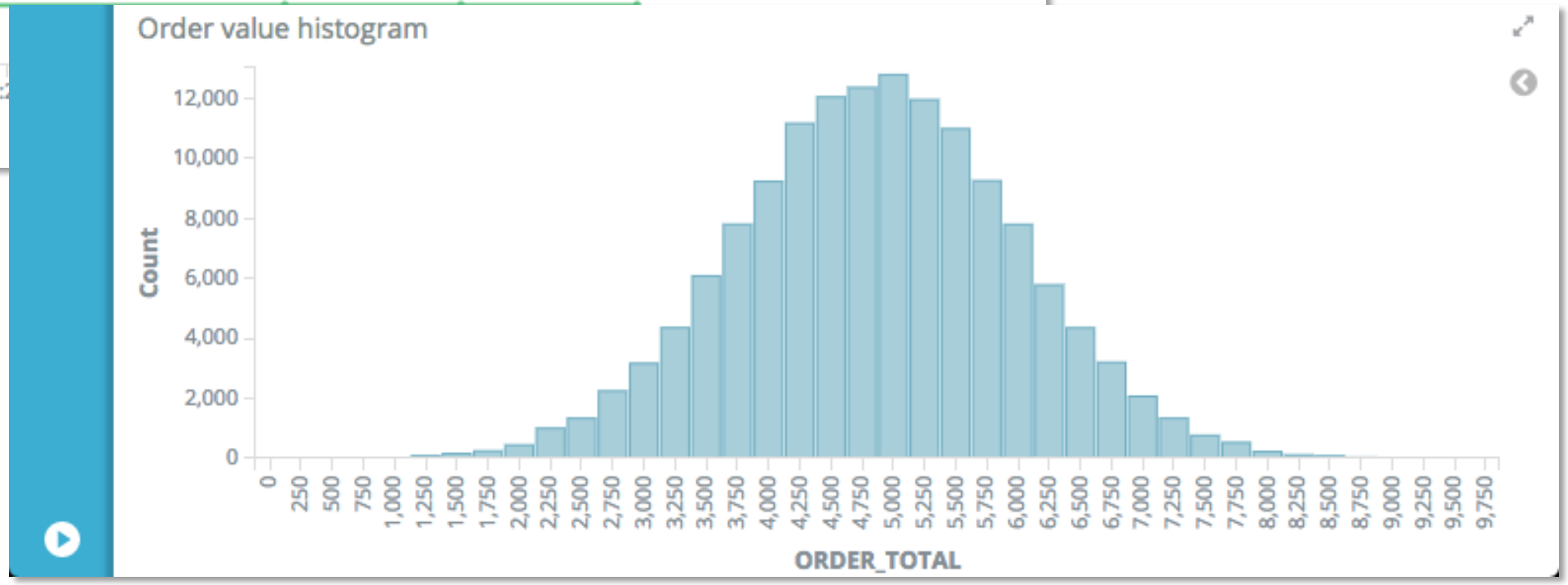
September 13th 2017, 15:51:00.000 - September 13th 2017, 15:52:00.000 — Auto



Time	CUSTOMER_CLASS	ORDER_MODE	ORDER_TOTAL	WAIT_TILL_ALL_AVAILABLE
September 13th 2017, 15:51:35.000	Prime	direct	2,366	ship_when_ready
September 13th 2017, 15:51:35.000	Occasional	online	3,039	ship_when_ready
September 13th 2017, 15:51:35.000	Occasional	direct	4,242	ship_asap
September 13th 2017, 15:51:35.000	Business	direct	4,594	ship_asap
September 13th 2017, 15:51:35.000	Occasional	direct	5,423	ship_when_ready

Realtime Analytics on Raw Stream of data from Oracle via Kafka

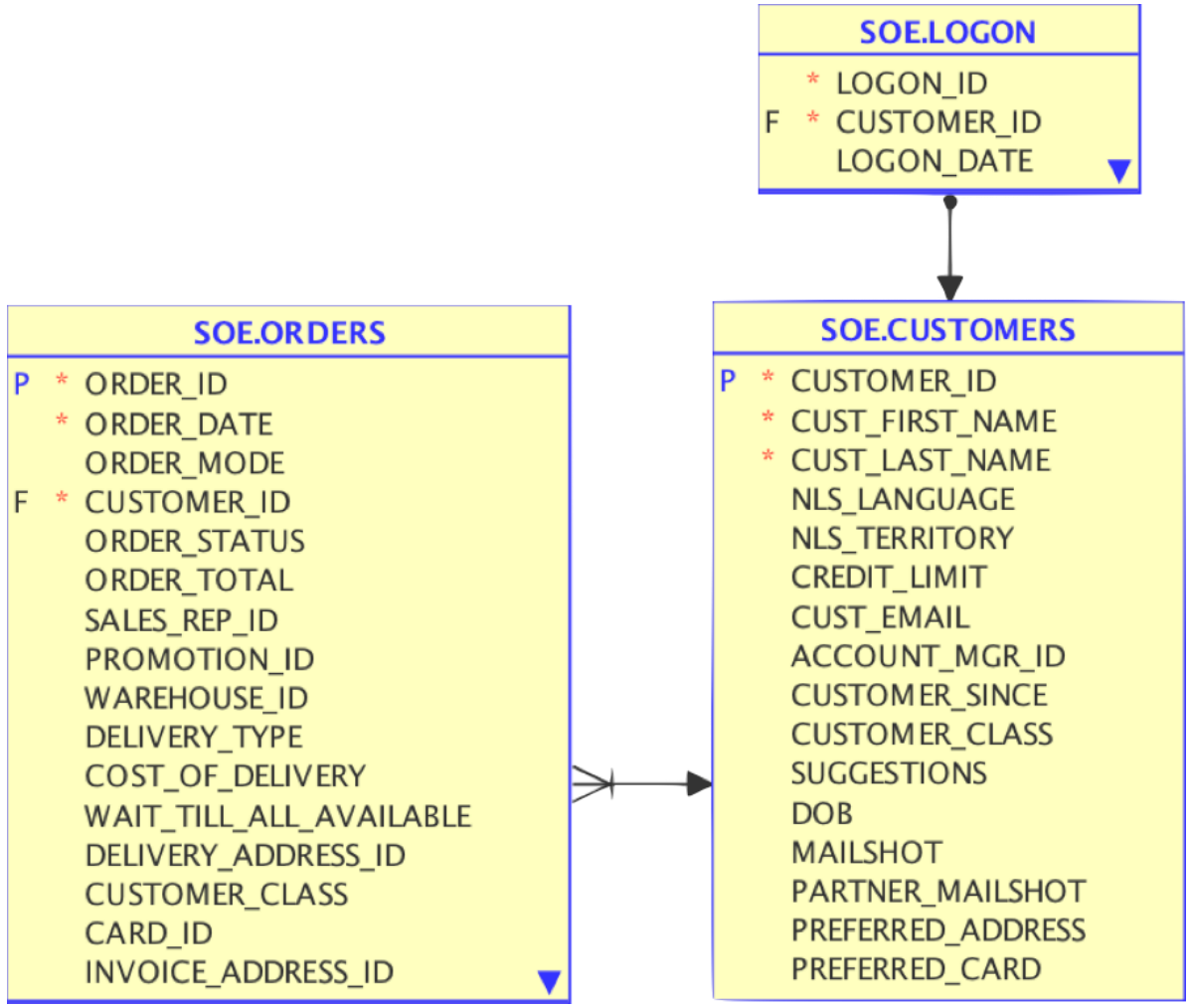




confluent

Realtime Stream Processing with Kafka and KSQL

- Enrich customer logon events and add 'Customer Class' data in order to filter some events
- Rolling aggregates of order values and counts
- **…all in realtime with continuous, event-by-event, processing!**

