@gamussa #jfokus @confluentinc

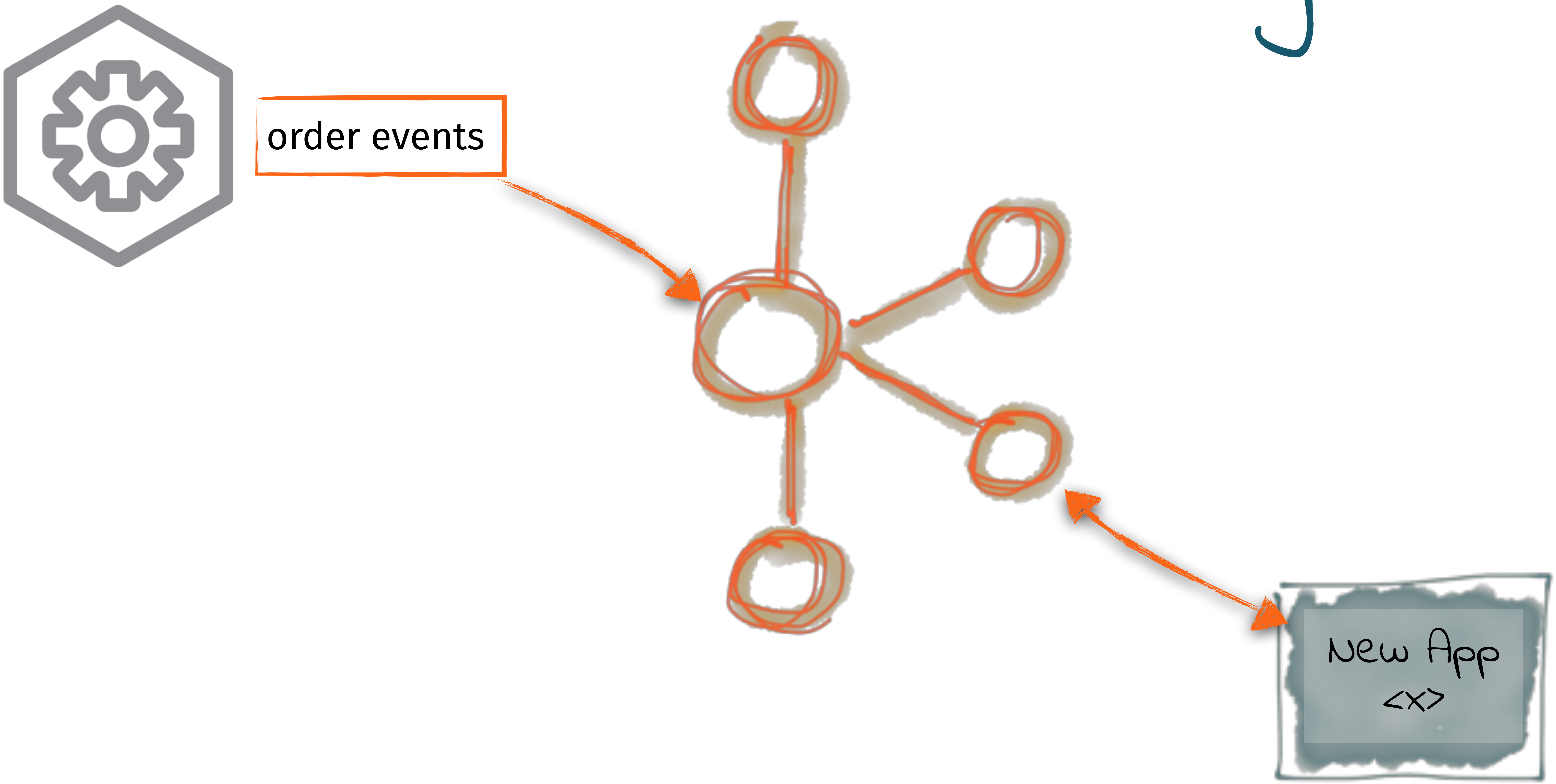
Apache Kafka and KSQL in Action : Let's Build a Streaming Data Pipeline!