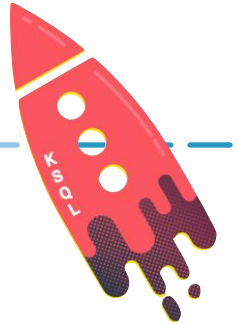


Join the Streams…
(but never cross them)

KSQL - Declare the Customers Table

```
ksql> CREATE TABLE CUSTOMERS
        (CUSTOMER_ID INT,
        CUST_FIRST_NAME STRING,
        CUST_LAST_NAME STRING,
        CUSTOMER_CLASS STRING)
        WITH
        (KAFKA_TOPIC='ora-ogg-SOE-CUSTOMERS',
        VALUE_FORMAT='JSON');
```

KSQL - Query the live stream of Kafka data!

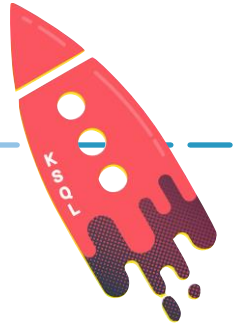



```
ksql> SELECT CUSTOMER_ID,
          CUST_FIRST_NAME,
          CUST_LAST_NAME,
          CUSTOMER_CLASS
        FROM CUSTOMERS
        LIMIT 3;
75003 | karl | greene | Occasional
75010 | samuel | cook | Prime
75012 | paul | taylor | Occasional
```

KSQL - Declare the Stream of logon events

```
ksql> CREATE STREAM LOGON
        (LOGON_ID INT,
        CUSTOMER_ID INT,
        LOGON_DATE STRING)
        WITH
        (KAFKA_TOPIC='ora-ogg-SOE-LOGON-json',
VALUE_FORMAT='JSON');
```

KSQL - Query the live stream of Kafka data!

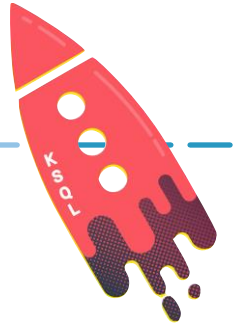



```
ksql> SELECT LOGON_ID,CUSTOMER_ID,LOGON_DATE
            FROM LOGON LIMIT 5;
178724 | 31809 | 2000-11-08 23:08:51
178725 | 91808 | 2009-06-29 02:38:11
178726 | 78742 | 2007-11-06 15:29:38
178727 | 4565 | 2010-03-25 09:31:44
178728 | 20575 | 2000-05-31 00:22:00
```

KSQL - Join Logon Events to Customer Reference Data

```
ksql> SELECT L.LOGON_ID,
            C.CUSTOMER_ID,
            C.CUST_FIRST_NAME, C.CUST_LAST_NAME,
C.CUSTOMER_CLASS
        FROM LOGON L
        LEFT OUTER JOIN
        CUSTOMERS C
        ON L.CUSTOMER_ID = C.CUSTOMER_ID;
```

KSQL - Join Logon Events to Customer Reference Data



178771 | 75515 | earl | butler | 2002-07-19 00:00:00 | Occasional

178819 | 76851 | cesar | mckinney | 2000-10-07 00:00:00 | Regular

178832 | 77941 | randall | tucker | 2010-04-23 00:00:00 | Prime

178841 | 80769 | ramon | hart | 2011-01-24 00:00:00 | Occasional

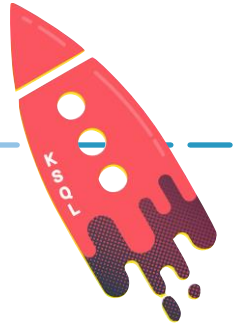
178870 | 77064 | willard | curtis | 2009-05-26 00:00:00 | Occasional

[...]

KSQL - Derive and Calcuate columns on the fly

```
ksql> CREATE STREAM LOGON_ENRICHED AS
  SELECT CONCAT(C.CUST_FIRST_NAME ,CONCAT(' ',C.CUST_LAST_NAME)) AS CUST_FULL_NAME,
  (CAST(C.ROWTIME AS DOUBLE)-CAST(STRINGTOTIMESTAMP(CUSTOMER_SINCE,'yyyy-MM-dd HH:mm:ss') AS DOUBLE))/ 60 / 60 / 24 / 1000/365 AS CUSTOMER_SINCE_YRS
  FROM LOGON L LEFT OUTER JOIN CUSTOMERS C
  ON L.CUSTOMER_ID = C.CUSTOMER_ID ;
```

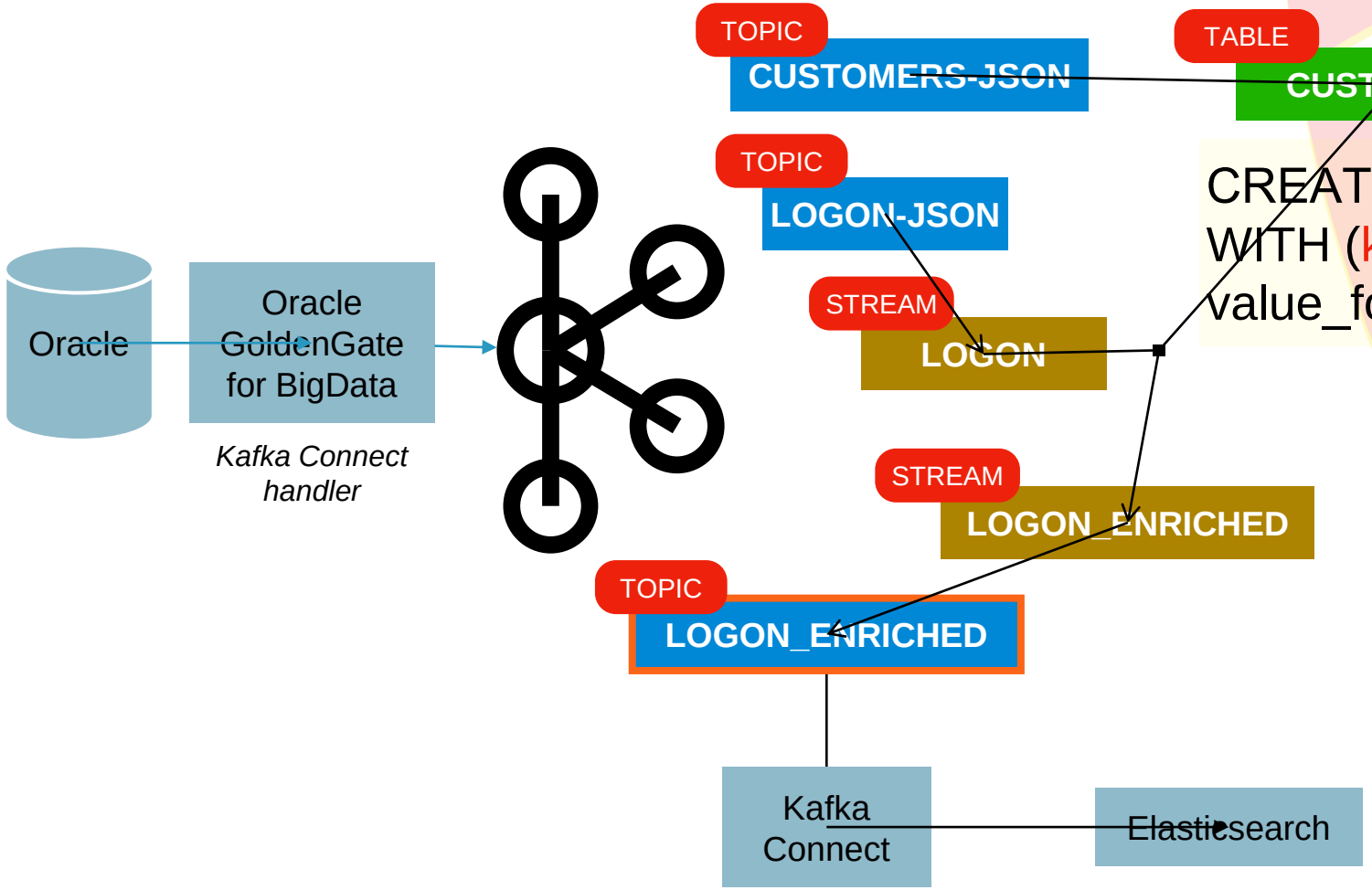
KSQL - Filter Streams of Data from Kafka




```
ksql> SELECT LOGON_ID, LOGON_DATE, CUST_FULL_NAME,
        CUSTOMER_CLASS, CUSTOMER_SINCE_YRS
      FROM LOGON_ENRICHED
      WHERE CUSTOMER_CLASS = 'Prime' AND
CUSTOMER_SINCE_YRS > 5;
```

```
181362 | 2011-02-16 13:01:16 | isaac wong | Prime | 10.771086241850583
181551 | 2007-01-15 11:21:19 | ryan turner | Prime | 6.762867074898529
181576 | 2009-07-04 02:19:35 | peter campbell | Prime | 14.779305415810505
181597 | 2006-07-12 04:54:40 | andres fletcher | Prime | 13.782045160768645
181631 | 2002-09-08 03:06:16 | john johnson | Prime | 6.762867062690258
```

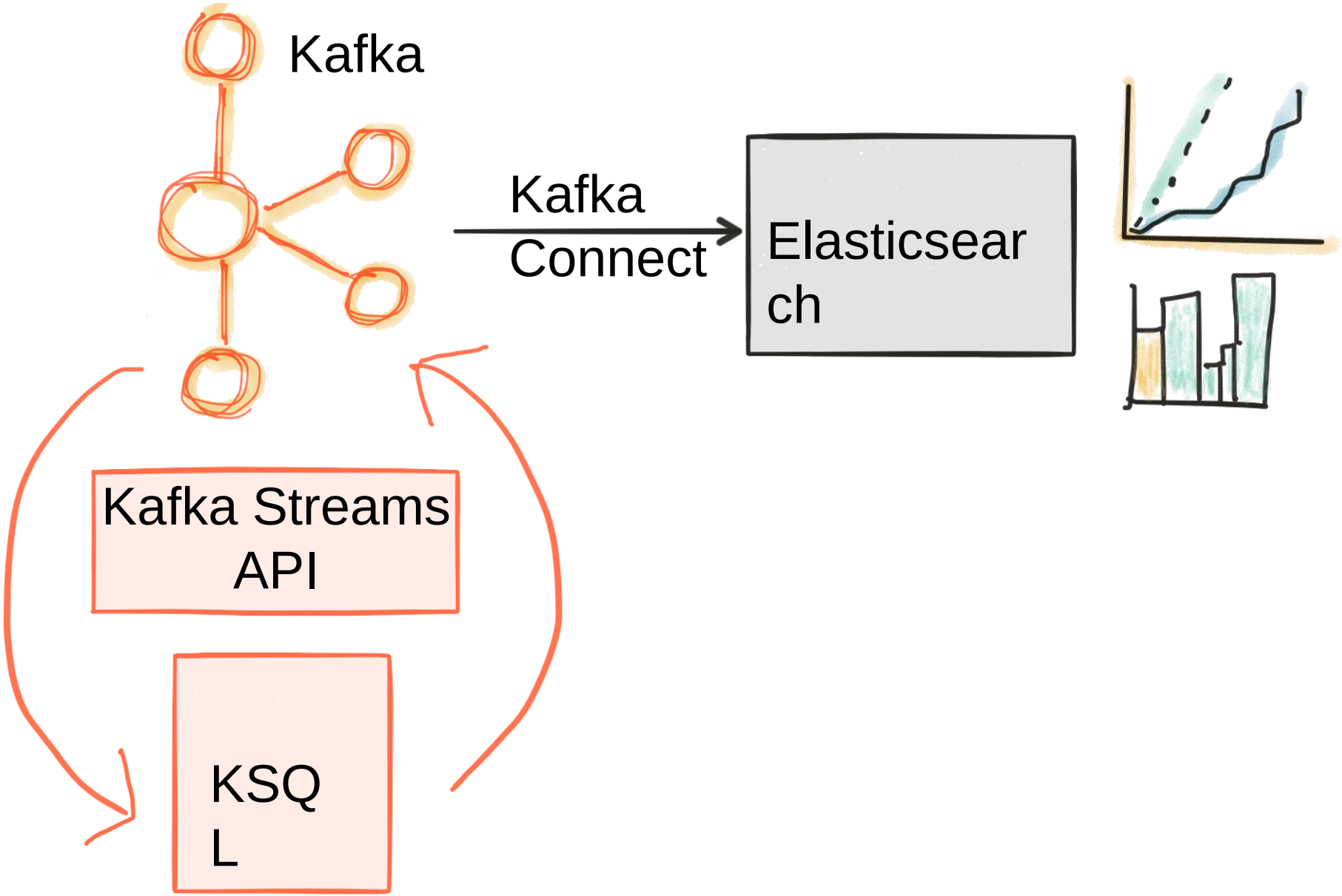
```
CREATE TABLE CUSTOMERS (CUSTOMER_ID INT…)
WITH (kafka_topic='CUSTOMERS-JSON',
value_format='JSON');
```




```
CREATE STREAM LOGON (LOGON_ID INT, …)
WITH (kafka_topic='LOGON-JSON',
value_format='JSON');
```

```
CREATE STREAM LOGON_ENRICHED AS
SELECT L.LOGON_ID, C.CUSTOMER_ID.
FROM LOGON L
LEFT OUTER JOIN CUSTOMERS C
ON L.CUSTOMER_ID = C.CUSTOMER_ID
```