APACHE kafka

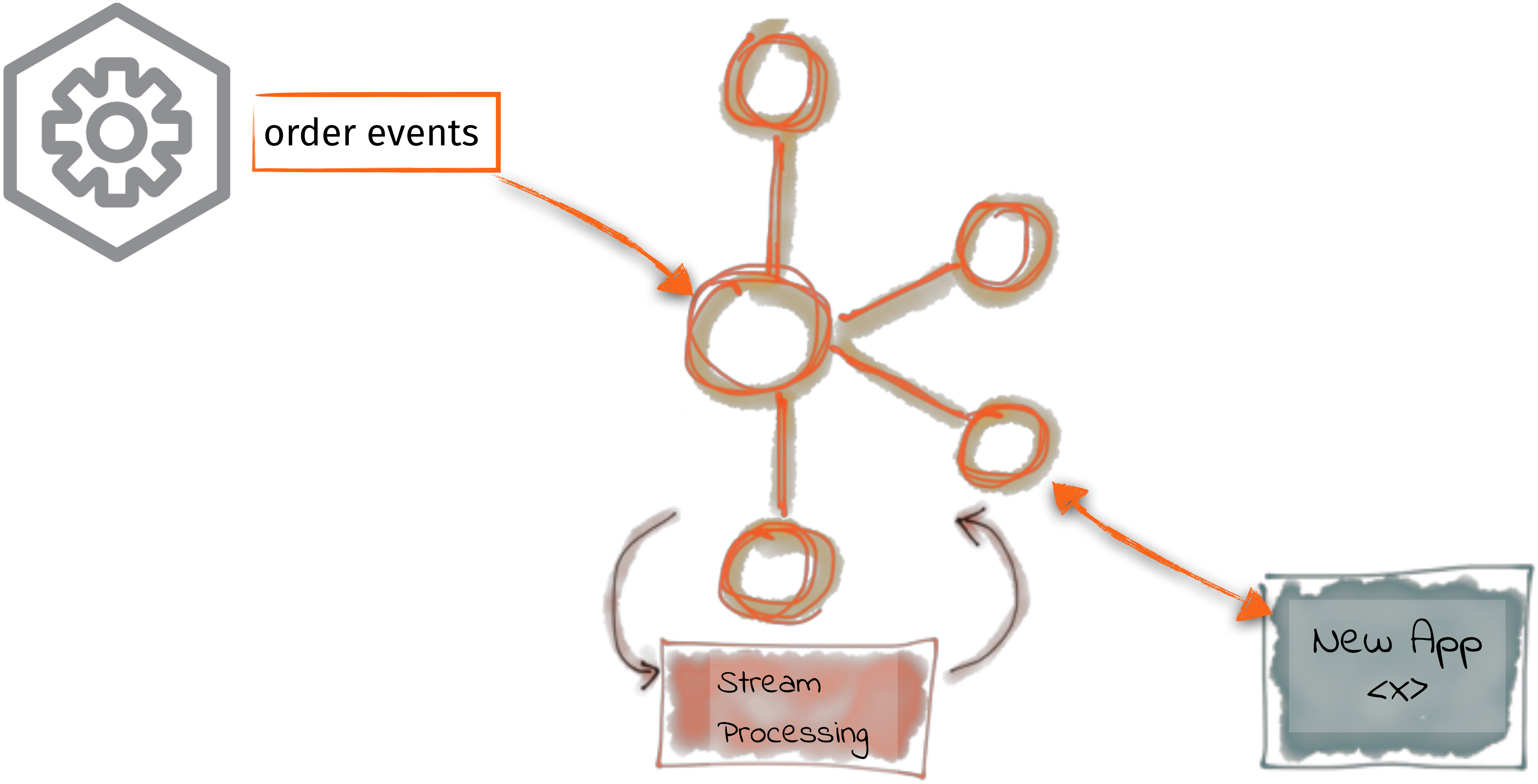
APACHE kafka

APACHE kafka

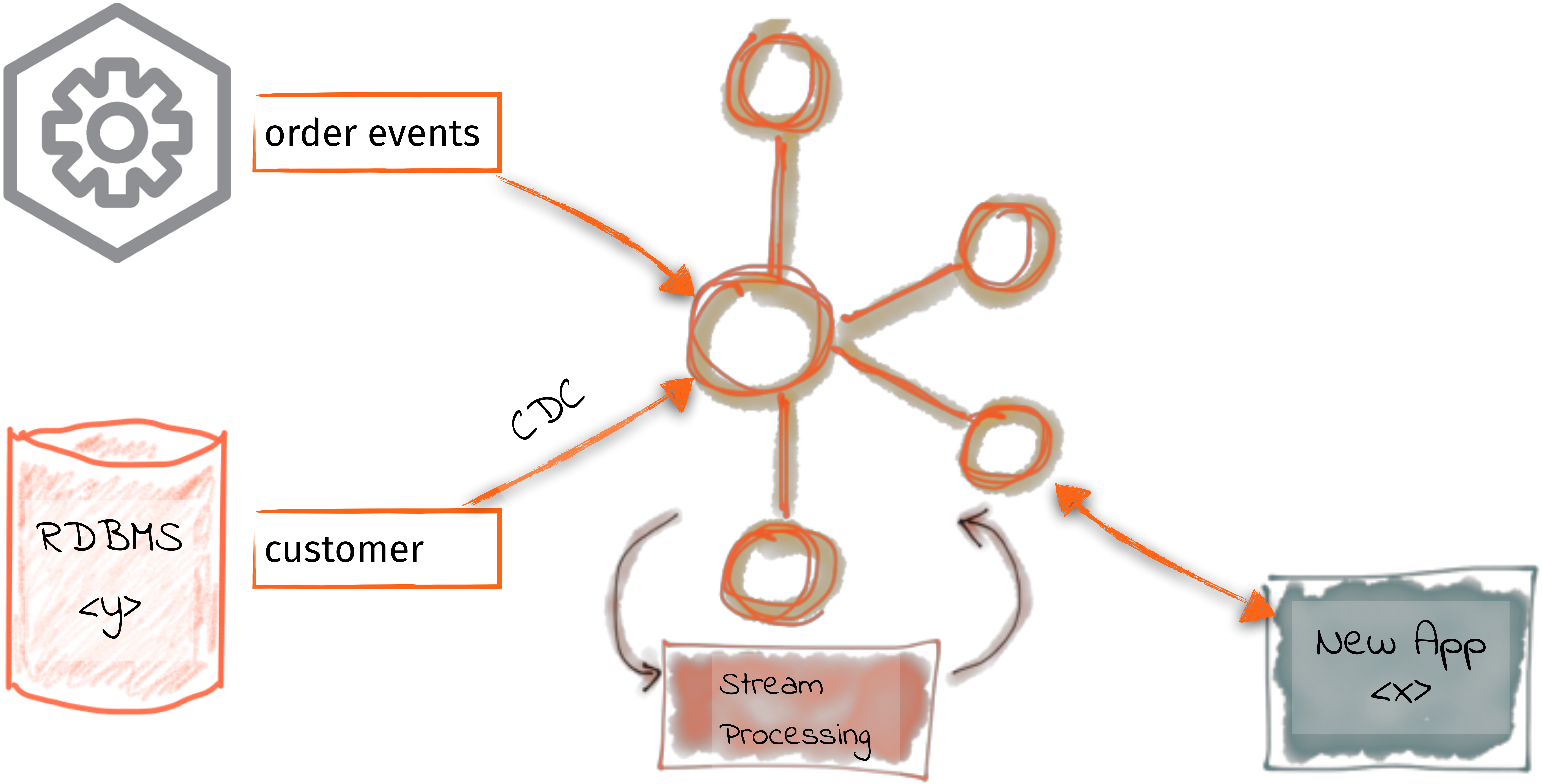
Message Bus



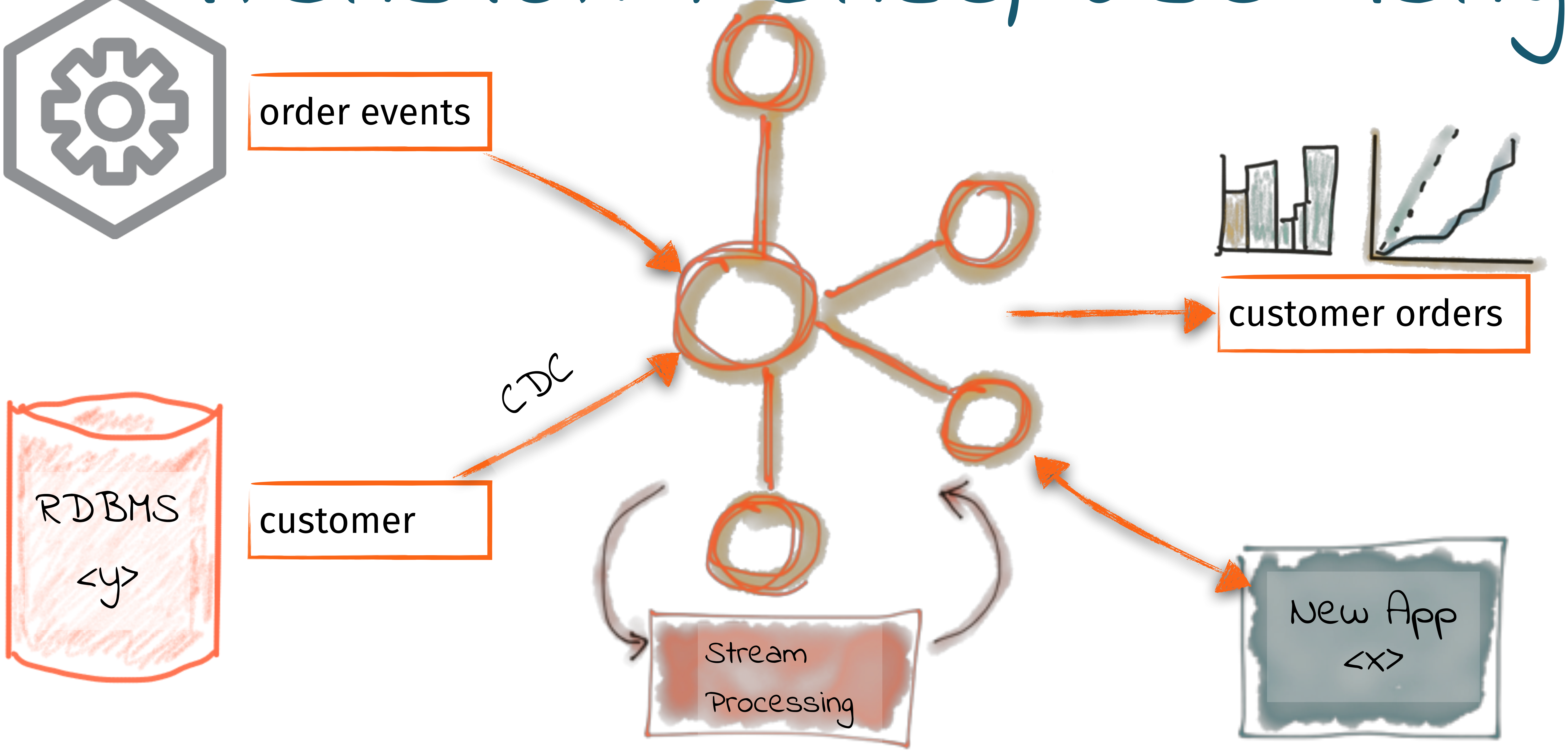
Stream Processing



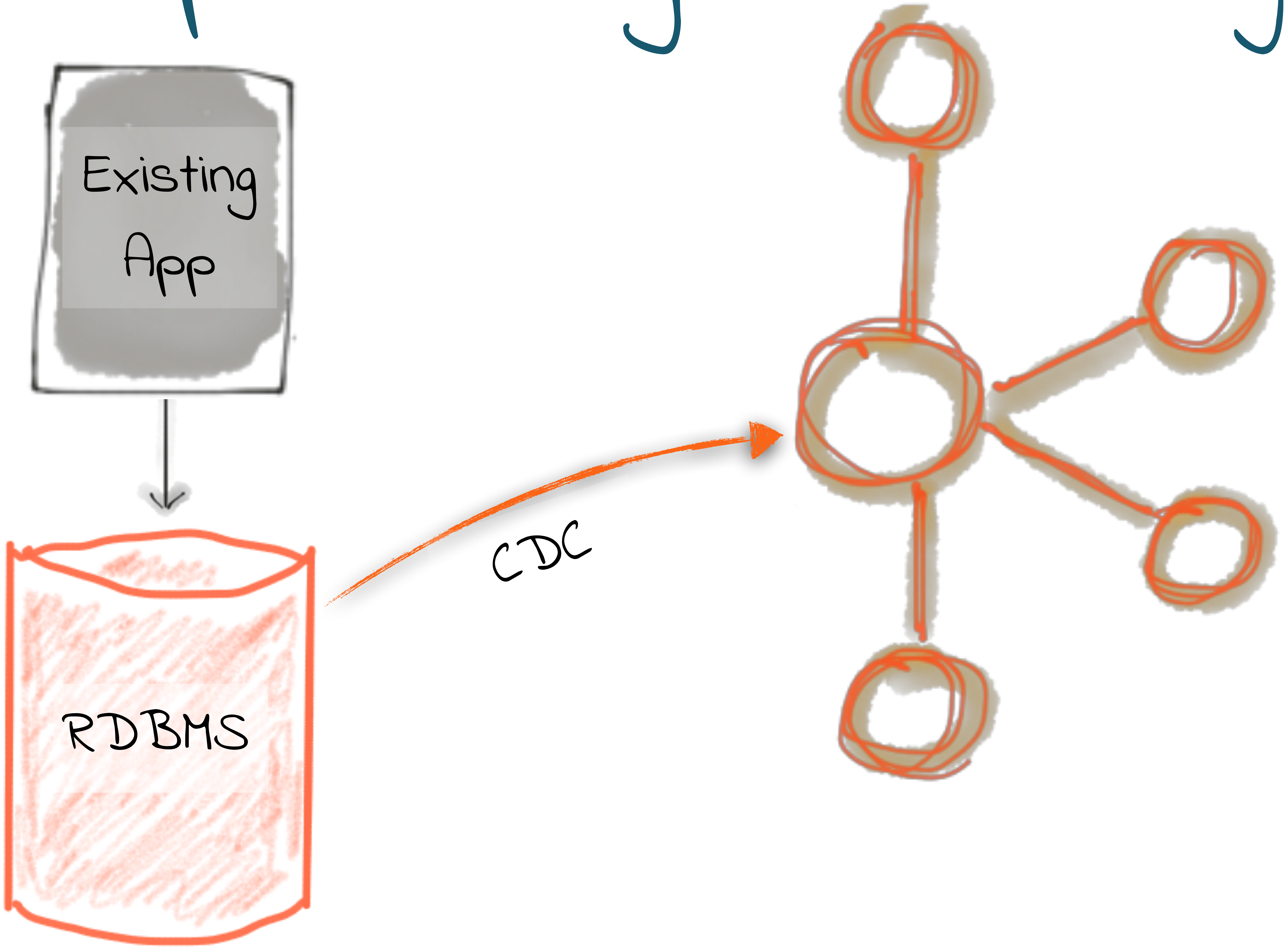
Data Enrichment



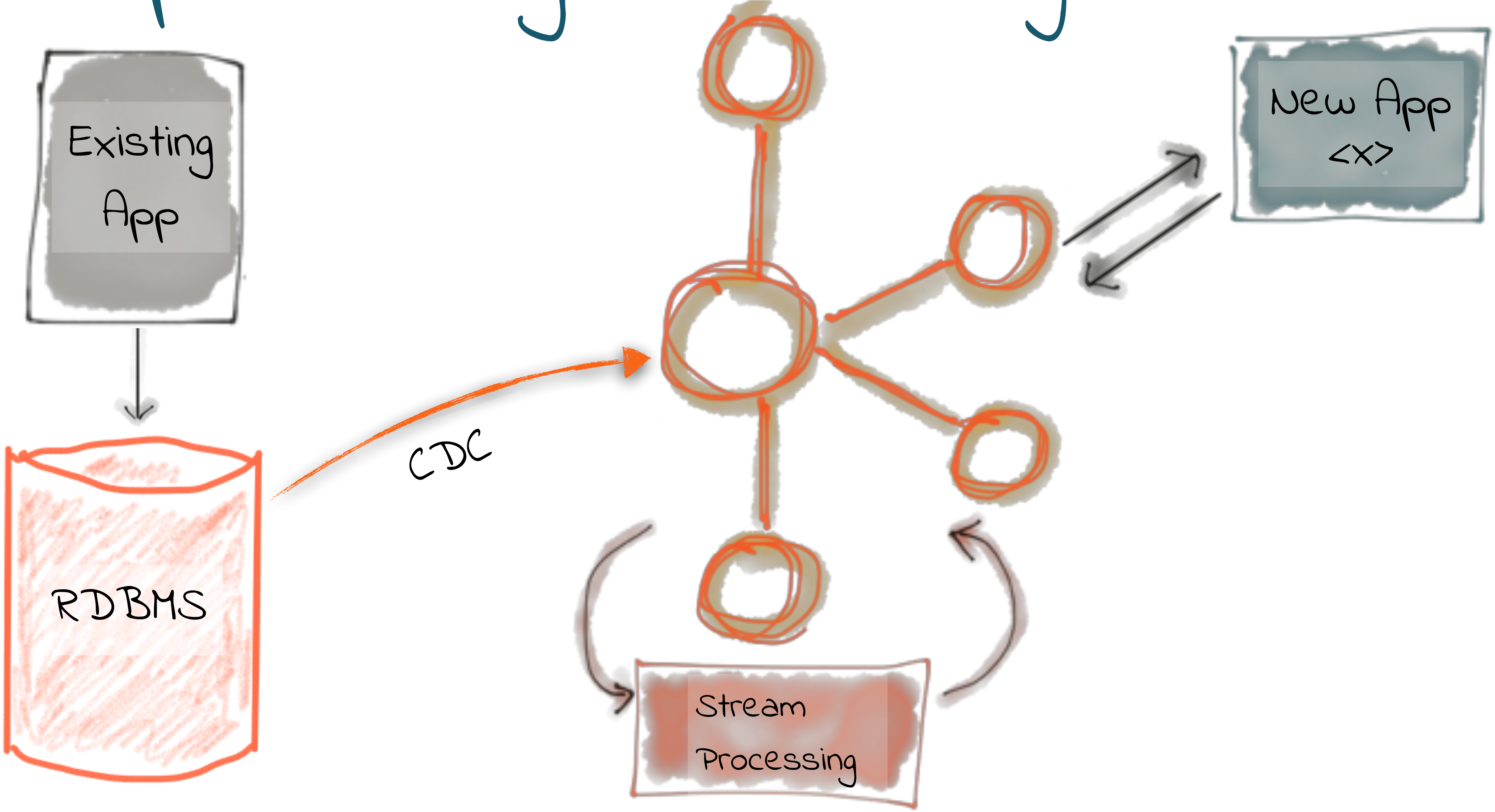
Transform Once, Use Many



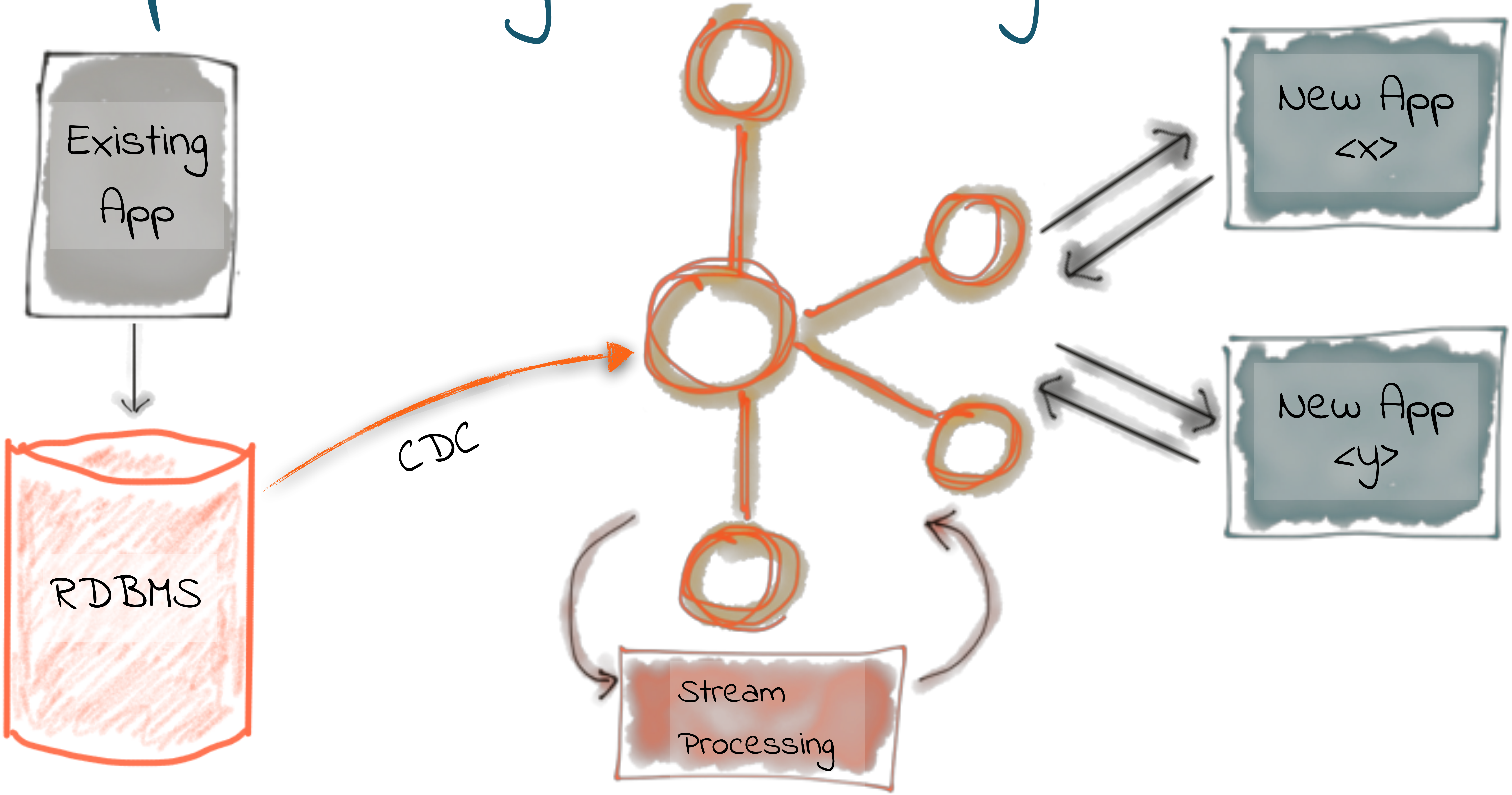
Evolve processing from old systems to new



Evolve processing from old systems to new



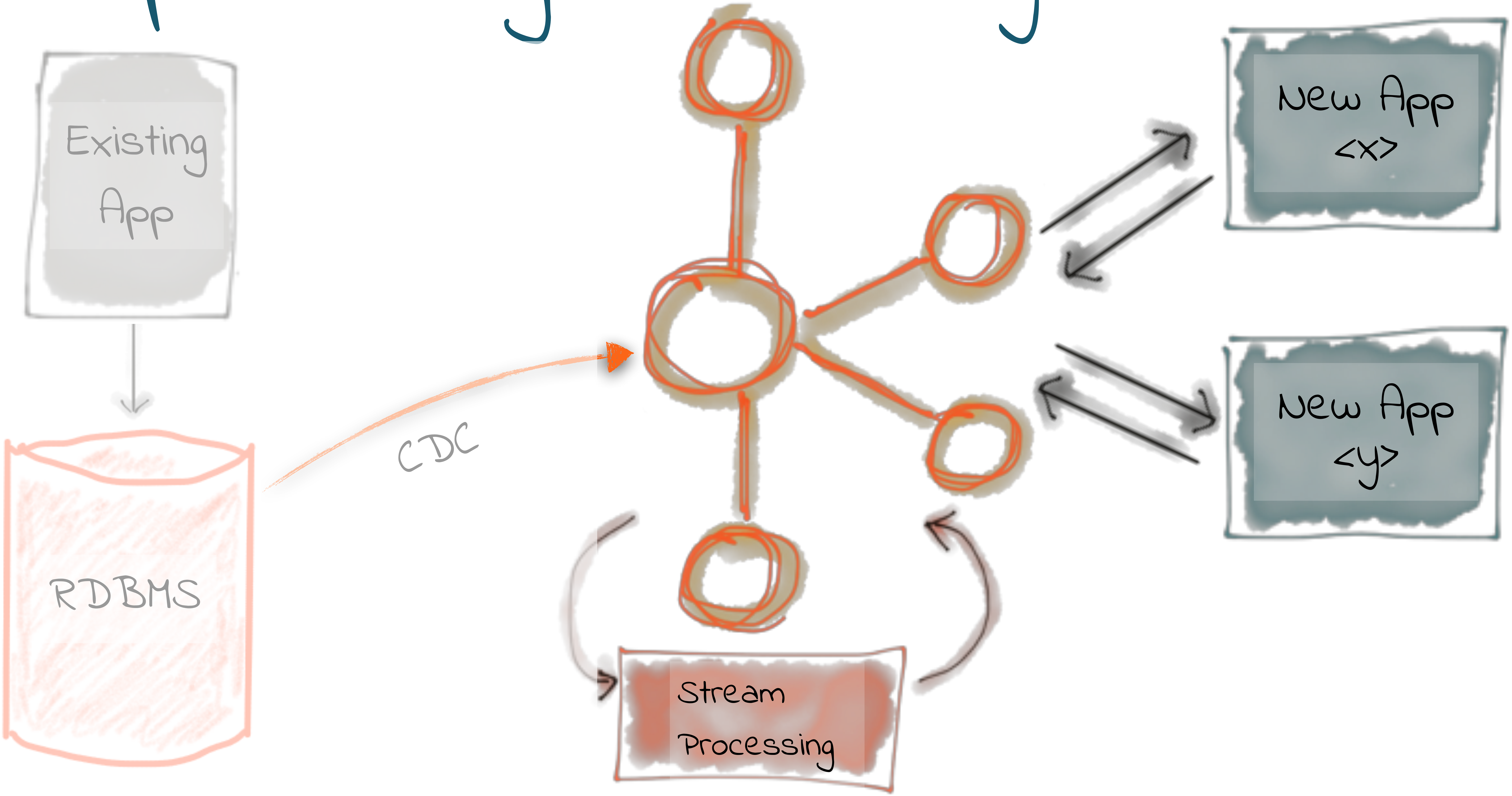
Evolve processing from old systems to new

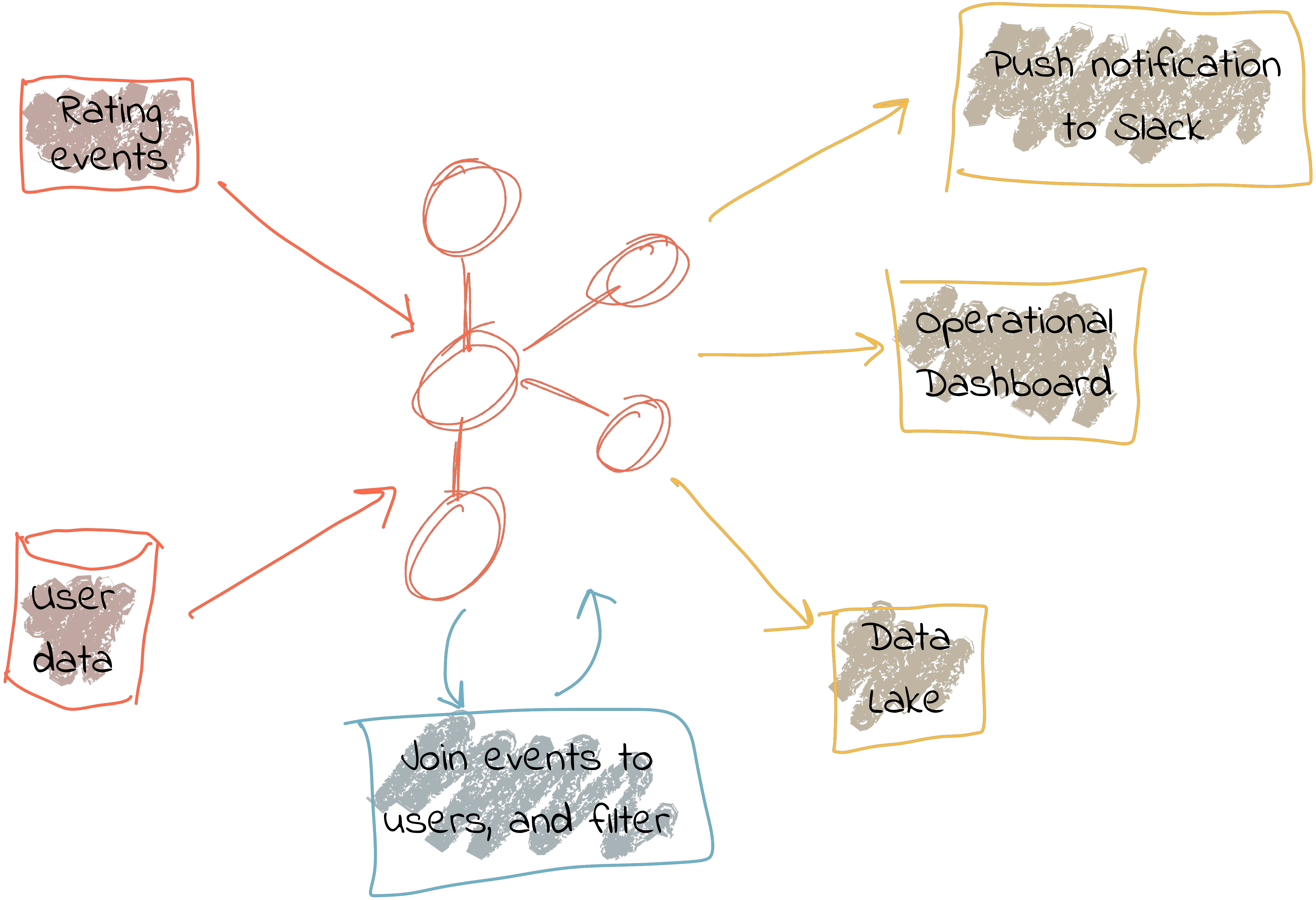


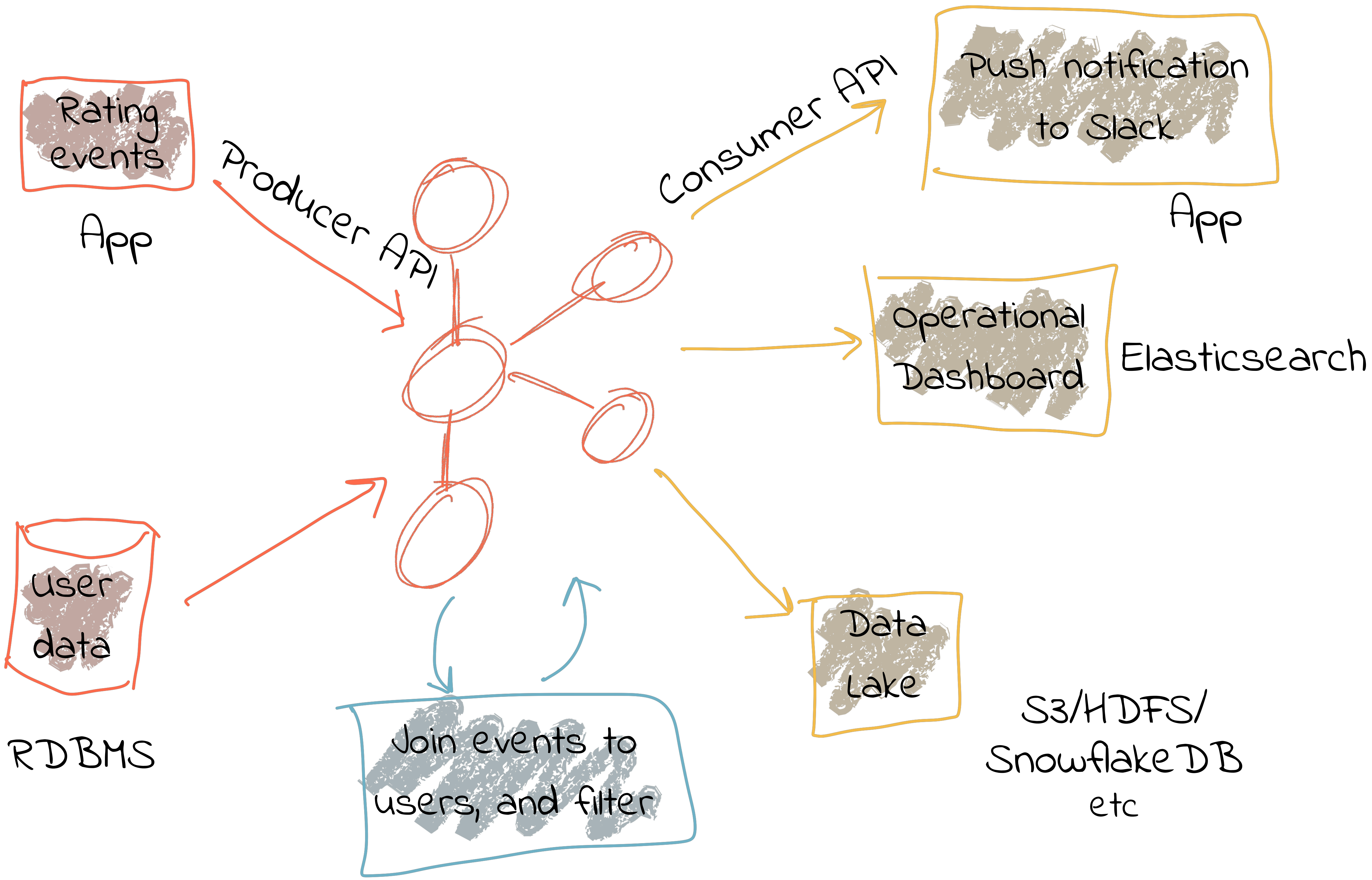
confluent

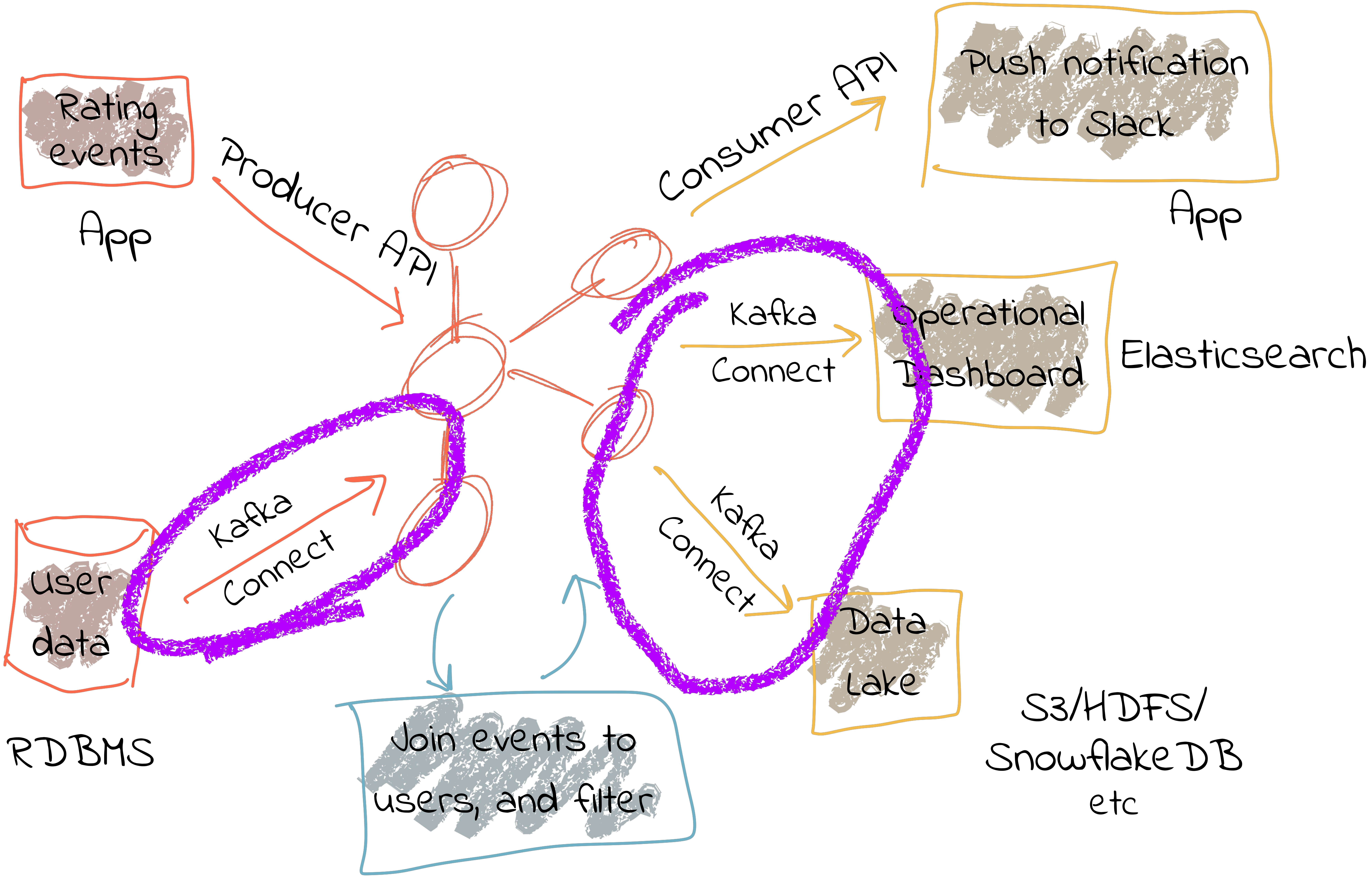
@gamussa #Jfokus @confluentinc

Evolve processing from old systems to new

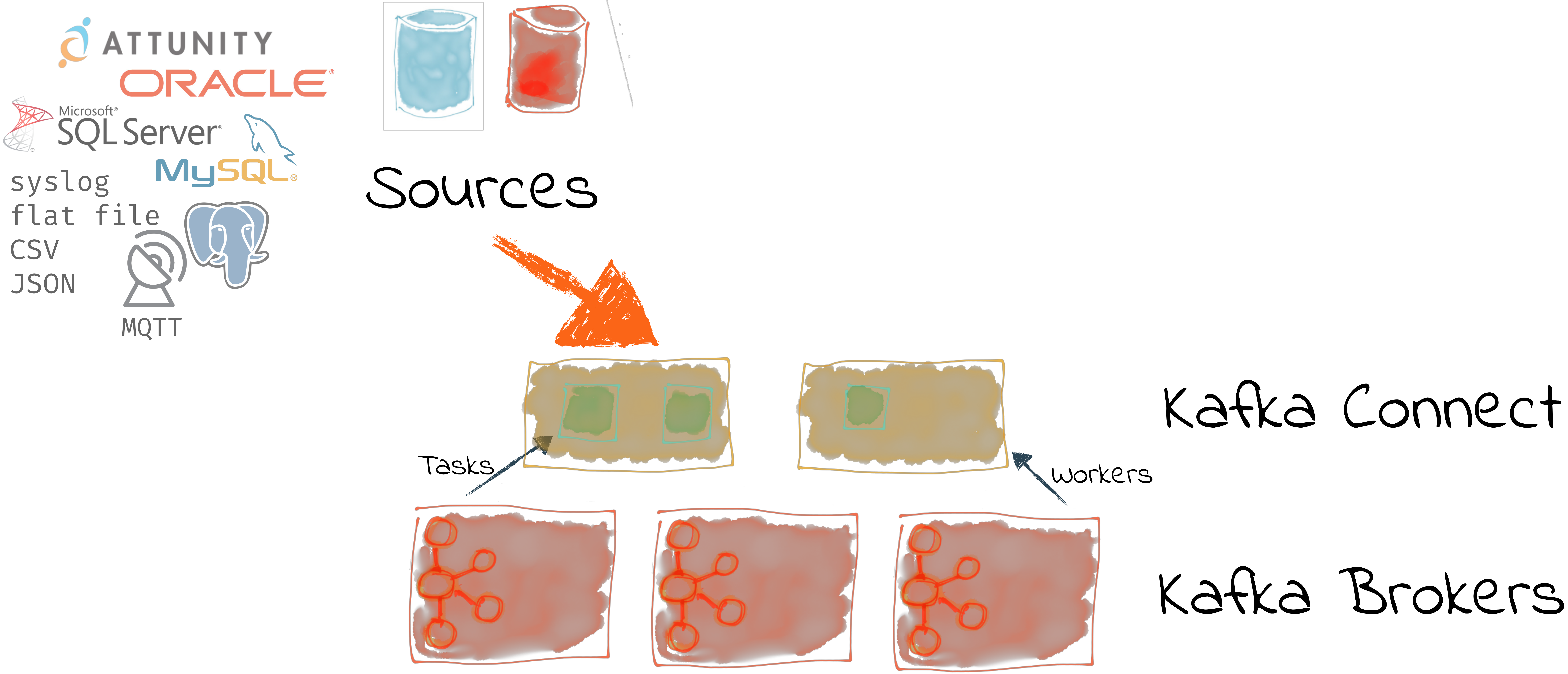



Rating events
User data
Push notification to Slack
Operational Dashboard
Data Lake
Join events to users, and filter

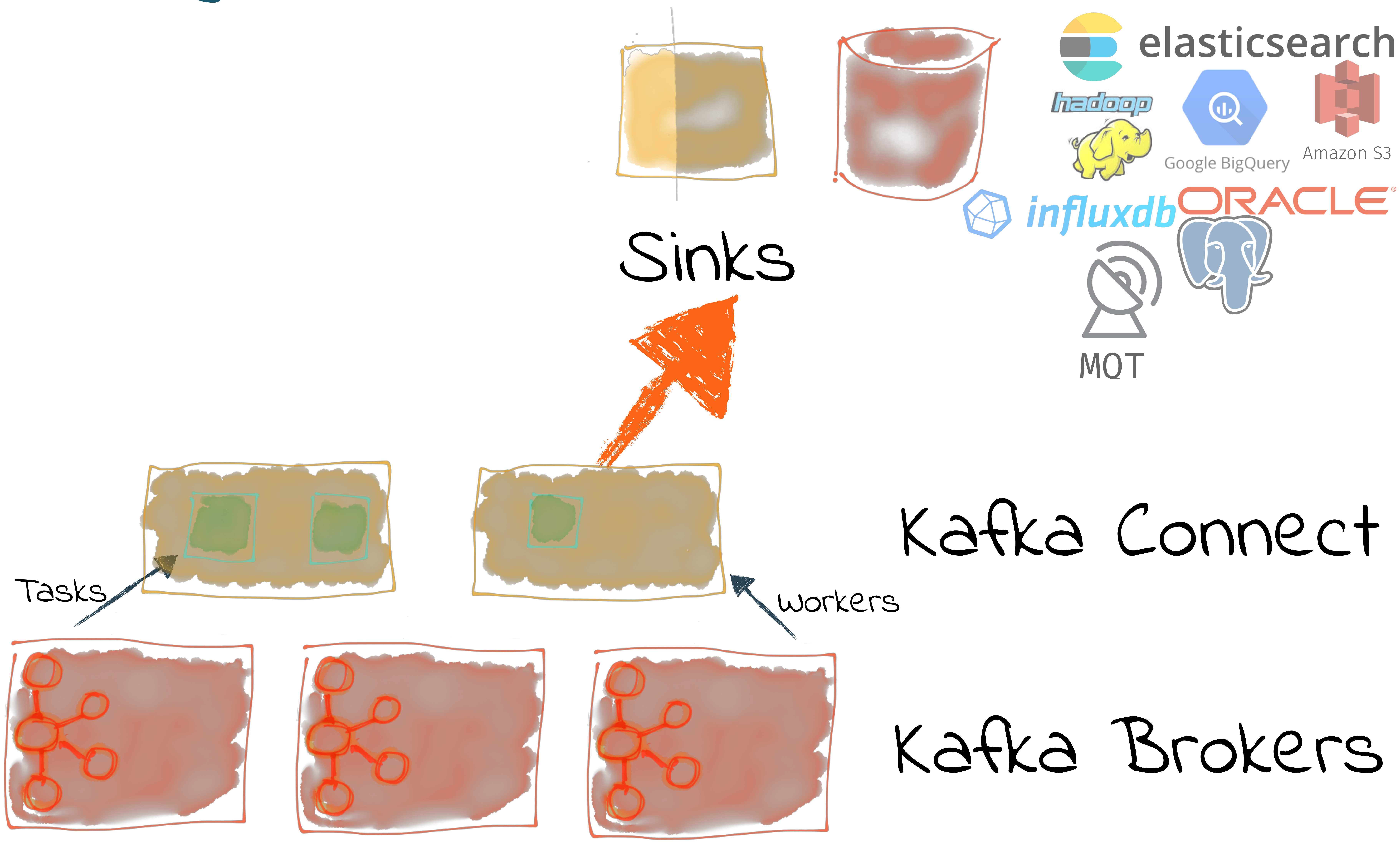

Rating events
App
Producer API
User data
RDBMS
Join events to users, and filter
Consumer API
Push notification to Slack
App
Operational Dashboard
Elasticsearch
Data Lake
S3/HDFS/
SnowflakeDB
etc


Rating events
App
Producer API
Consumer API
Push notification to Slack
App
Kafka Connect
Operational Dashboard
Elasticsearch
Kafka Connect
User data
RDBMS
Kafka Connect
Data Lake
S3/HDFS/ SnowflakeDB etc
Join events to users, and filter

Streaming Integration with Kafka Connect



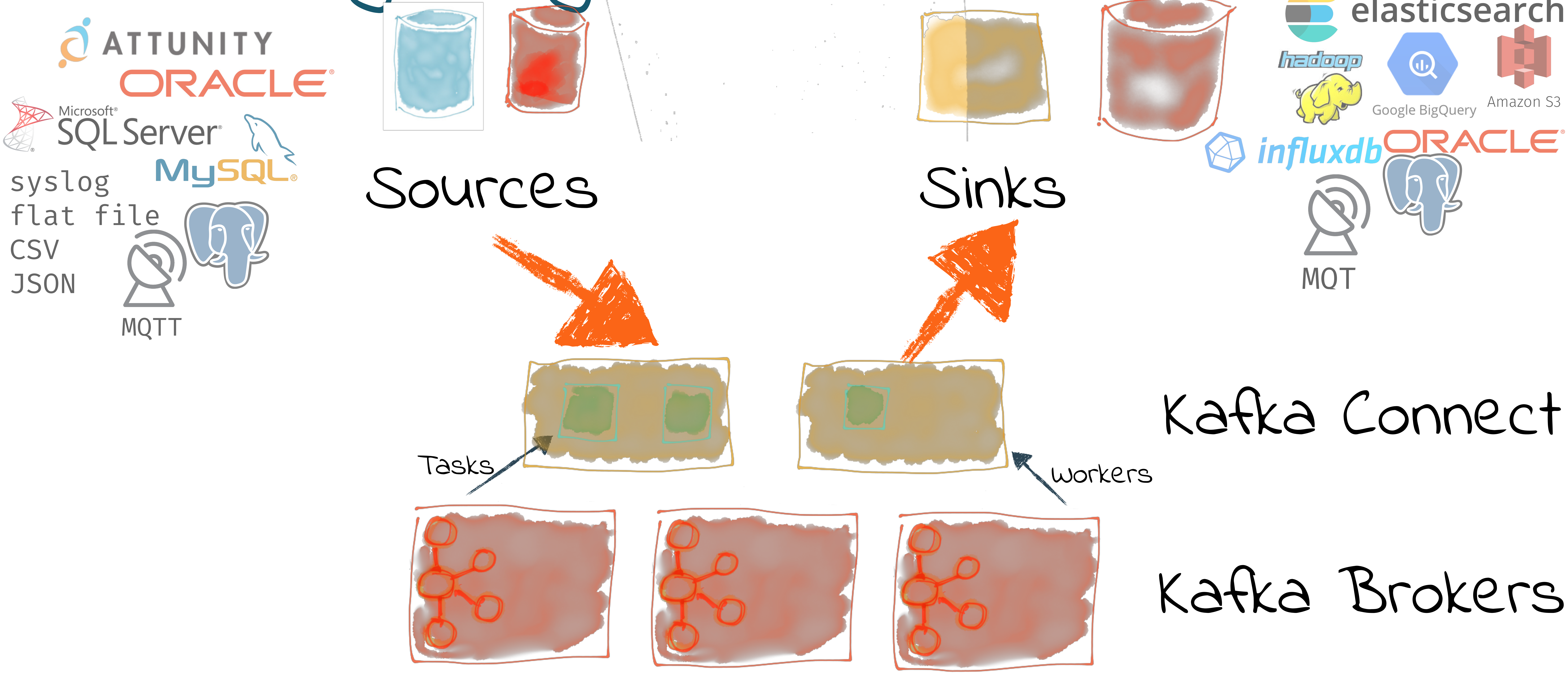
Streaming Integration with Kafka Connect

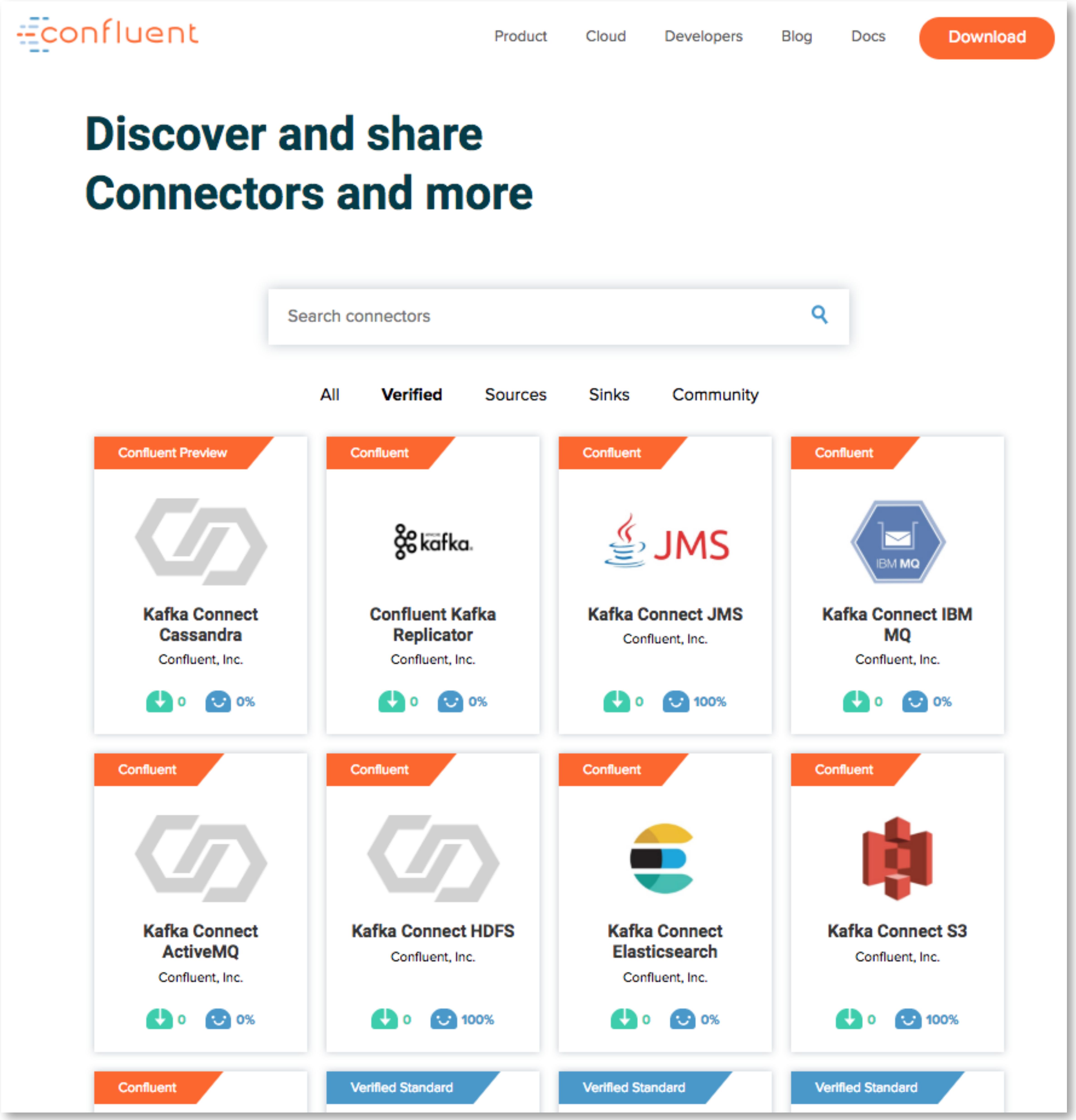


confluent

@gamussa #Jfokus @confluentinc

Streaming Integration with Kafka Connect





Confluent Hub

- one-stop place to discover and download :
  - Connectors
  - Transformations
  - Converters

hub.confluent.io

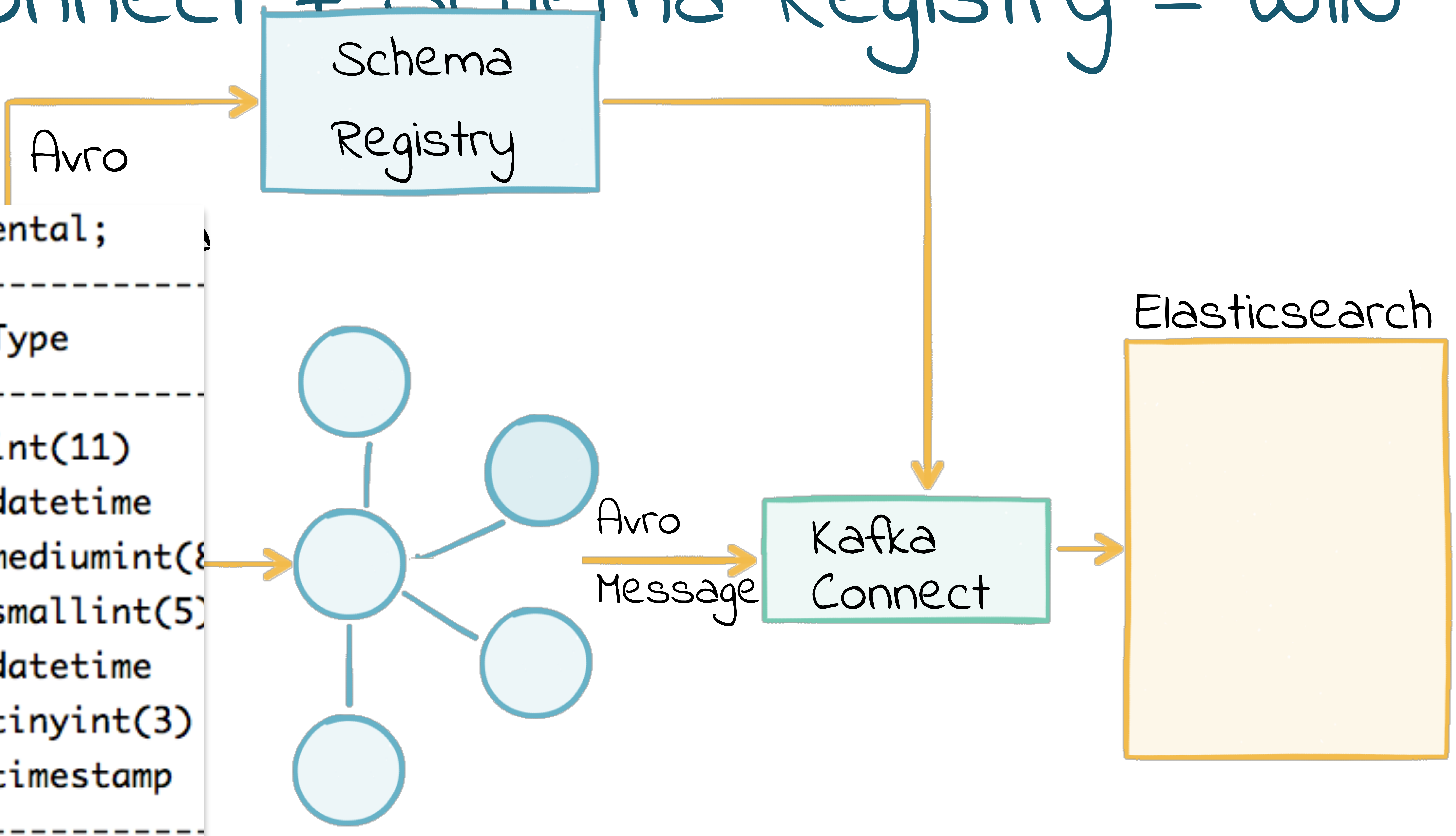
Kafka Connect + Schema Registry = WIN

confluent

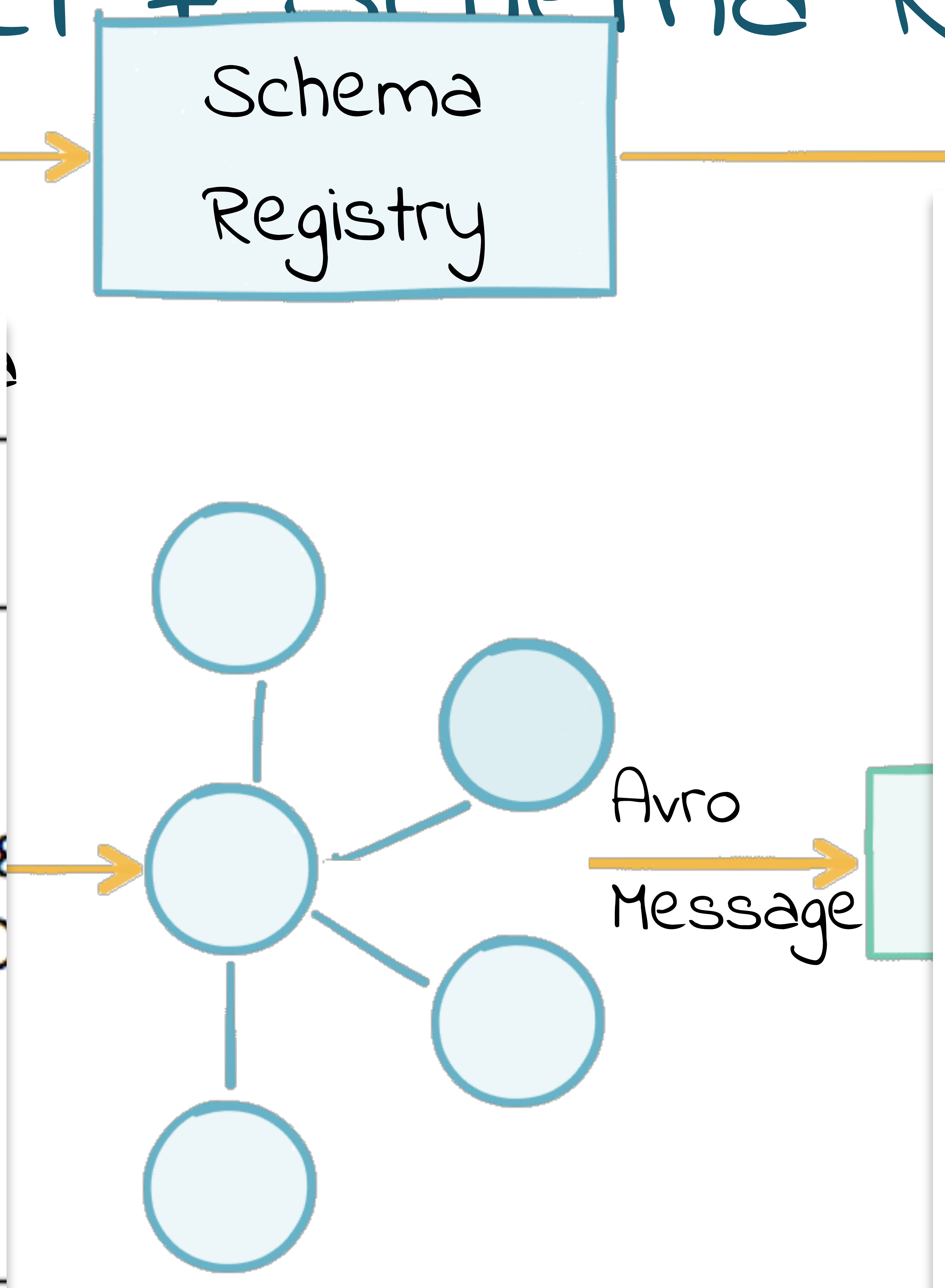
@gamussa #Jfokus @confluentinc

Kafka Connect + Schema Registry = WIN

```
mysql> describe rental;
+---------------+------------
| Field         | Type
+---------------+------------
| rental_id     | int(11)
| rental_date   | datetime
| inventory_id  | mediumint(8
| customer_id   | smallint(5)
| return_date   | datetime
| staff_id      | tinyint(3)
| last_update   | timestamp
+---------------+------------
```