Realtime Analytics with Kafka and KSQL



logon_enriched

September 27th 2017, 18:41:43.616 - September 27th 2017, 19:01:26.098 — Auto

Selected Fields

CUSTOM...

t CUST_EM...

t CUST_FU...

Available Fields

t CUSTOMER_... add

Quick Count
(500 /500 records)

Occasional 48.6%

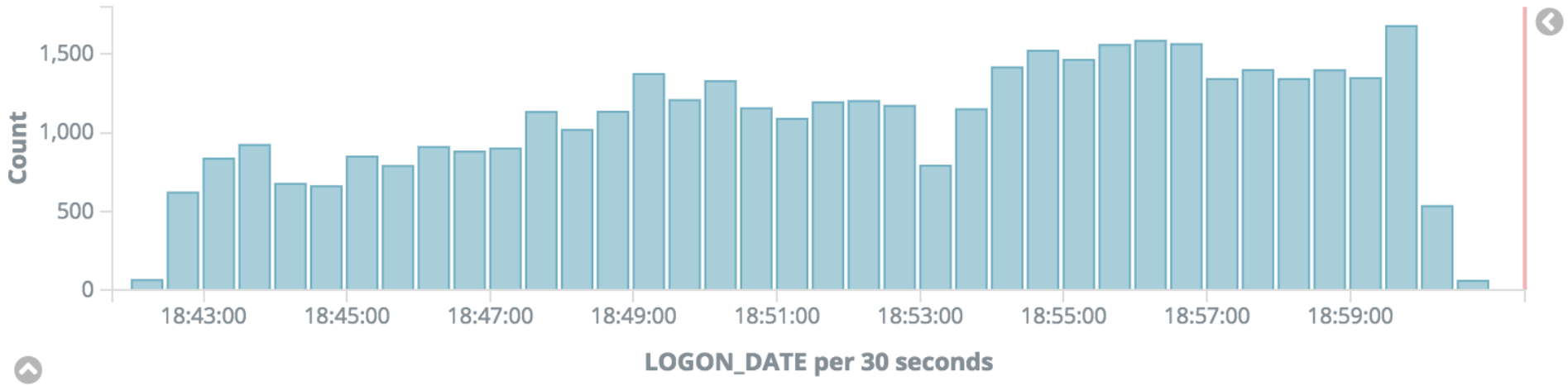
Prime 19.6%

Ocasional 13.8%

Regular 9.2%

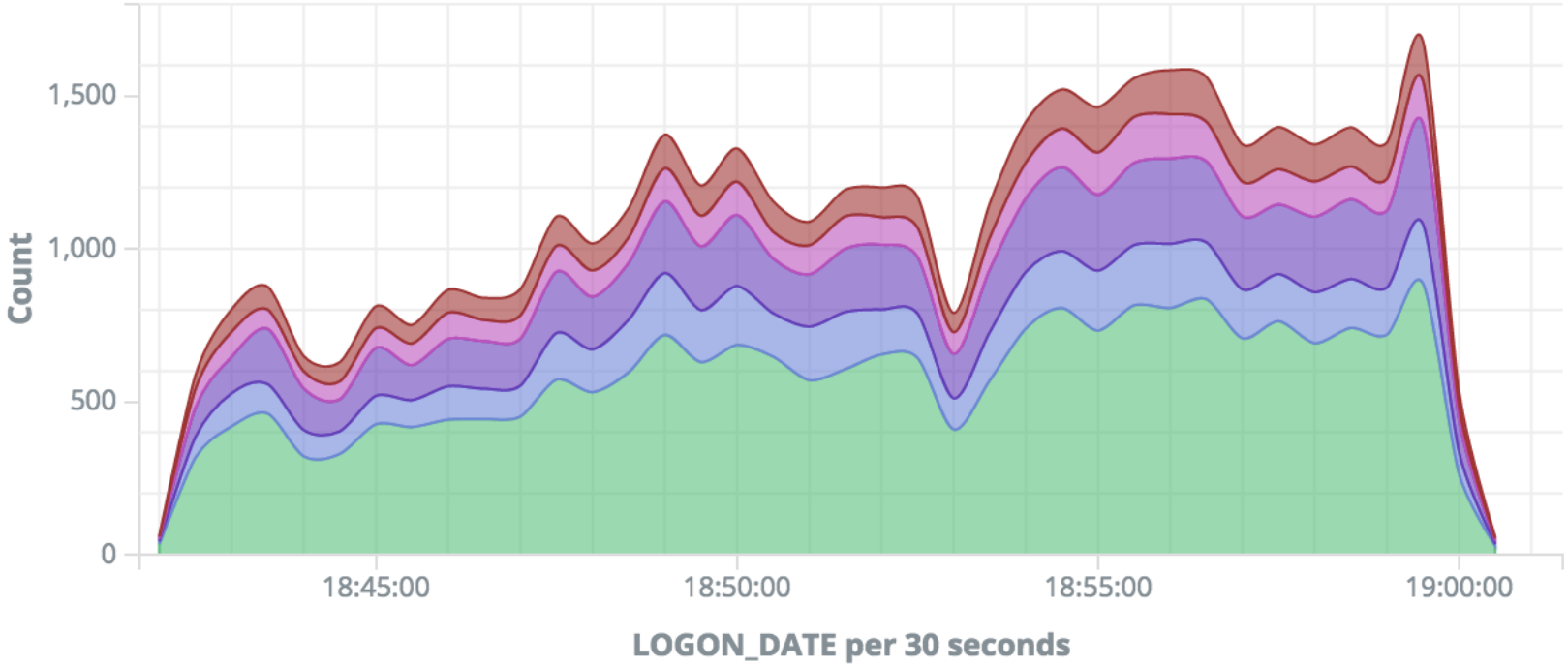
Business 8.8%

t CUSTOMER_...



Time	CUST_FULL_NAME	CUST_EMAIL	CUSTOMER_SINCE_YRS
September 27th 2017, 19:00:40.000	brian white	brian.white@virgin.com	12.804
September 27th 2017, 19:00:39.000	orville gill	orville.gill@yahoo.com	15.815
September 27th 2017, 19:00:39.000	jimmy barnes	jimmy.barnes@msn.com	9.812
September 27th 2017, 19:00:39.000	emory webb	emory.webb@oracle.com	2.812
September 27th 2017, 19:00:39.000	cory burke	cory.burke@ntlworld.com	12.804
September 27th 2017, 19:00:39.000	salvador douglas	salvador.douglas@msn.com	5.804

Logons by Customer Class



Login count vs target

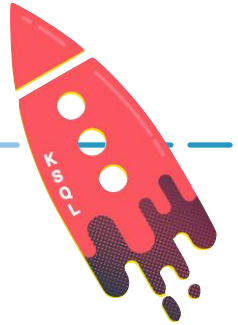
Login count vs target

Logon Detail

1–50 of 41,231

Time	CUST_FULL_NAME	CUST_EMAIL	CUSTOMER_SINCE_YRS
September 27th 2017, 19:00:40.000	brian white	brian.white@virgin.com	12.804
September 27th 2017, 19:00:39.000	orville gill	orville.gill@yahoo.com	15.815
September 27th 2017, 19:00:39.000	glenn hayes	glenn.hayes@verizon.com	6.801
September 27th 2017, 19:00:39.000	jimmy barnes	jimmy.barnes@msn.com	9.812

KSQL - Building Streaming Aggregates



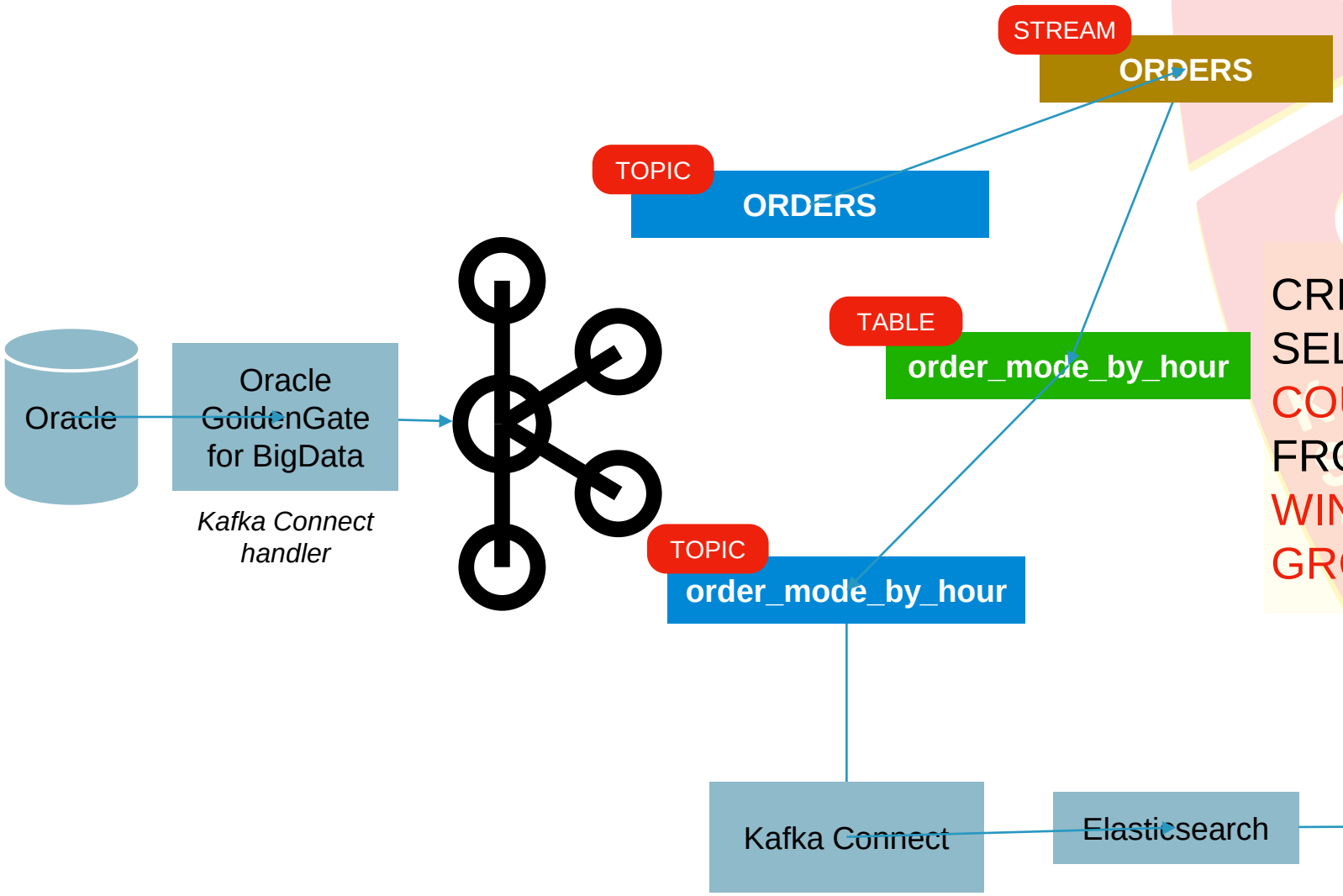

```
ksql> CREATE TABLE ORDERS_AGG_HOURLY AS
        SELECT ORDER_STATUS,
        COUNT(*) AS ORDER_COUNT,
        MAX(ORDER_TOTAL) AS MAX_ORDER_TOTAL,
SUM(ORDER_TOTAL) AS SUM_ORDER_TOTAL
        FROM ORDERS
        WINDOW TUMBLING (SIZE 1 HOUR)
        GROUP BY ORDER_STATUS;
```

KSQL - Building Streaming Aggregates

```
ksql> SELECT TIMESTAMPTOSTRING(ROWTIME, 'yyyy-MM-dd HH:mm:ss') ,
        ORDER_STATUS, MAX_ORDER_TOTAL, ORDER_COUNT,
SUM_ORDER_TOTAL
        FROM ORDERS_AGG_HOURLY LIMIT 5;
2008-04-21 16:00:00 | 4 | 4067.0 | 1 | 4067.0
2007-11-20 21:00:00 | 4 | 3745.0 | 1 | 3745.0
2008-08-24 06:00:00 | 7 | 7354.0 | 1 | 7354.0
2008-03-25 05:00:00 | 3 | 2269.0 | 1 | 2269.0
2009-11-13 23:00:00 | 3 | 2865.0 | 1 | 2865.0
```