Kafka Connect + Schema Registry = WIN




```
mysql> describe rental;
+--------------+------------
| Field        | Type
+--------------+------------
| rental_id    | int(11)
| rental_date  | datetime
| inventory_id | mediumint(8
| customer_id  | smallint(5)
| return_date  | datetime
| staff_id     | tinyint(3)
| last_update  | timestamp
+--------------+------------
```

```
"mappings": {
  "type.name=kafka-connect": {
    "dynamic_templates": [ ],
    "properties": {
      "customer_id": {
        "type": "integer"
      },
      "inventory_id": {
        "type": "long"
      },
      "last_update": {
        "type": "date"
      },
      "rental_date": {
        "type": "date"
      },
      "rental_id": {
        "type": "integer"
      },
      "return_date": {
        "type": "date"
      },
```

@gar

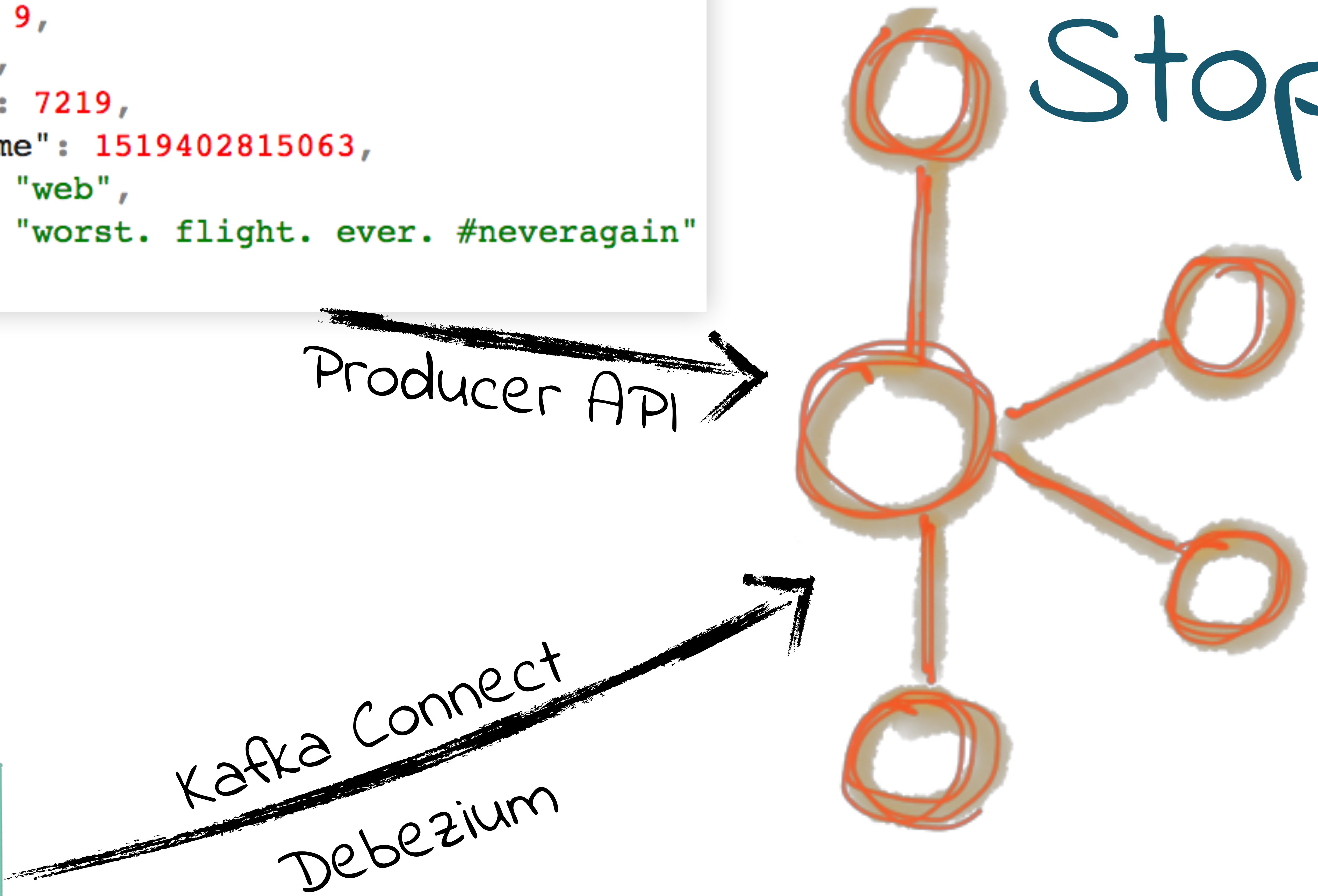
confluent

Kafka → Elasticsearch

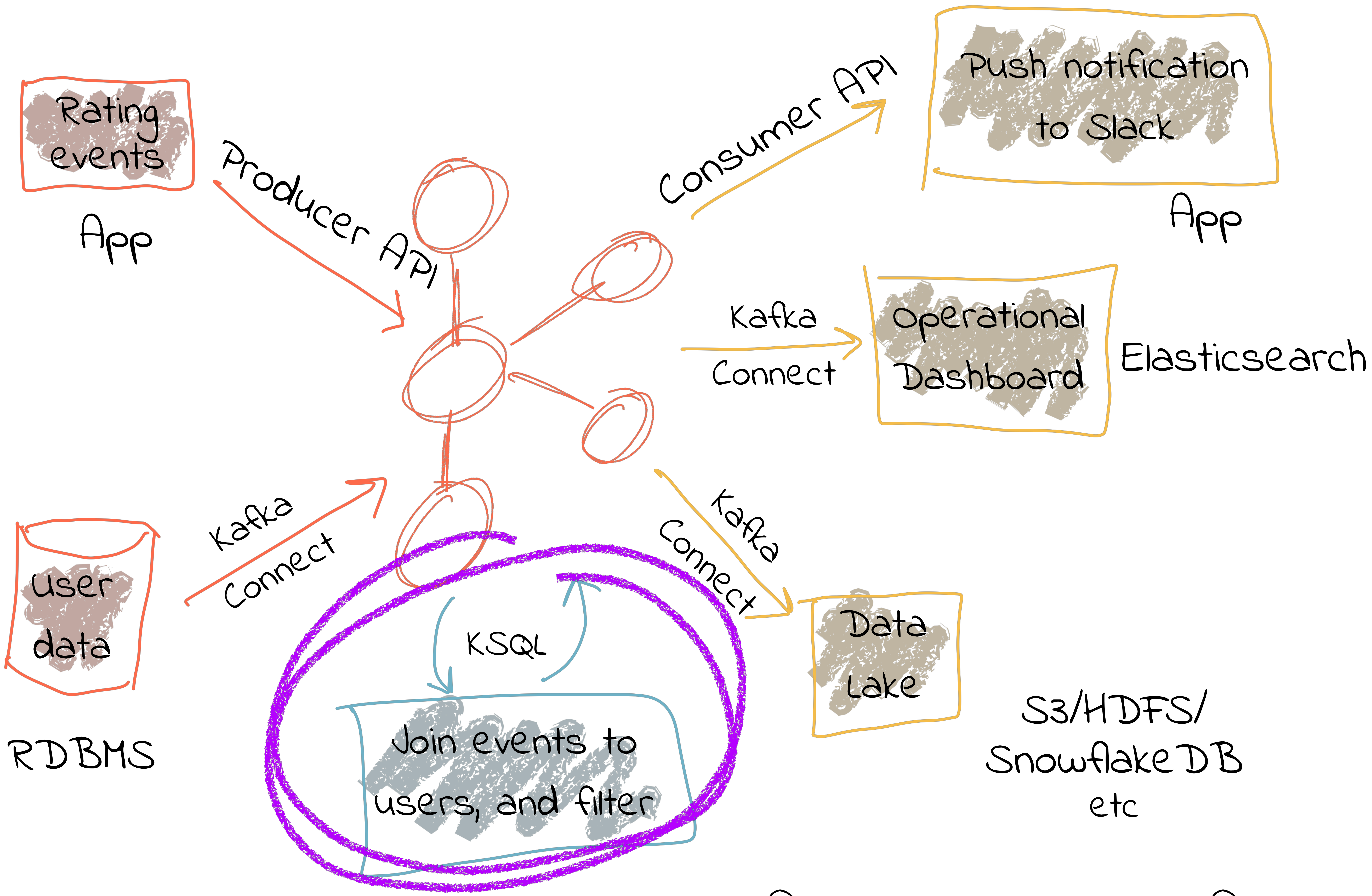
```
curl -X "POST" "http://kafka-connect-cp:18083/connectors/" \
     -H "Content-Type: application/json" \
     -d '{
    "name": "es_sink_jfokus19",
    "config": {
    "connector.class": "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter",
    "value.converter.schemas.enable": false,
    "topics": "lisa18",
    "key.ignore": "true",
    "schema.ignore": "true",
    "type.name": "type.name=kafkaconnect",
    "connection.url": "http://elasticsearch:9200"
  }
}'
```

Stop! Demo time!

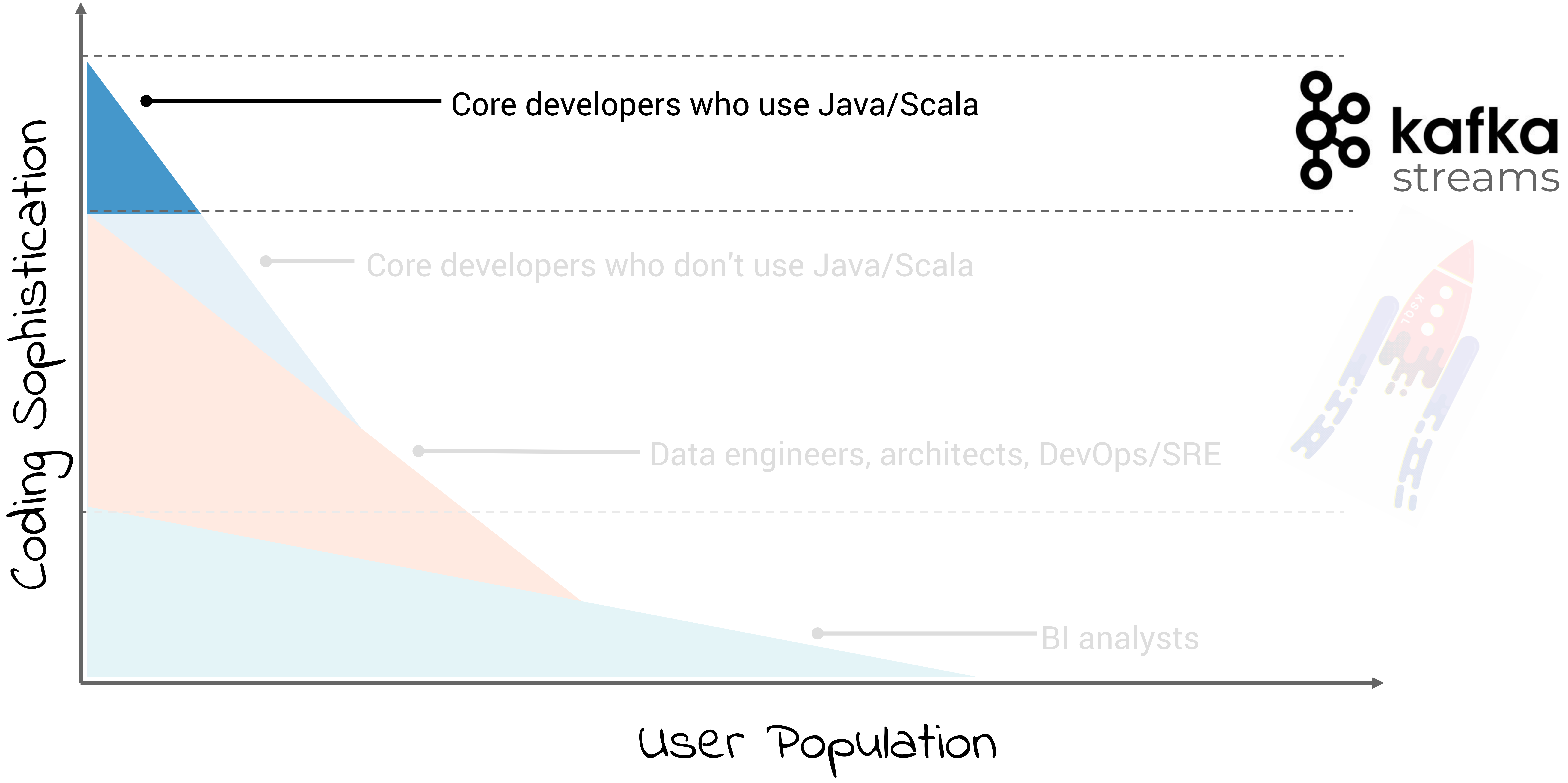

```
{
  "rating_id": 2133,
  "user_id": 9,
  "stars": 1,
  "route_id": 7219,
  "rating_time": 1519402815063,
  "channel": "web",
  "message": "worst. flight. ever. #neveragain"
}
```

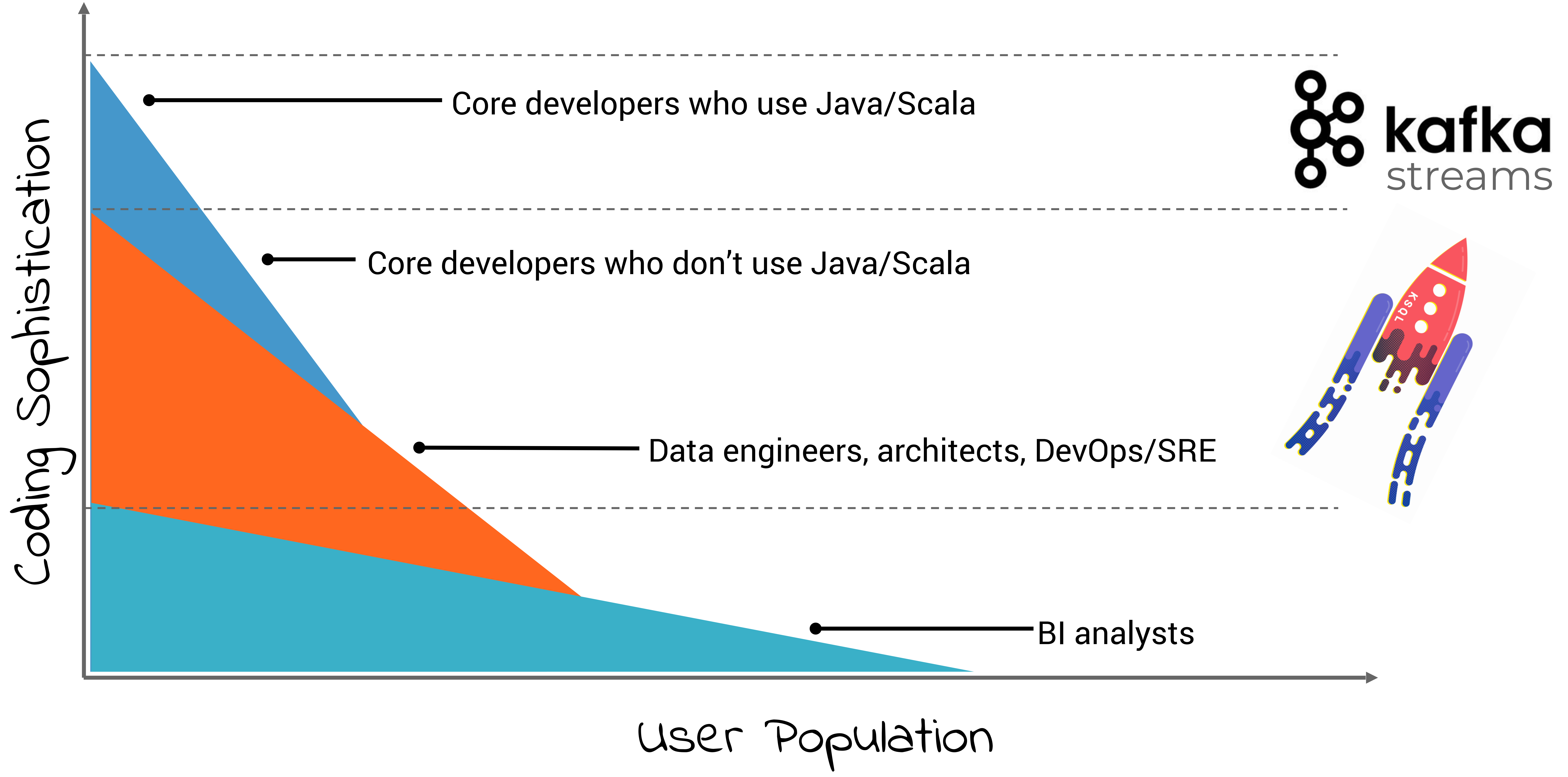

```
{
  "uid": 9,
  "name": "Neha",
  "locale": "en_US",
  "address_city": "Palo Alto",
  "elite": "P"
}
```


Rating events
App
Producer API
Consumer API
Push notification to Slack
App
Kafka Connect
Operational Dashboard
Elasticsearch
Kafka Connect
User data
RDBMS
KSQL
Join events to users, and filter
Kafka Connect
Data Lake
S3/HDFS/
SnowflakeDB
etc