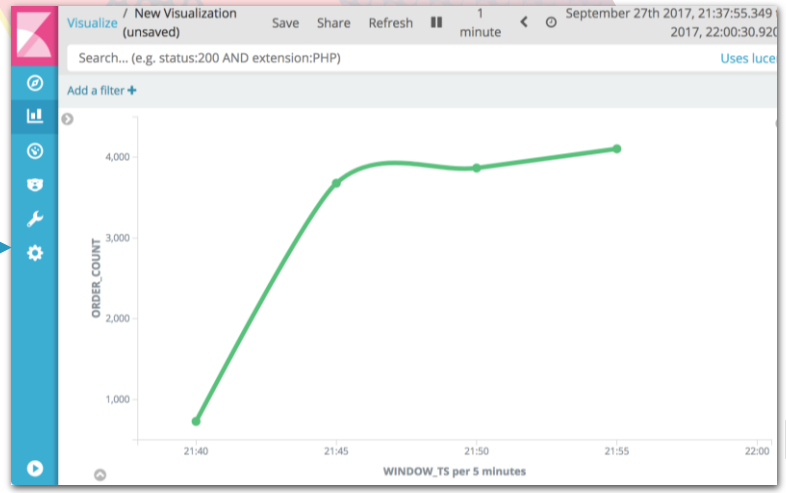
KSQL - Building Streaming Aggregates

```
$ kafkacat -C -c1 -b localhost:9092 -t ORDERS_AGG_HOURLY_WITH_WINDOW -p -K
':'
{
  "MAX_ORDER_TOTAL": 4735,
  "ORDER_COUNT": 1,
  "WINDOW_START_TS": "2008-04-23 15:00:00",
  "ORDER_STATUS": 5,
  "SUM_ORDER_TOTAL": 4735
}
```




```
CREATE STREAM ORDERS
(ORDER_DATE STRING …
WITH (KAFKA_TOPIC='ORDERS',
VALUE_FORMAT='JSON');
```

```
CREATE TABLE ORDERS_HOURLY AS
SELECT ORDER_STATUS,
COUNT(*) as ORDER_COUNT
FROM ORDERS
WINDOW TUMBLING (SIZE 1 HOUR)
GROUP BY ORDER_STATUS;
```

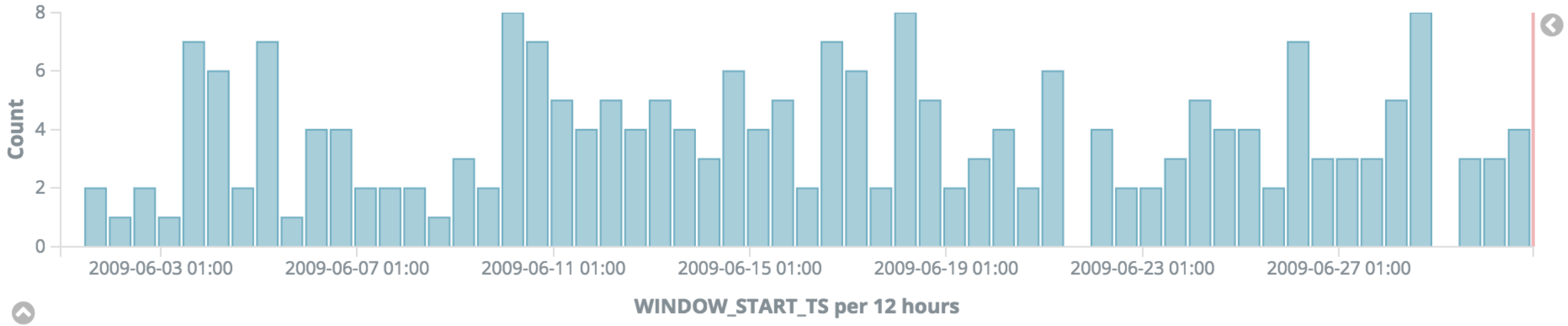


221 hits New Save Open Share June 1st 2009, 00:00:00.000 to July 1st 2009, 00:00:00.000

Search... (e.g. status:200 AND extension:PHP) Uses lucene query syntax

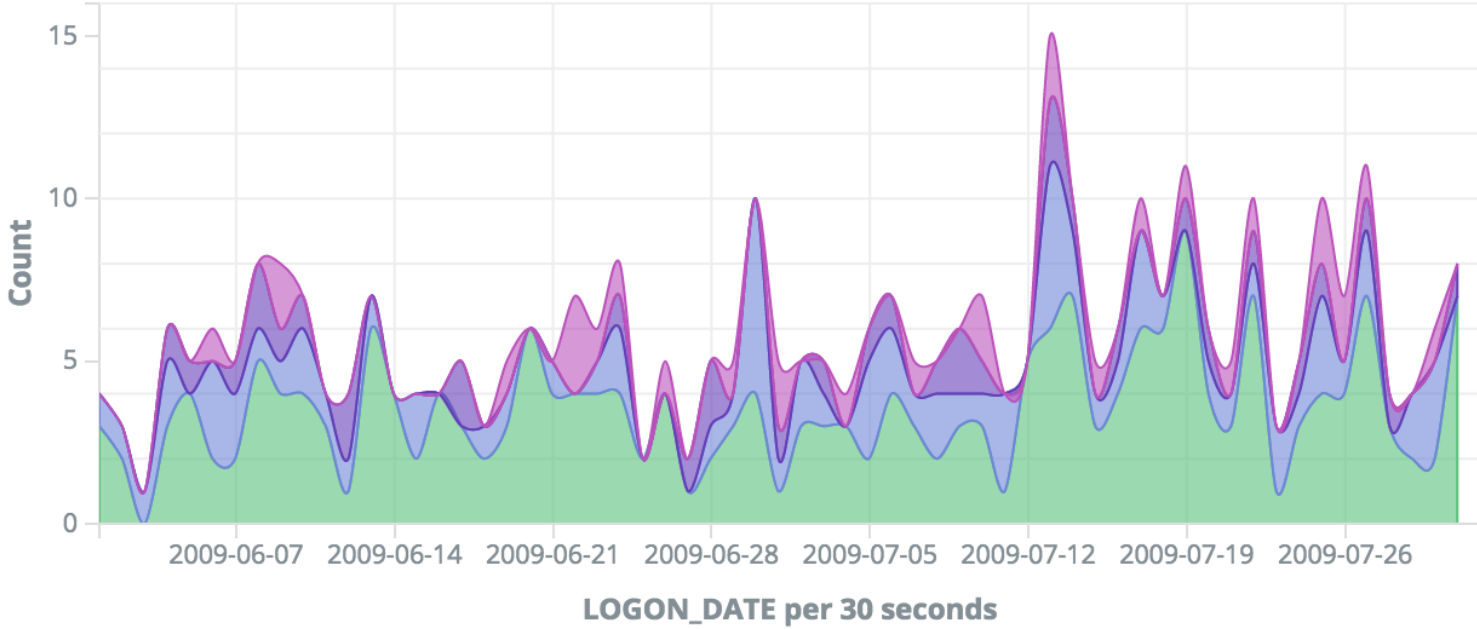
Add a filter +

June 1st 2009, 00:00:00.000 - July 1st 2009, 00:00:00.000 — Auto

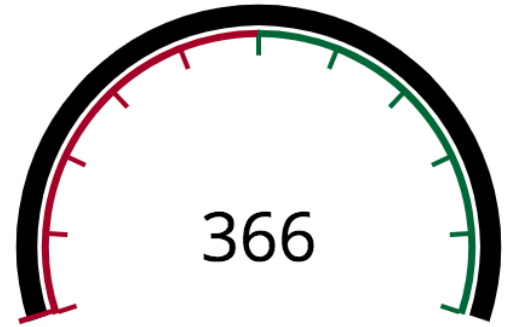


Time	ORDER_STATUS	ORDER_COUNT	MAX_ORDER_TOTAL
June 1st 2009, 12:00:00.000	4	1	3,789
June 1st 2009, 17:00:00.000	4	1	3,731
June 2nd 2009, 10:00:00.000	4	1	4,294