Lower the bar to enter the world of streaming

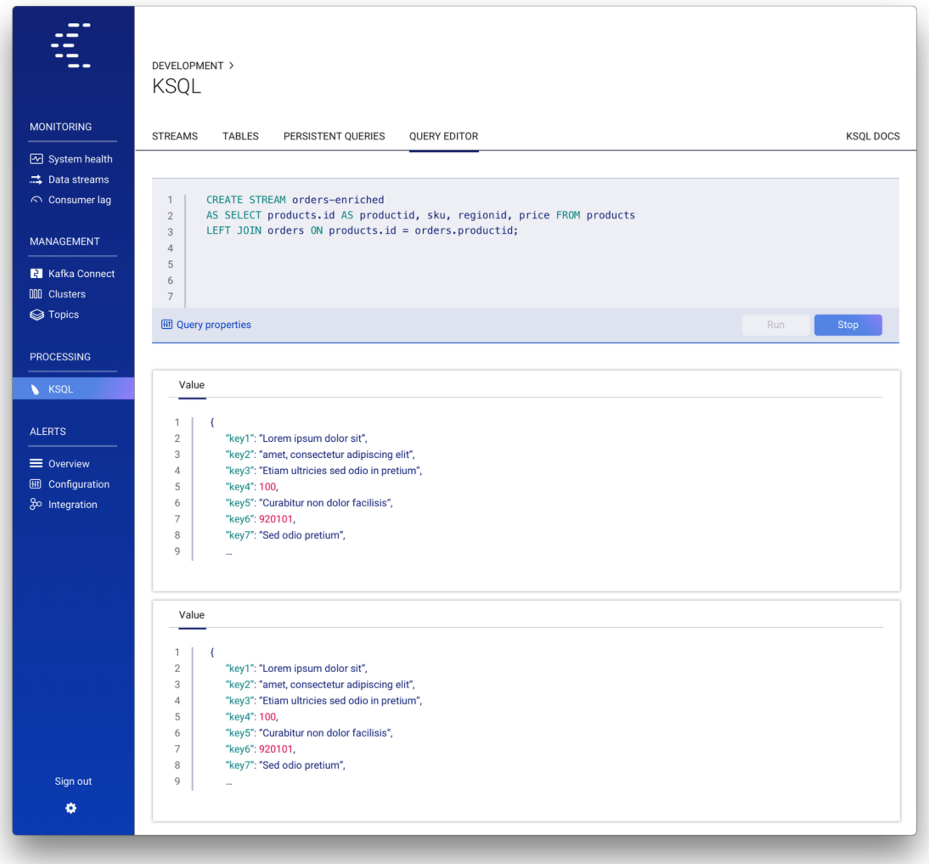


Lower the bar to enter the world of streaming



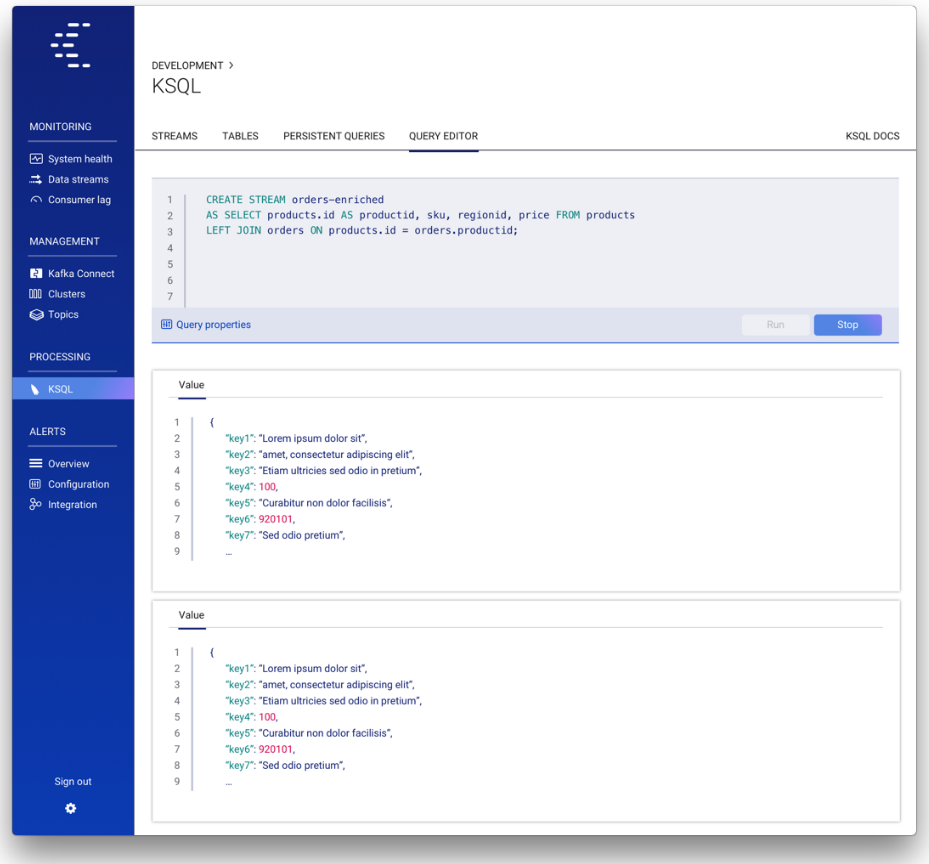
KSQL #FTW

KSQL #FTW


DEVELOPMENT >
KSQL
STREAMS
TABLES
PERSISTENT QUERIES
QUERY EDITOR
KSQL DOCS
MONITORING
System health
Data streams
Consumer lag
MANAGEMENT
Kafka Connect
Clusters
Topics
PROCESSING
KSQL
ALERTS
Overview
Configuration
Integration
CREATE STREAM orders-enriched
AS SELECT products.id AS productid, sku, regionid, price FROM products
LEFT JOIN orders ON products.id = orders.productid;
Query properties
Run
Stop
Value
{
"key1": "Lorem ipsum dolor sit",
"key2": "amet, consectetur adipiscing elit",
"key3": "Etiam ultricies sed odio in pretium",
"key4": 100,
"key5": "Curabitur non dolor facilisis",
"key6": 920101,
"key7": "Sed odio pretium",
...
Value
{
"key1": "Lorem ipsum dolor sit",
"key2": "amet, consectetur adipiscing elit",
"key3": "Etiam ultricies sed odio in pretium",
"key4": 100,
"key5": "Curabitur non dolor facilisis",
"key6": 920101,
"key7": "Sed odio pretium",
...
Sign out

1 UI

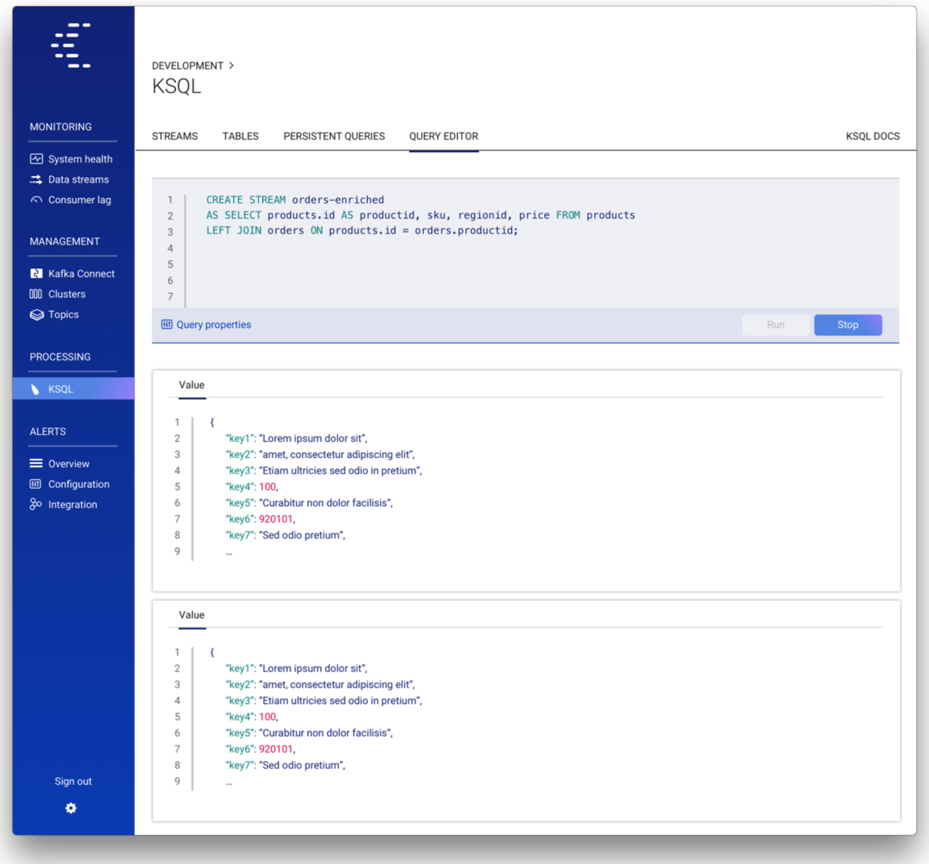
KSQL #FTW



1 UI

2 CLI

KSQL #FTW

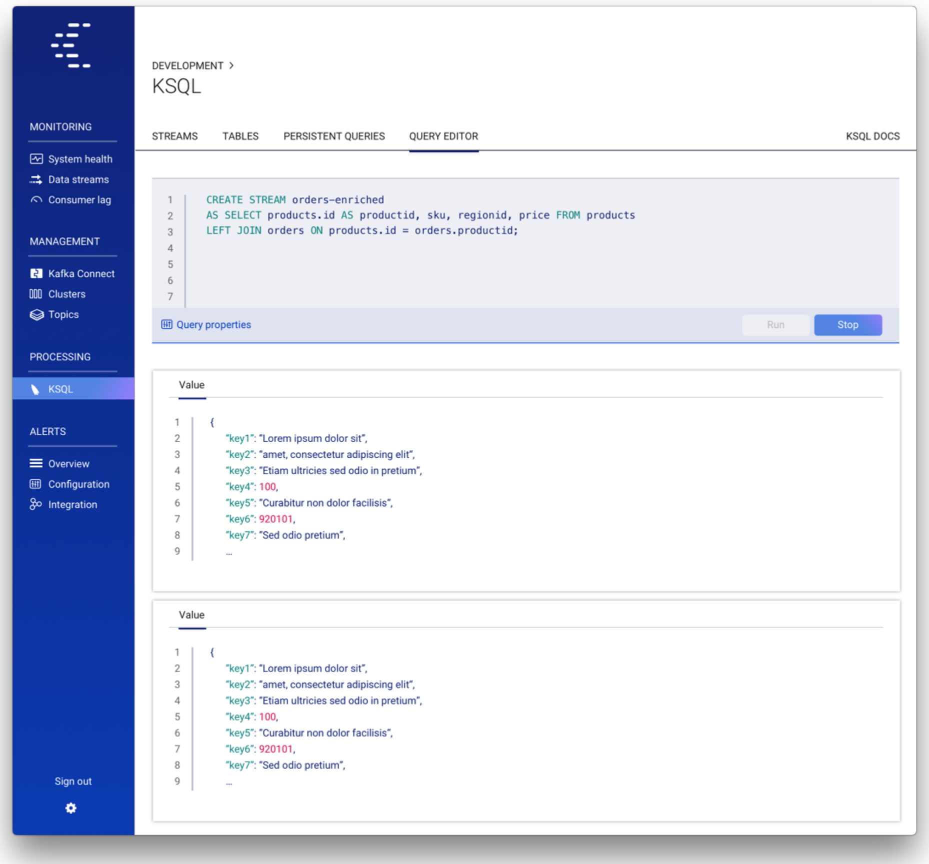


1 UI

2 CLI

3 REST

KSQL #FTW



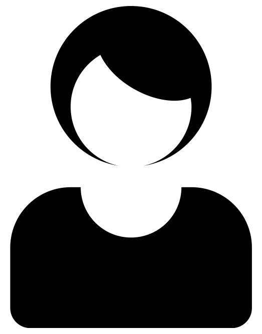
1 UI

2 CLI

3 REST

4 Headless

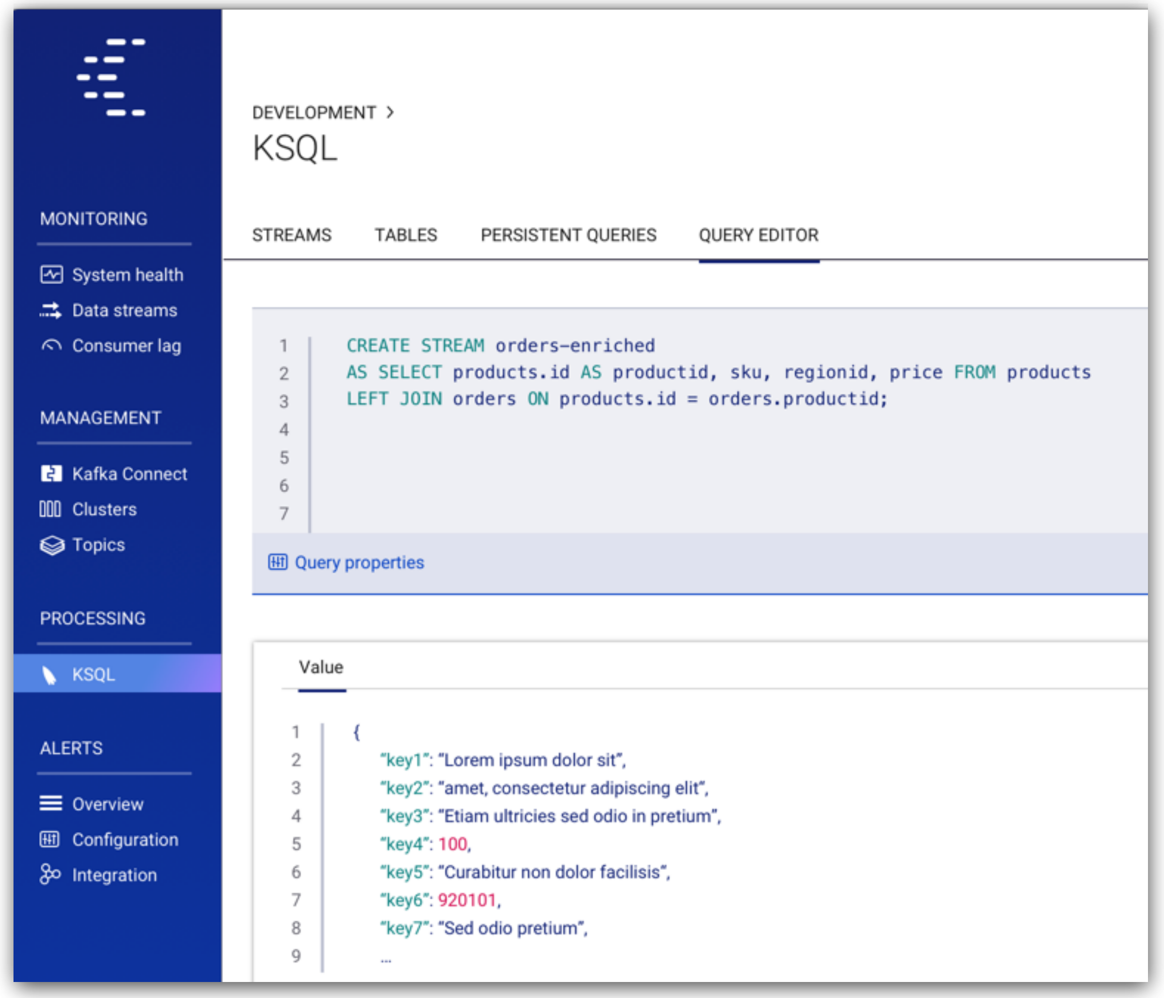
Interactive KSQL
for development and testing



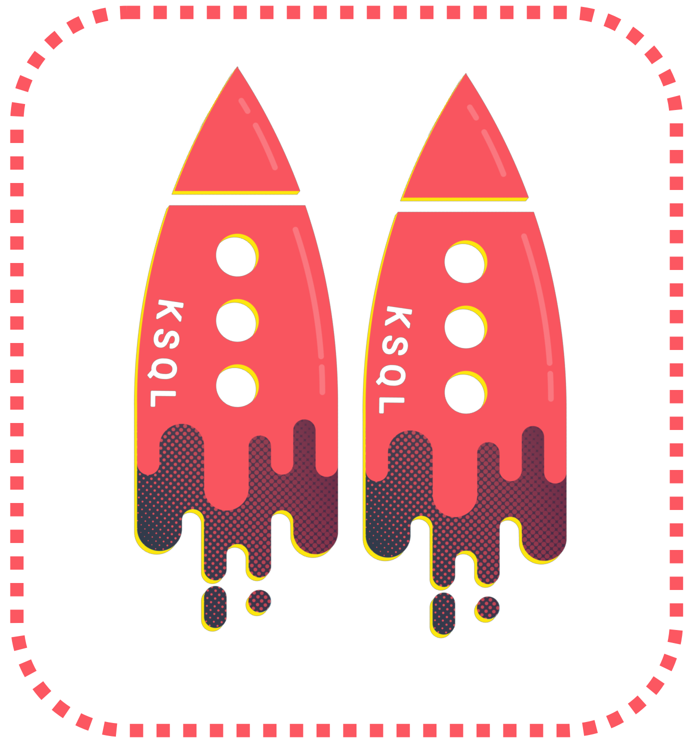
"Hmm, let me try out this idea…"

Interactive KSQL
for development and testing

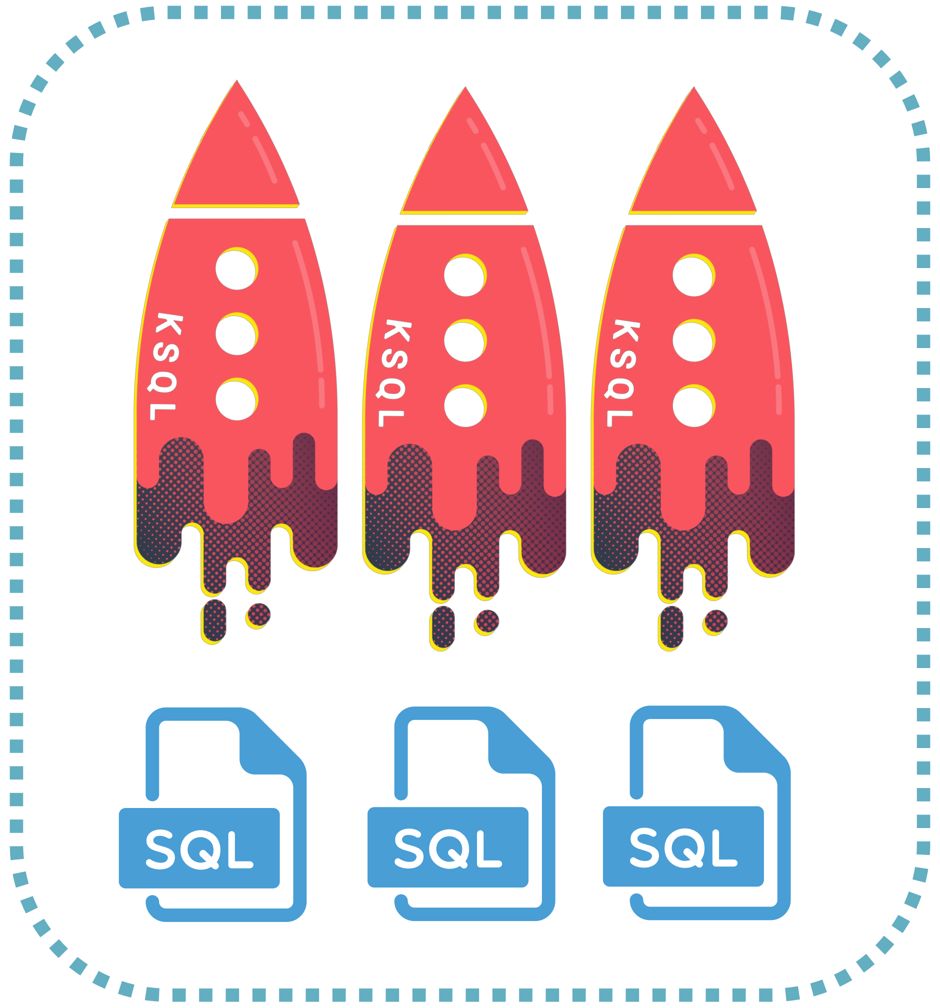
Headless KSQL
for Production



REST



Desired KSQL queries have been identified



"Hmm, let me try out this idea..."

Interaction with Kafka

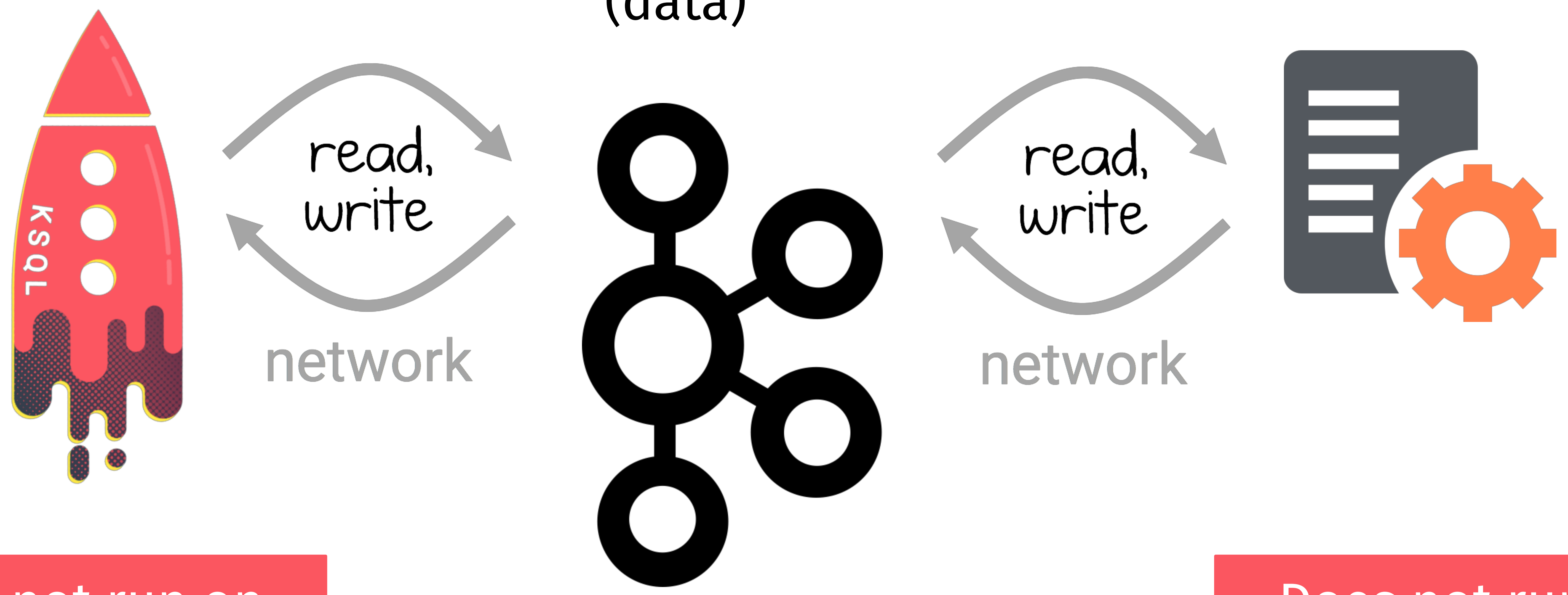
Kafka
(data)

Interaction with Kafka

KSQL
(processing)

Kafka
(data)

JVM application
with Kafka Streams (processing)

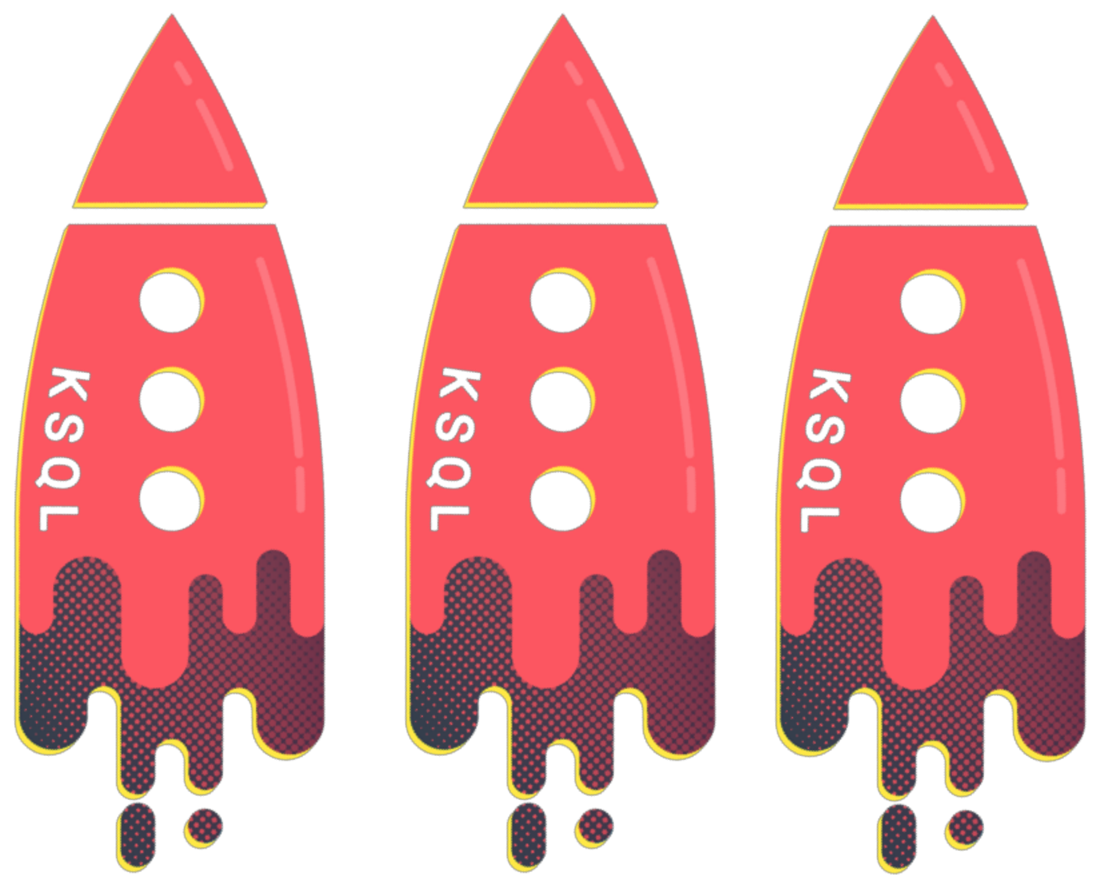


Does not run on
Kafka brokers

Does not run on
Kafka brokers

Interaction with Kafka

KSQL
(processing)



read, write

network

Kafka
(data)

read, write

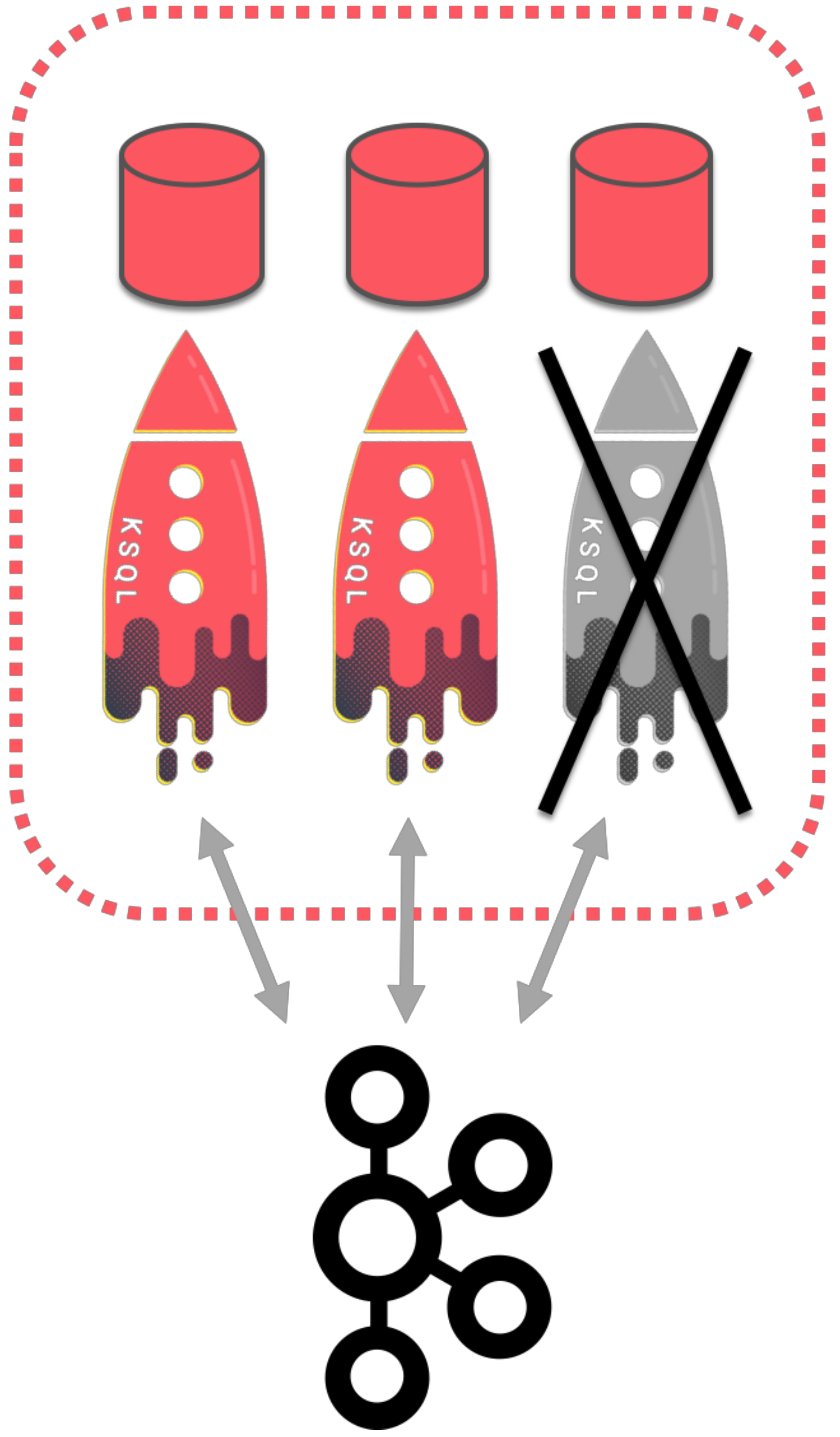
network

JVM application
with Kafka Streams (processing)

Does not run on Kafka brokers

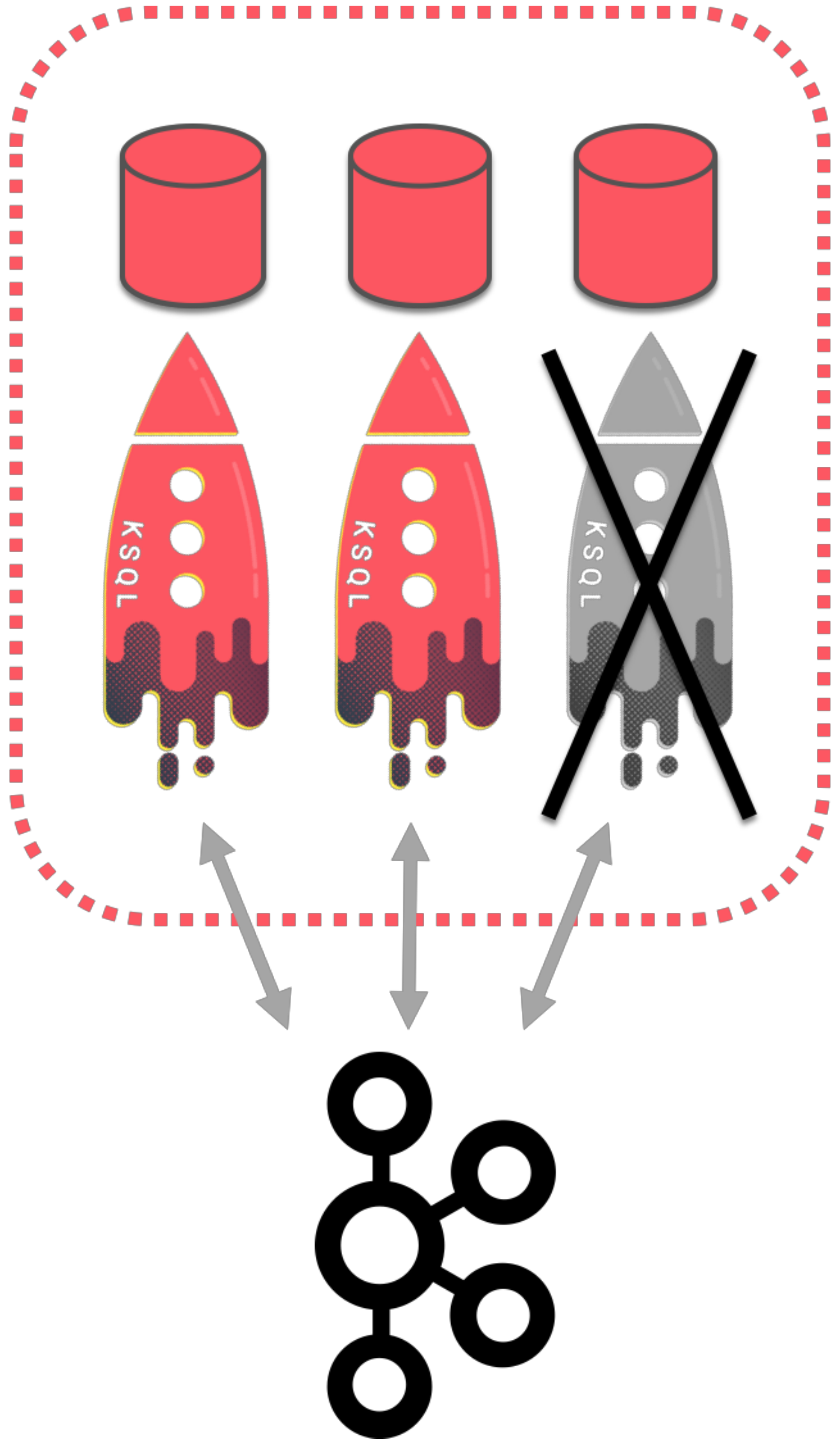
Does not run on Kafka brokers

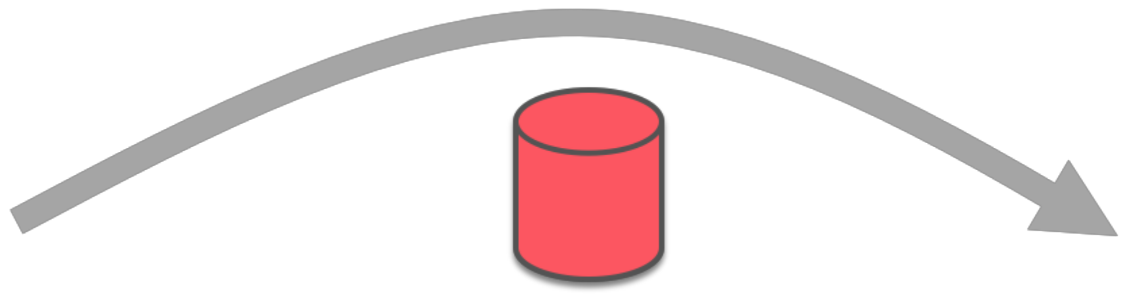
Fault-Tolerance, powered by Kafka

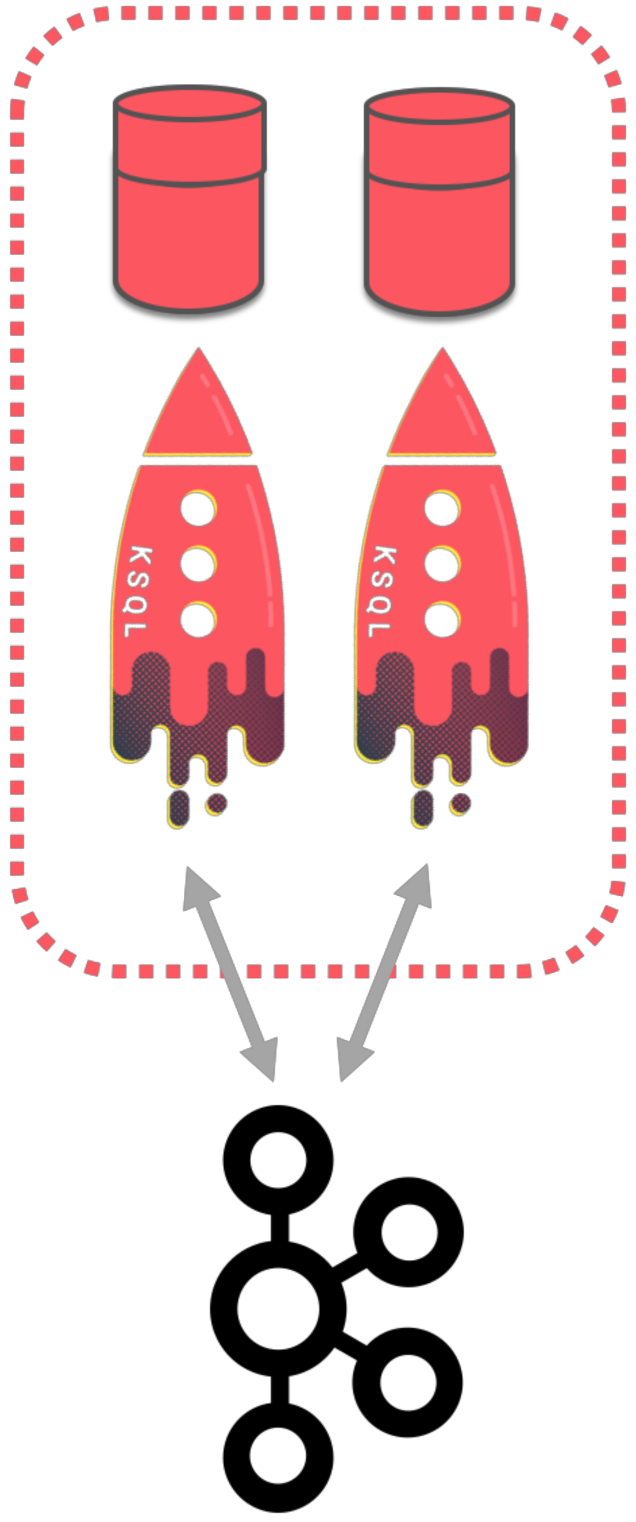


Fault-Tolerance, powered by Kafka

#3 died so #1 and #2 take over



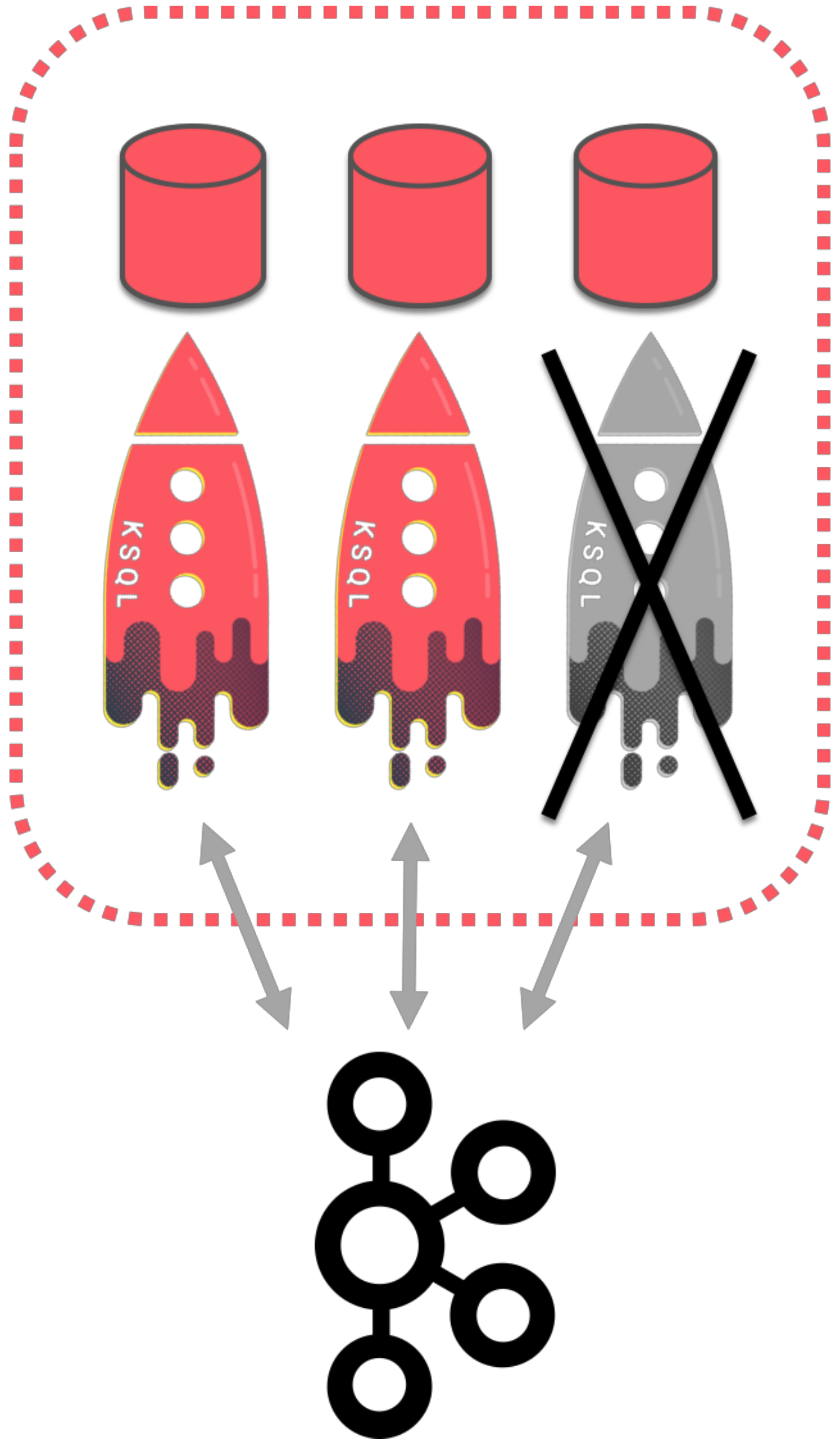


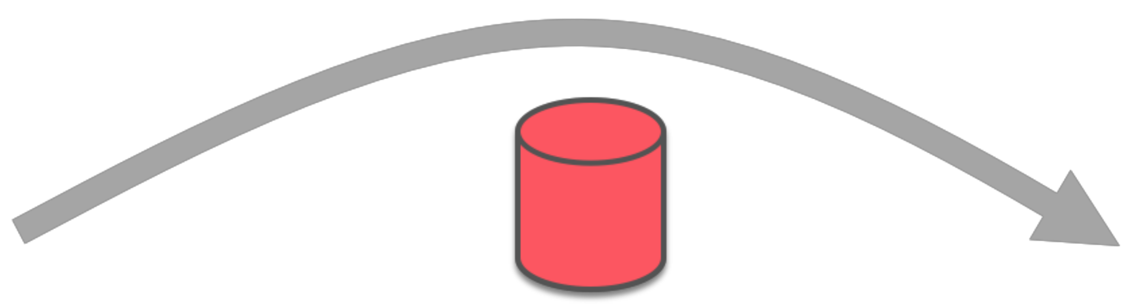


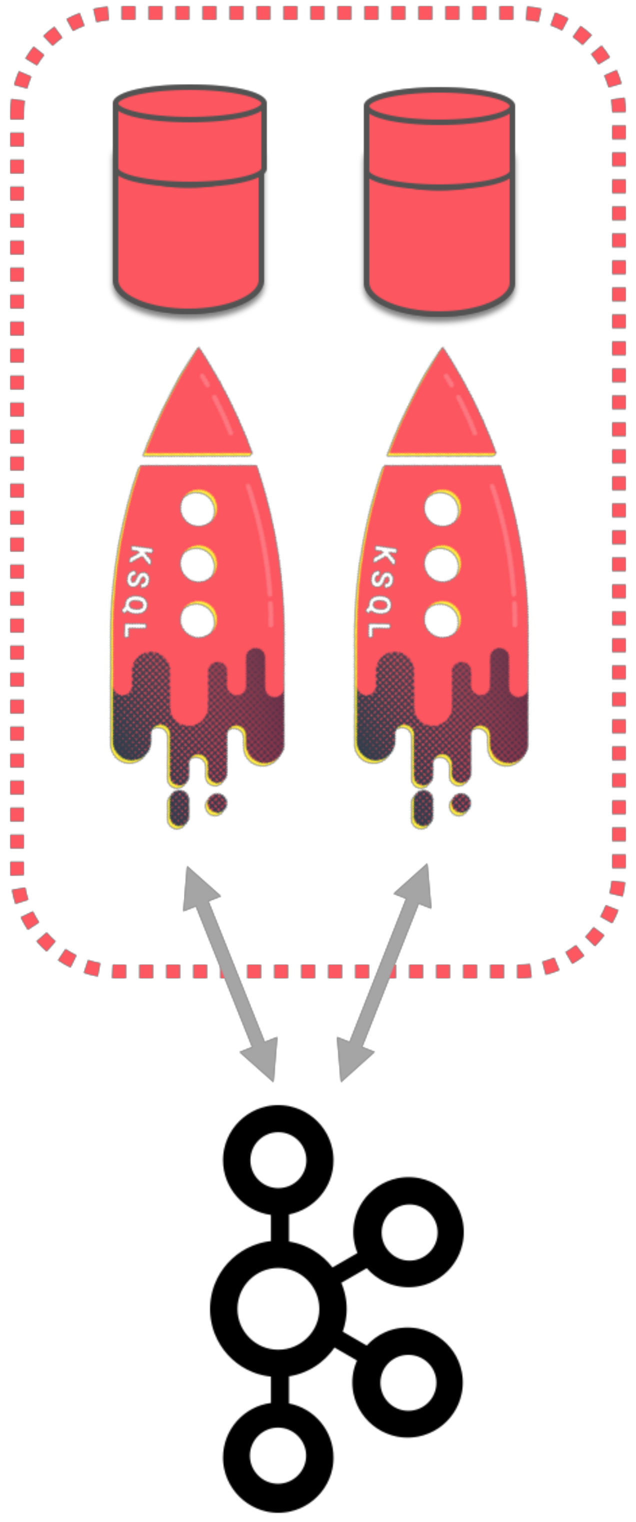
1. Kafka consumer group <u>rebalance</u> is triggered
2. Processing and state of #3 is <u>migrated</u> via Kafka to remaining servers #1 + #2