Logons by Customer Class

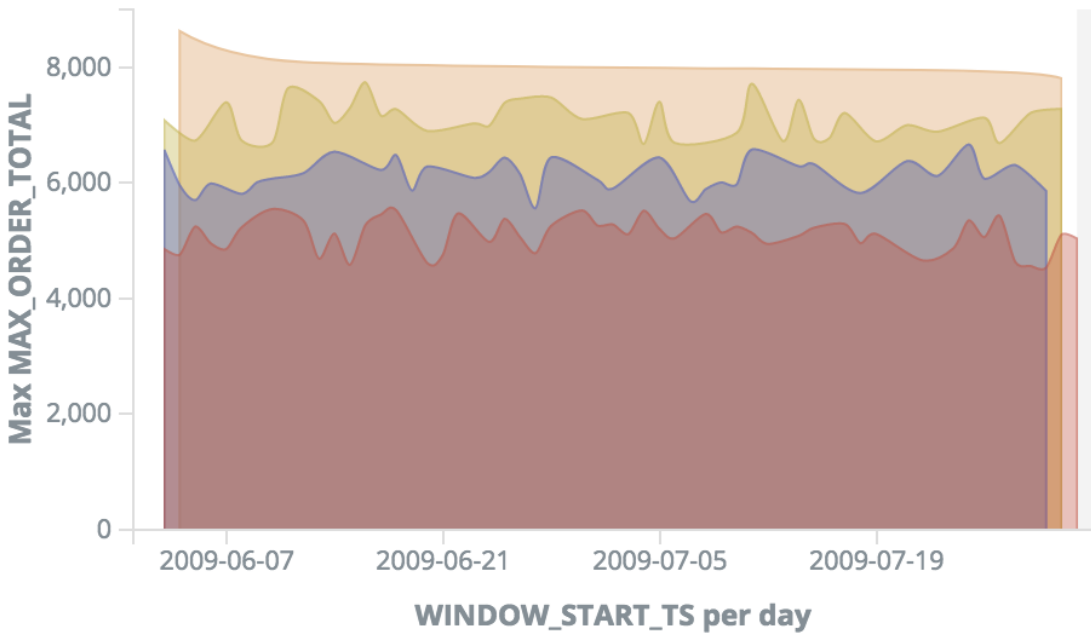


Login count vs target

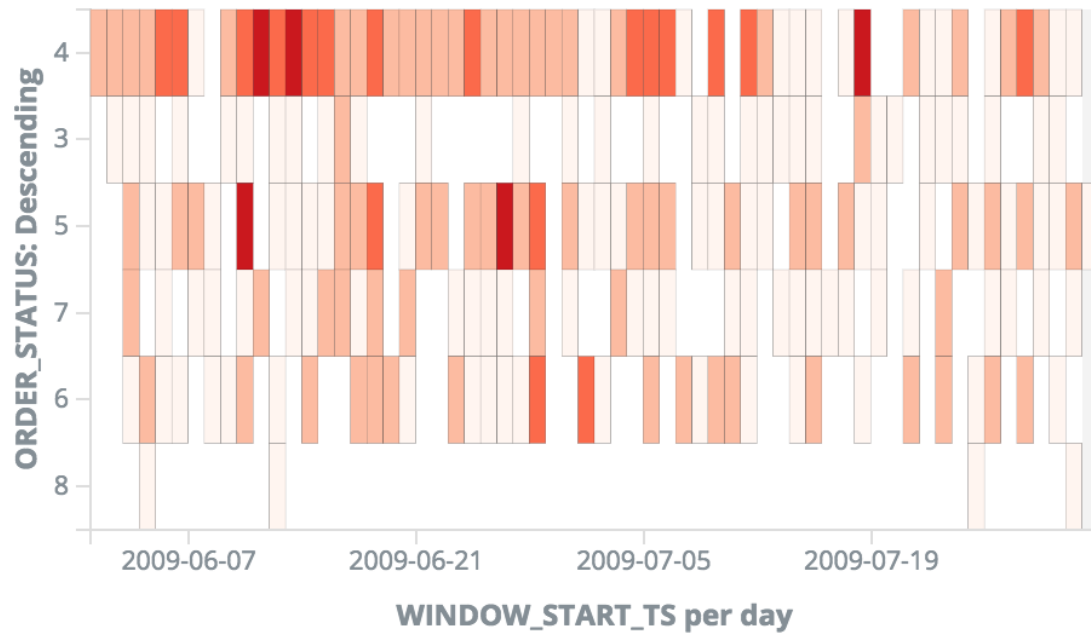


Login count vs target

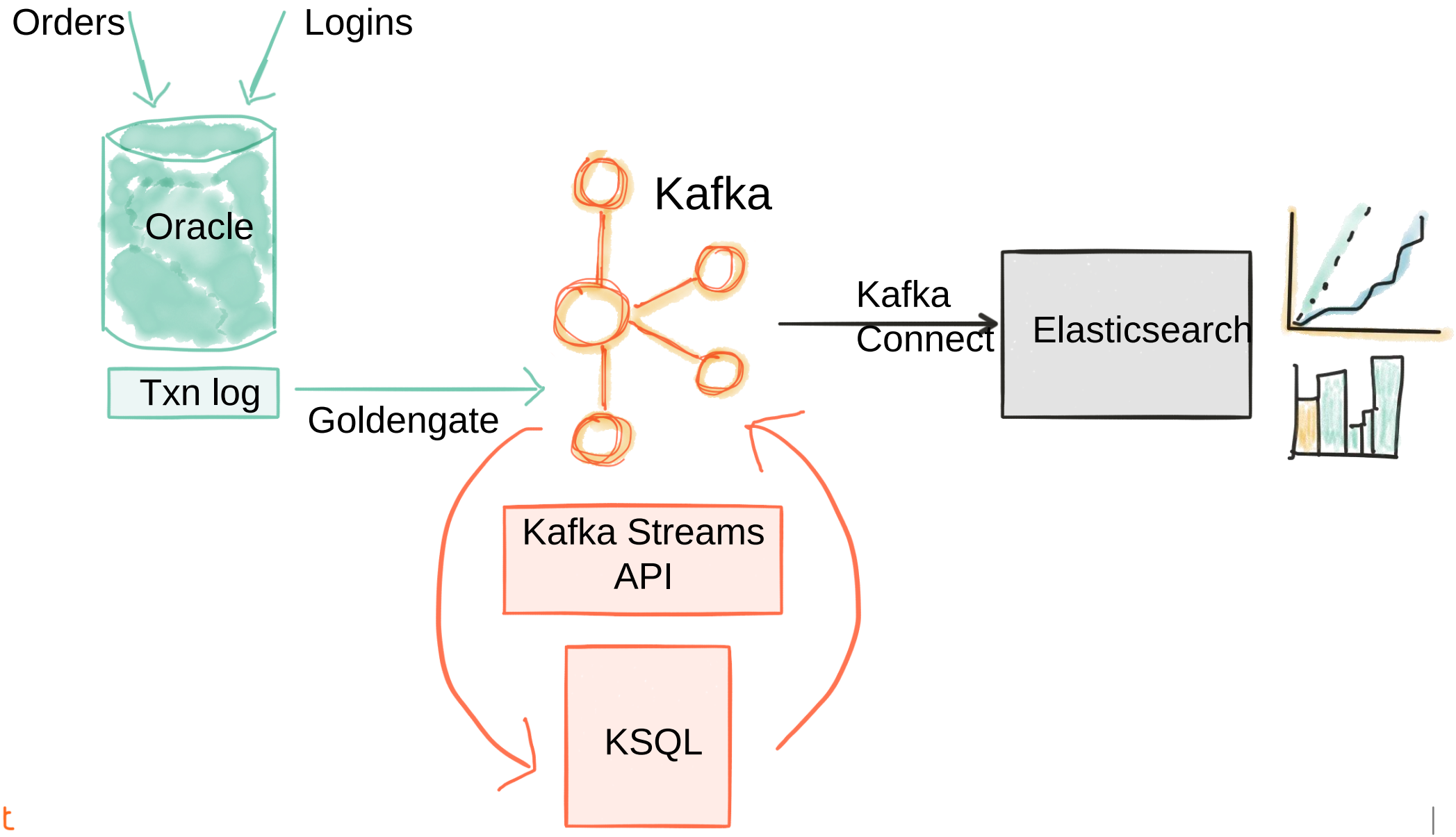
Max Order value by Order Status



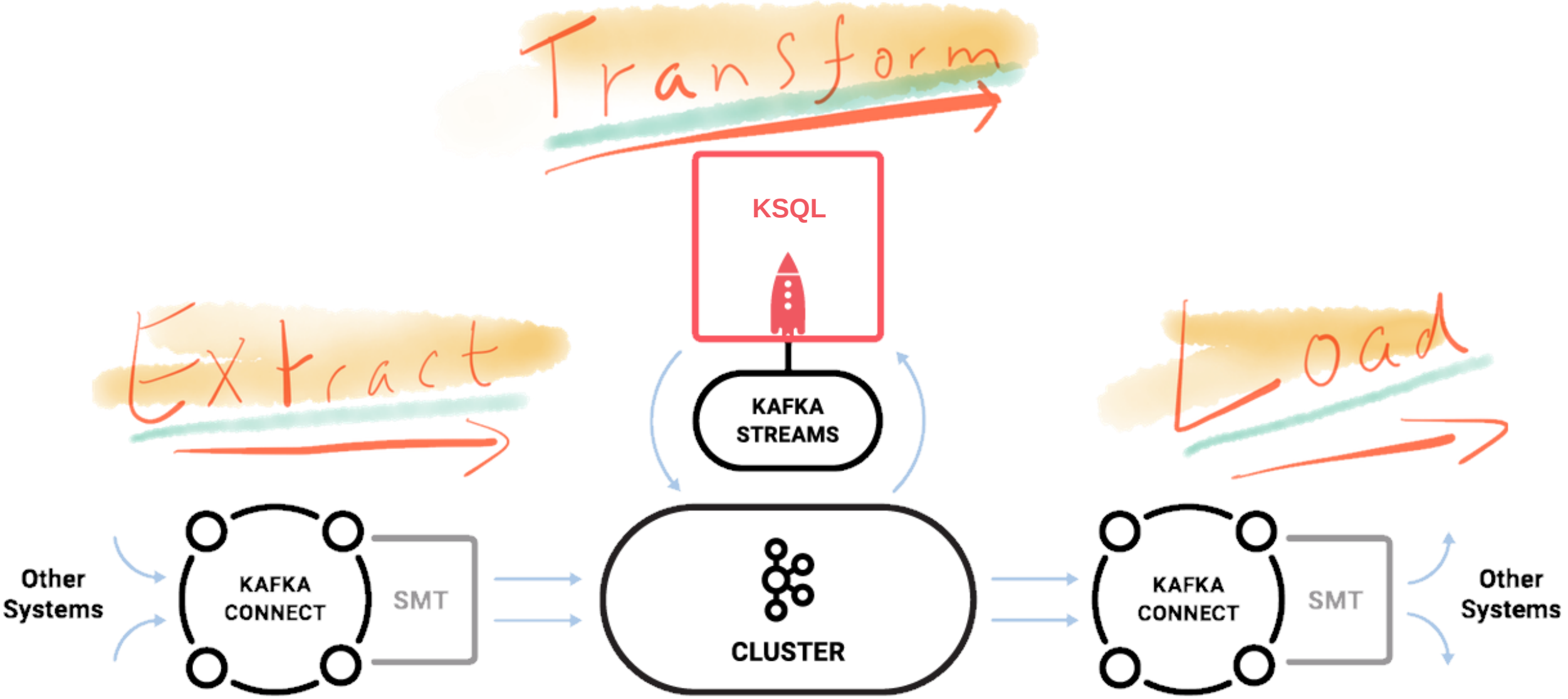
Order Count by Order Status over Time



Realtime Analytics with Kafka and KSQL



Streaming ETL, powered by Apache Kafka and Confluent Platform



Resources and Next Steps

https://github.com/confluentinc/ksql

https://www.confluent.io/download/

https://www.confluent.io/blog

https://slackpass.io/confluentcommunity

Thank you!

@gwenshap
gwen@confluent.io

confluent