Fault-Tolerance, powered by Kafka

#3 died so #1 and #2 take over

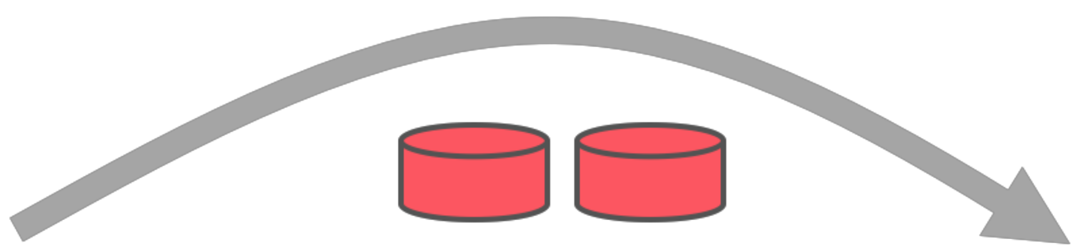


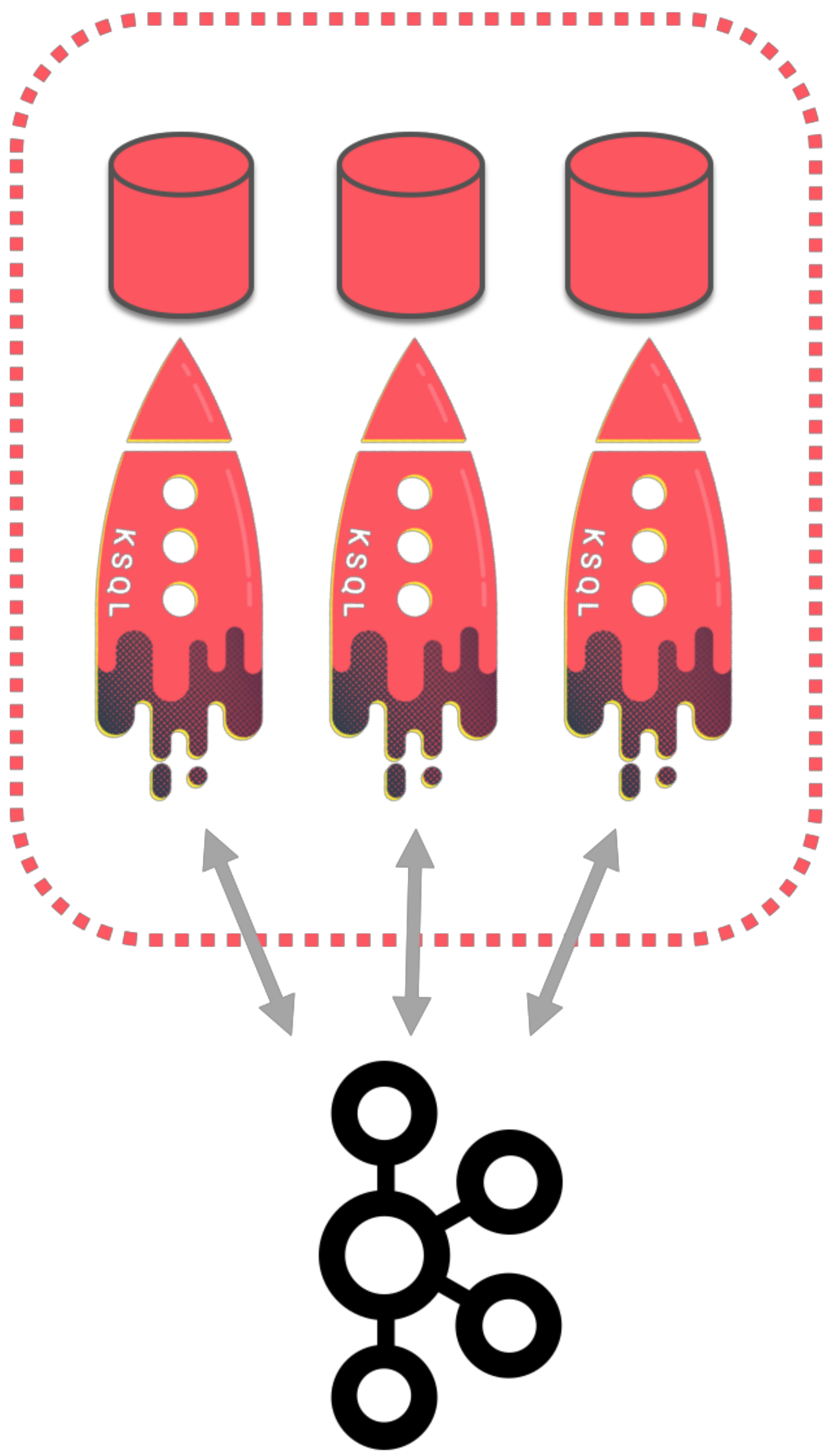




1. Kafka consumer group rebalance is triggered
2. Processing and state of #3 is migrated via Kafka to remaining servers #1 + #2

#3 is back so the work is split again





1. Kafka consumer group rebalance is triggered
2. Part of processing incl. state is migrated via Kafka from #1 + #2 to server #3

Standing on the shoulders of Streaming Giants

Standing on the shoulders of Streaming Giants

Standing on the shoulders of Streaming Giants

Producer,
Consumer APIs

```
subscribe(), poll(), send(),
flush(), beginTransaction(), …
```

Standing on the shoulders of Streaming Giants

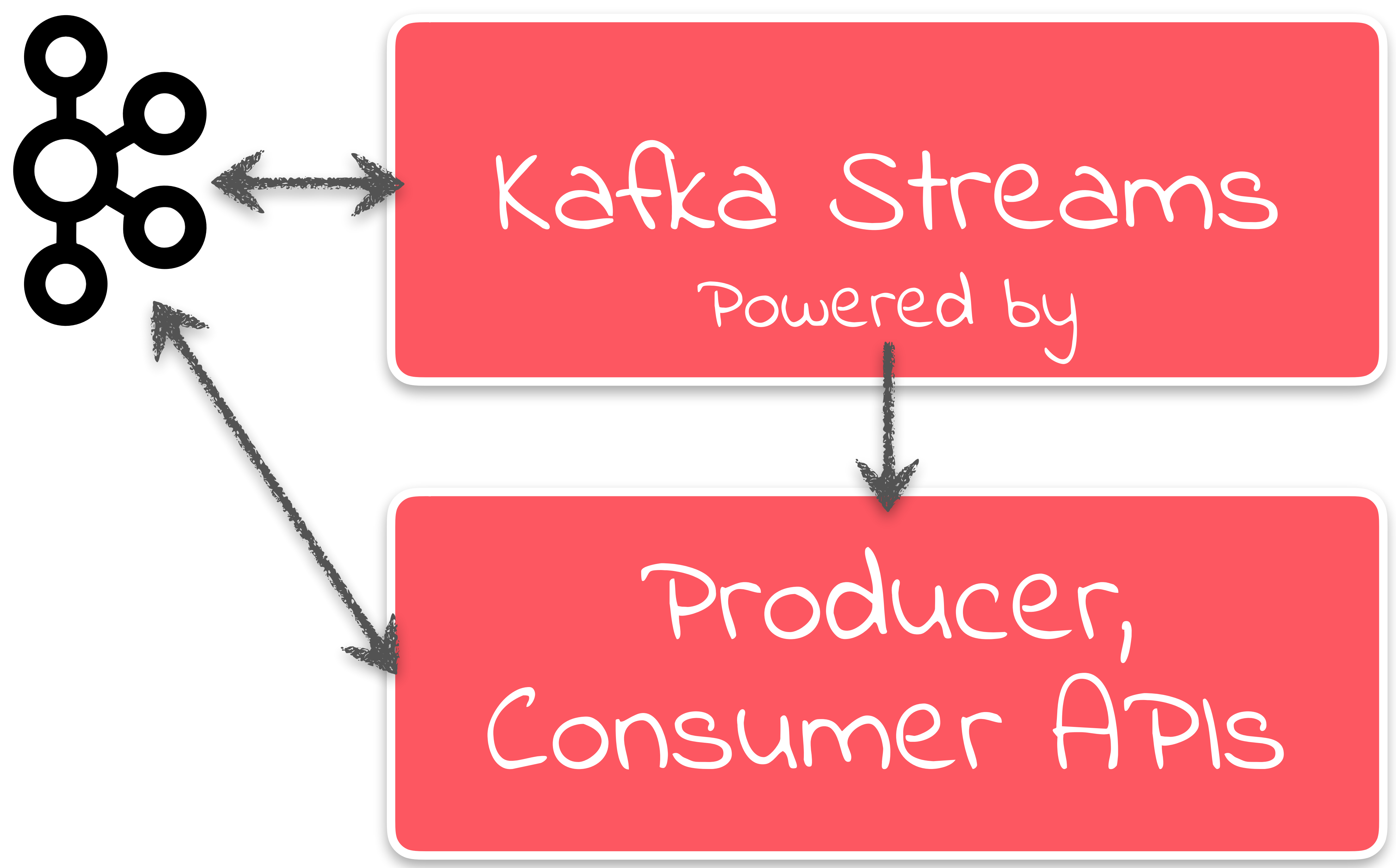
Kafka Streams

Producer, Consumer APIs

```
KStream<>, KTable<>, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …
```

```
subscribe(), poll(), send(),
flush(), beginTransaction(), …
```

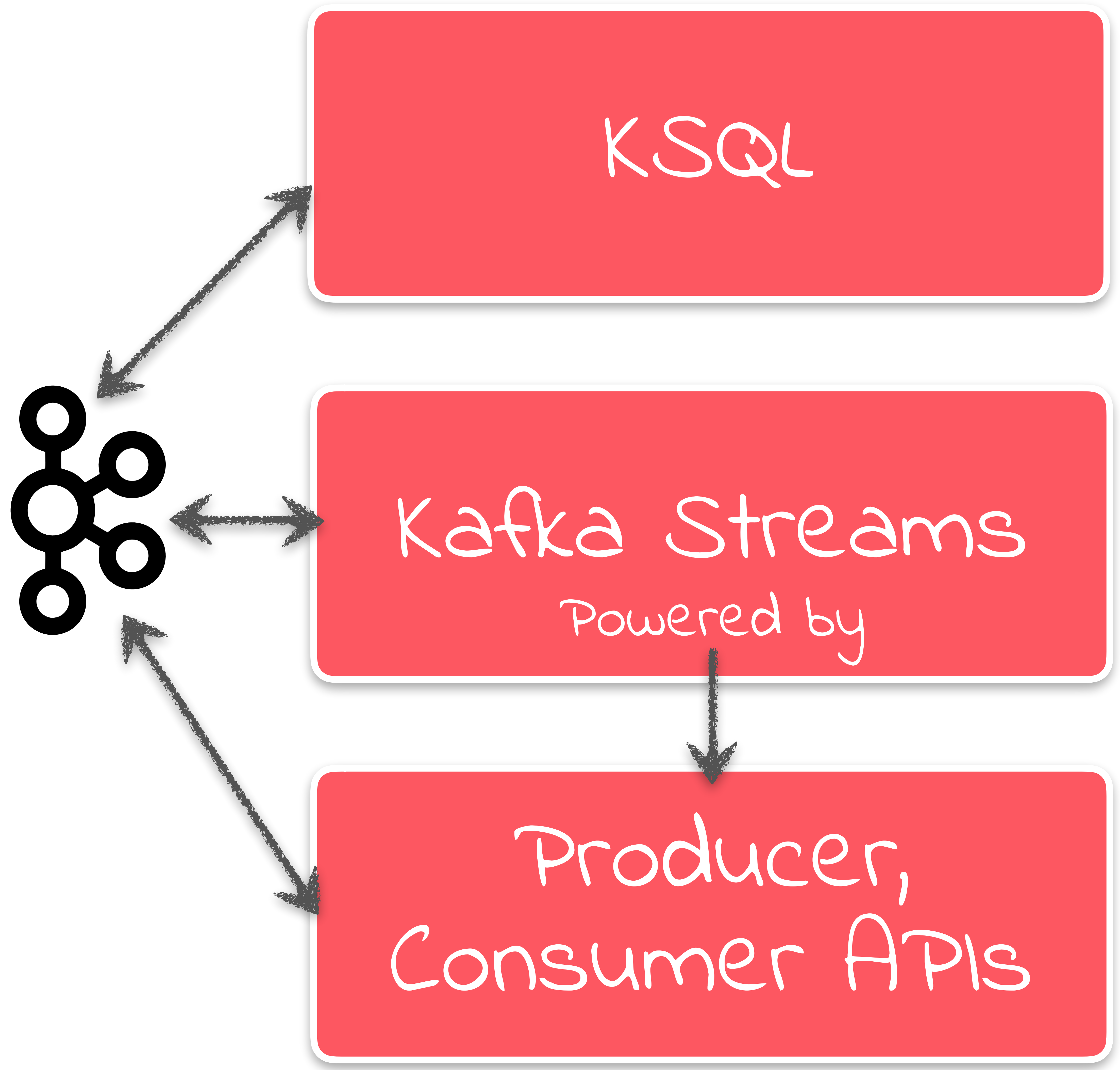
Standing on the shoulders of Streaming Giants




```
KStream<>, KTable<>, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …
```

```
subscribe(), poll(), send(),
flush(), beginTransaction(), …
```

Standing on the shoulders of Streaming Giants



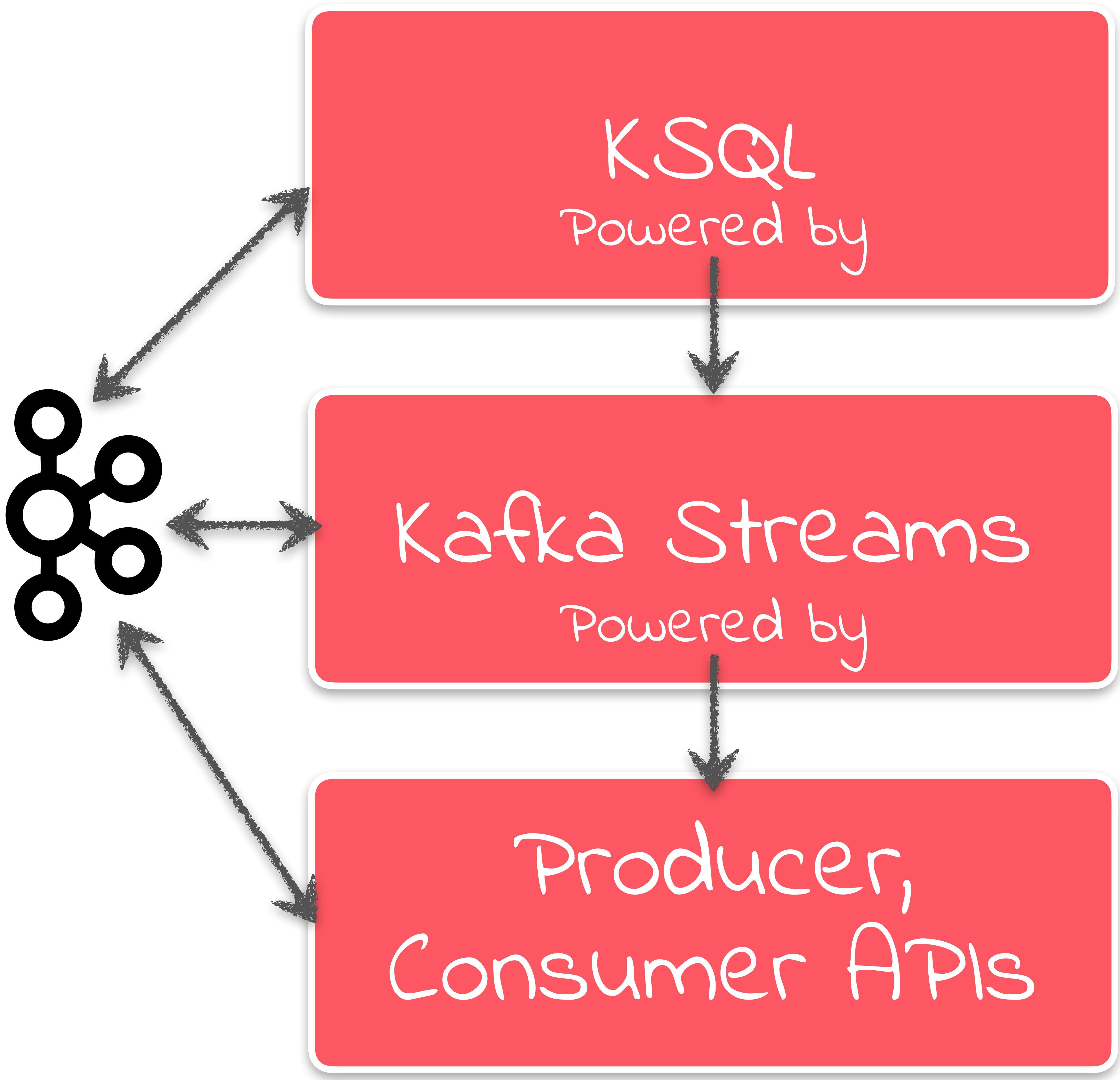

```
CREATE STREAM,
CREATE TABLE, SELECT, JOIN,
GROUP BY, SUM, …
```

KSQL UDFs

```
KStream<>, KTable<>, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …
```

```
subscribe(), poll(), send(),
flush(), beginTransaction(), …
```

Standing on the shoulders of Streaming Giants




```
CREATE STREAM,
CREATE TABLE, SELECT, JOIN,
GROUP BY, SUM, …
```

KSQL UDFs

```
KStream<>, KTable<>, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …
```

```
subscribe(), poll(), send(),
flush(), beginTransaction(), …
```

Standing on the shoulders of Streaming Giants

KSQL

Powered by

Kafka Streams

Powered by

Producer, Consumer APIs

Ease of use

Flexibility

```
CREATE STREAM,
CREATE TABLE, SELECT, JOIN,
GROUP BY, SUM, …
```

KSQL UDFs

```
KStream<>, KTable<>, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …
```

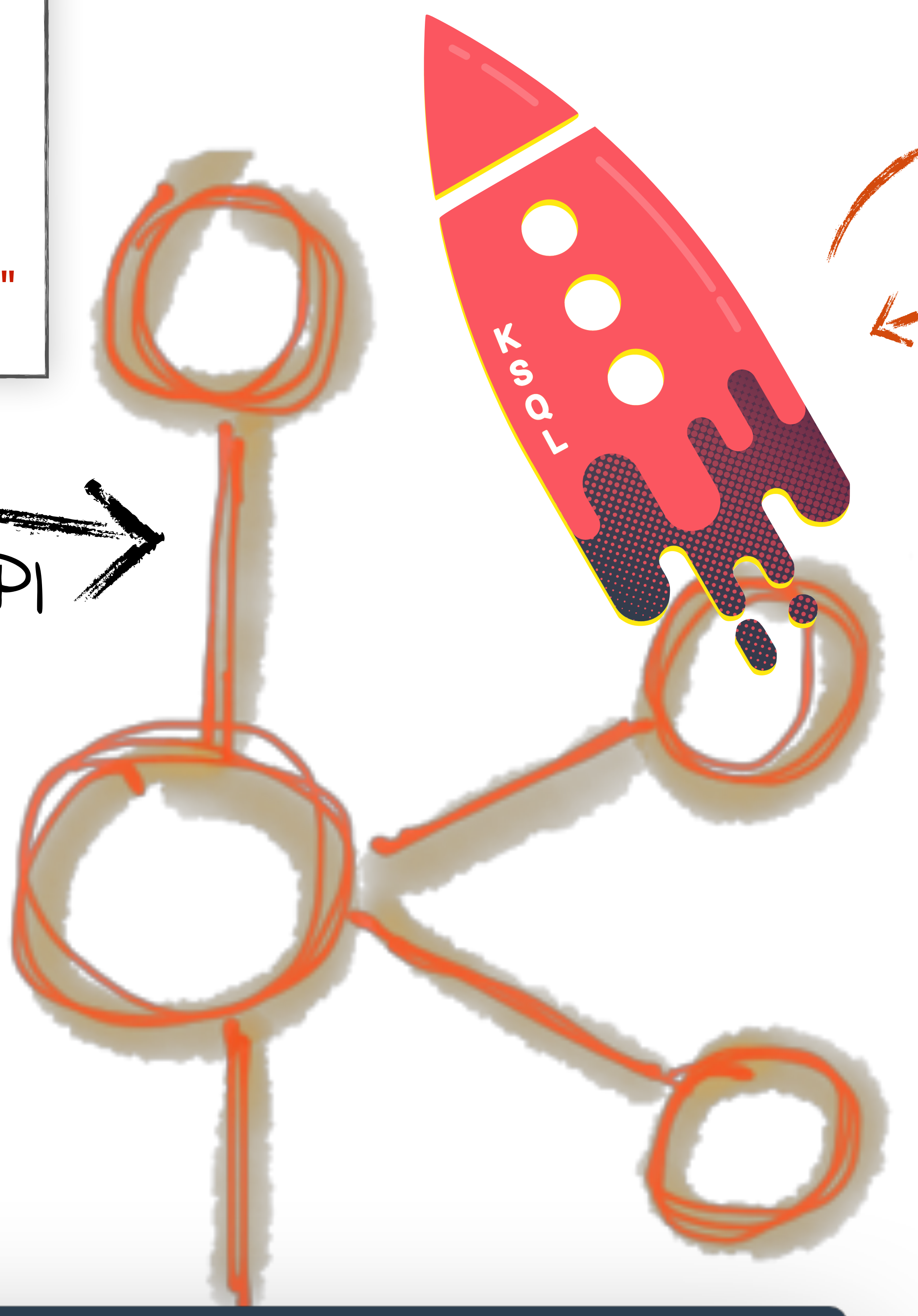
```
subscribe(), poll(), send(),
flush(), beginTransaction(), …
```

Demo time!

{
"rating_id": 2133,
"user_id": 9,
"stars": 1,
"route_id": 7219,
"rating_time": 1519402815063,
"channel": "web",
"message": "worst. flight. ever. #neveragain"
}
Producer API
Kafka Connect
MySQL
Kafka Connect
Debezium
{
"uid": 9,
"name": "Neha",
"locale": "en_US",
"address_city": "Palo Alto",
"elite": "P"
}
KSQL
kibana
80
60
46
31
26
Elasticsearch

```
{
 "rating_id": 5313,
 "user_id": 3,
 "stars": 4,
 "route_id": 6975,
 "rating_time": 1519304105213,
 "channel": "web",
 "message": "worst. flight. ever. #neveragain"
}
```


Producer API



Filter all ratings where STARS<3

POOR_RATINGS

```
CREATE STREAM POOR_RATINGS AS
SELECT * FROM ratings WHERE STARS <3
```


Do you think that’s a **table**
you are querying ?

The Stream-Table Duality

Table
(balance)

Stream
(payments)

time

The Stream-Table Duality

Table
(balance)

Stream
(payments)

Alice	+ €50

time

The Stream-Table Duality

Table
(balance)

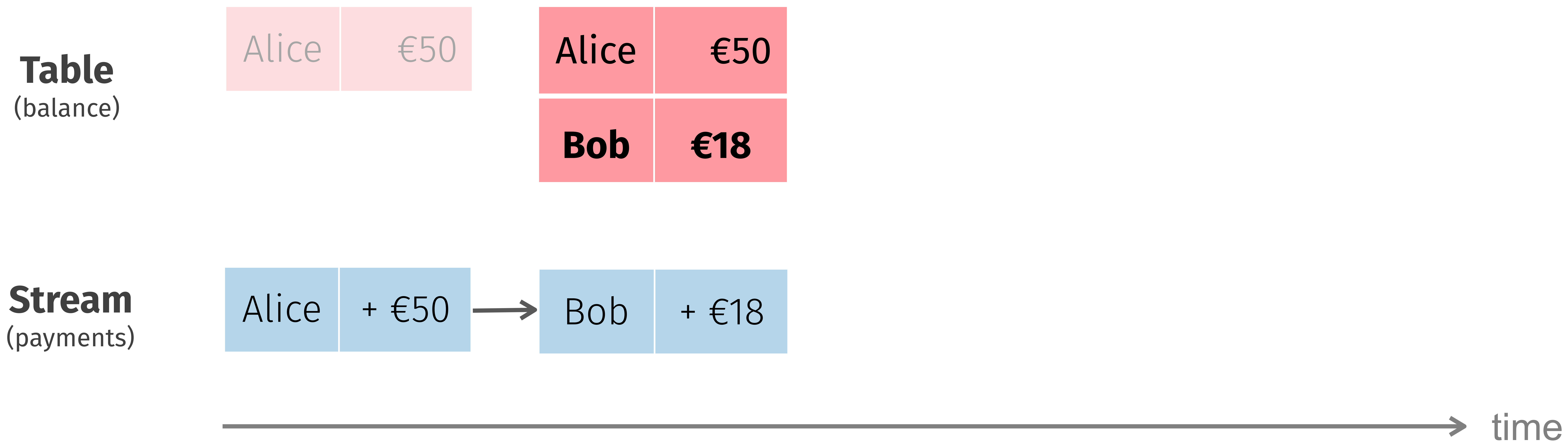
Alice	€50

Stream
(payments)

Alice	+ €50

time

The Stream-Table Duality



The Stream-Table Duality

Table
(balance)

Alice	€50

Alice	€50
Bob	€18

Alice	**€75**
Bob	€18

Stream
(payments)

Alice	+ €50	→	Bob	+ €18	→	Alice	+ €25

time

The Stream-Table Duality

Table
(balance)

Alice	€50

Alice	€50
Bob	€18

Alice	€75
Bob	€18

Alice	**€15**
Bob	€18

Stream
(payments)

Alice + €50 → Bob + €18 → Alice + €25 → Alice – €60

→ time


THE TRUTH IS OUT THERE

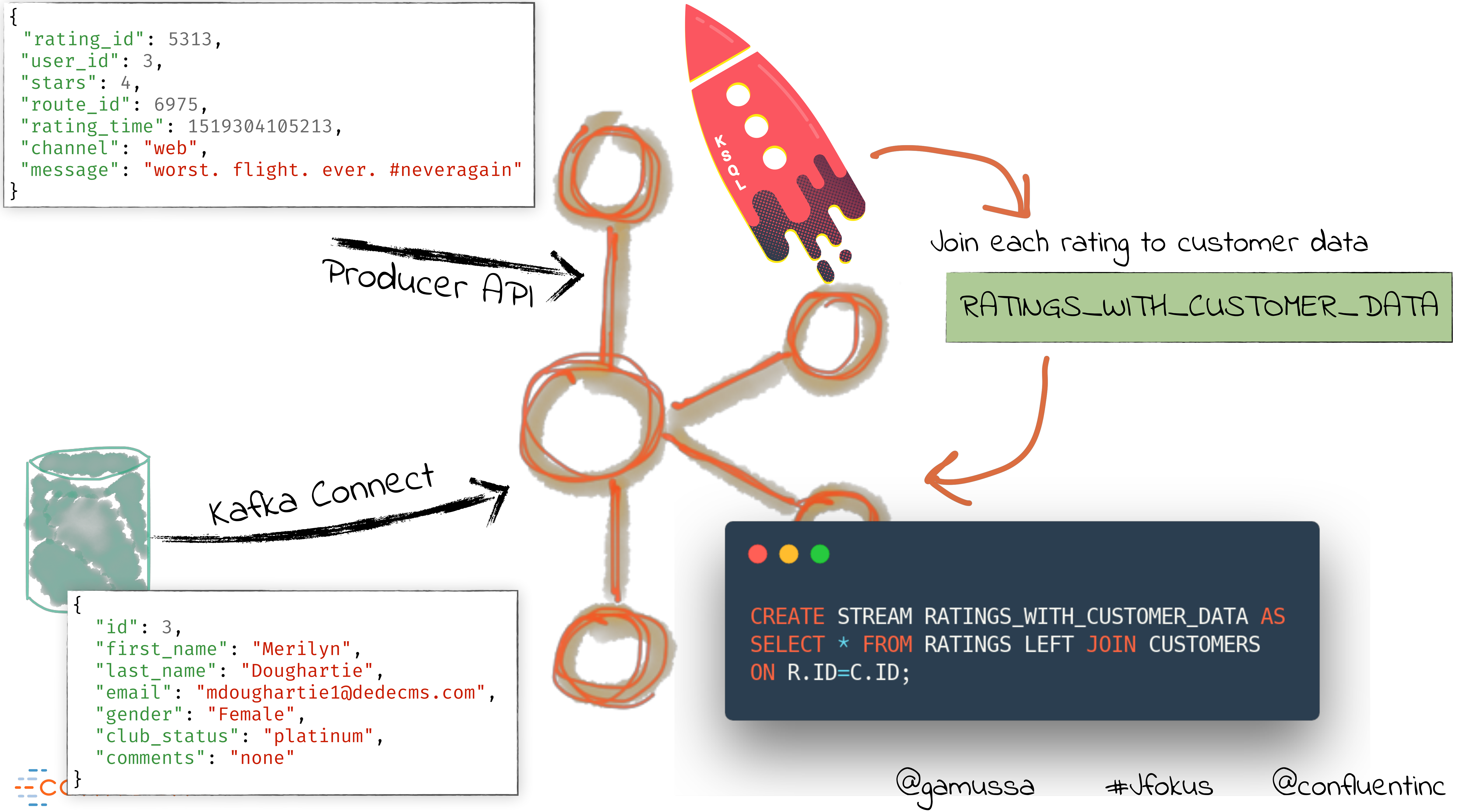
THE TRUTH IS OUT THERE

The truth is the log.
The database is a cache of a subset of the log.

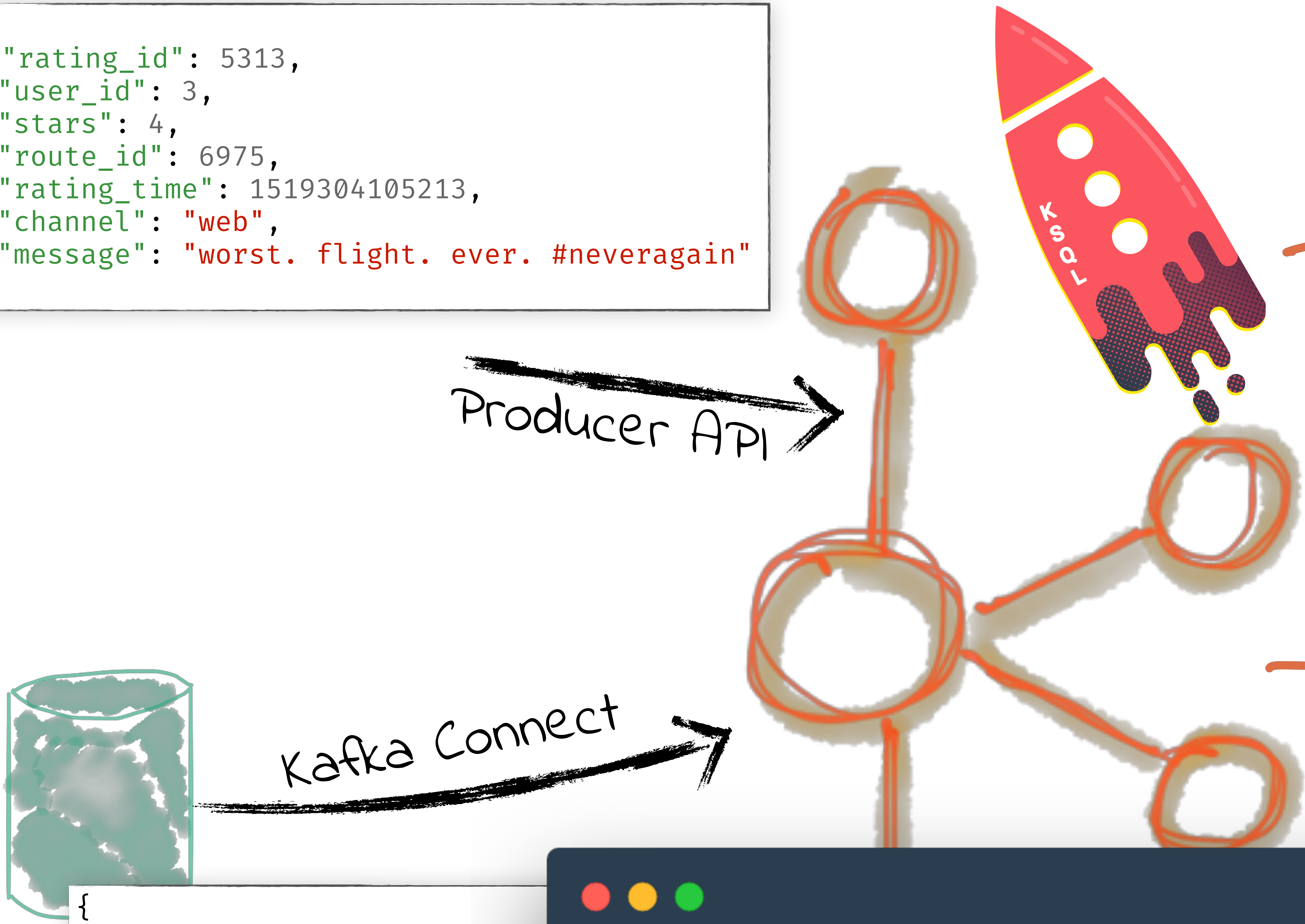
THE TRUTH IS OUT THERE

The truth is the log.
The database is a cache of a subset of the log.

—Pat Helland
Immutability Changes Everything
http://cidrdb.org/cidr2015/Papers/CIDR15_Paper16.pdf


{
"rating_id": 5313,
"user_id": 3,
"stars": 4,
"route_id": 6975,
"rating_time": 1519304105213,
"channel": "web",
"message": "worst. flight. ever. #neveragain"
}
Producer API
KSQL
Join each rating to customer data
RATINGS_WITH_CUSTOMER_DATA
Kafka Connect
{
"id": 3,
"first_name": "Merilyn",
"last_name": "Doughartie",
"email": "mdoughartie1@dedecms.com",
"gender": "Female",
"club_status": "platinum",
"comments": "none"
}
CREATE STREAM RATINGS_WITH_CUSTOMER_DATA AS
SELECT * FROM RATINGS LEFT JOIN CUSTOMERS
ON R.ID=C.ID;
@gamussa
#Jfokus
@confluentinc

{
 "rating_id": 5313,
 "user_id": 3,
 "stars": 4,
 "route_id": 6975,
 "rating_time": 1519304105213,
 "channel": "web",
 "message": "worst. flight. ever. #neveragain"
}



Join each rating to customer data

RATINGS_WITH_CUSTOMER_DATA

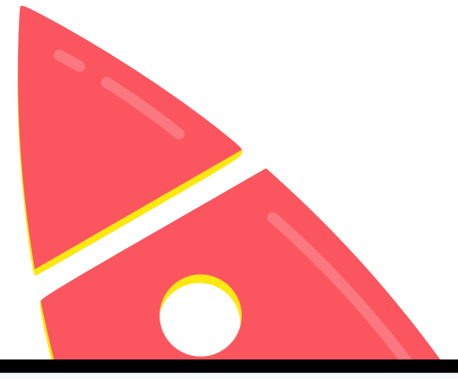
Filter for just PLATINUM customers

UNHAPPY_PLATINUM_CUSTOMERS

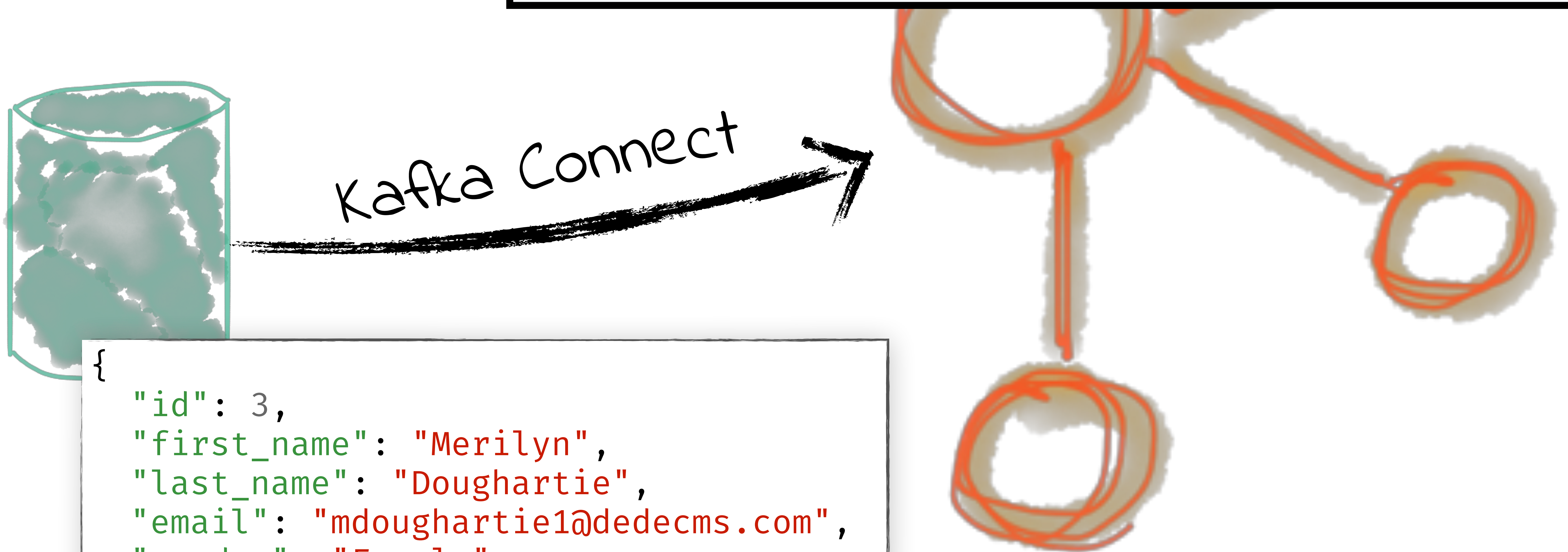
{
 "id": 3,
 "first_name": "Merilyn",
 "last_name": "Doughartie",
 "email": "mdoughartie1@ded
 "gender": "Female",
 "club_status": "platinum",
 "comments": "none"
}


```
CREATE STREAM UNHAPPY_PLATINUM_CUSTOMERS AS
SELECT * FROM RATINGS_WITH_CUSTOMER_DATA
WHERE  STARS < 3
```

```
{
 "rating_id": 5313,
 "user_id": 3,
 "stars": 4,
 "route_id": 6975,
 "rating_time": 1519304105
 "channel": "web",
 "message": "worst. flight.
}
```

```sql
CREATE TABLE RATINGS_BY_CLUB_STATUS AS
SELECT CLUB_STATUS, COUNT(*)
FROM RATINGS_WITH_CUSTOMER_DATA
     WINDOW TUMBLING (SIZE 1 MINUTES)
GROUP BY CLUB_STATUS;
```




```
{
  "id": 3,
  "first_name": "Merilyn",
  "last_name": "Doughartie",
  "email": "mdoughartie1@dedecms.com",
  "gender": "Female",
  "club_status": "platinum",
  "comments": "none"
}
```


Aggregate per-minute by CLUB_STATUS

RATINGS_BY_CLUB_STATUS_1MIN

Resources and Next Steps

https://github.com/confluentinc/demo-scene

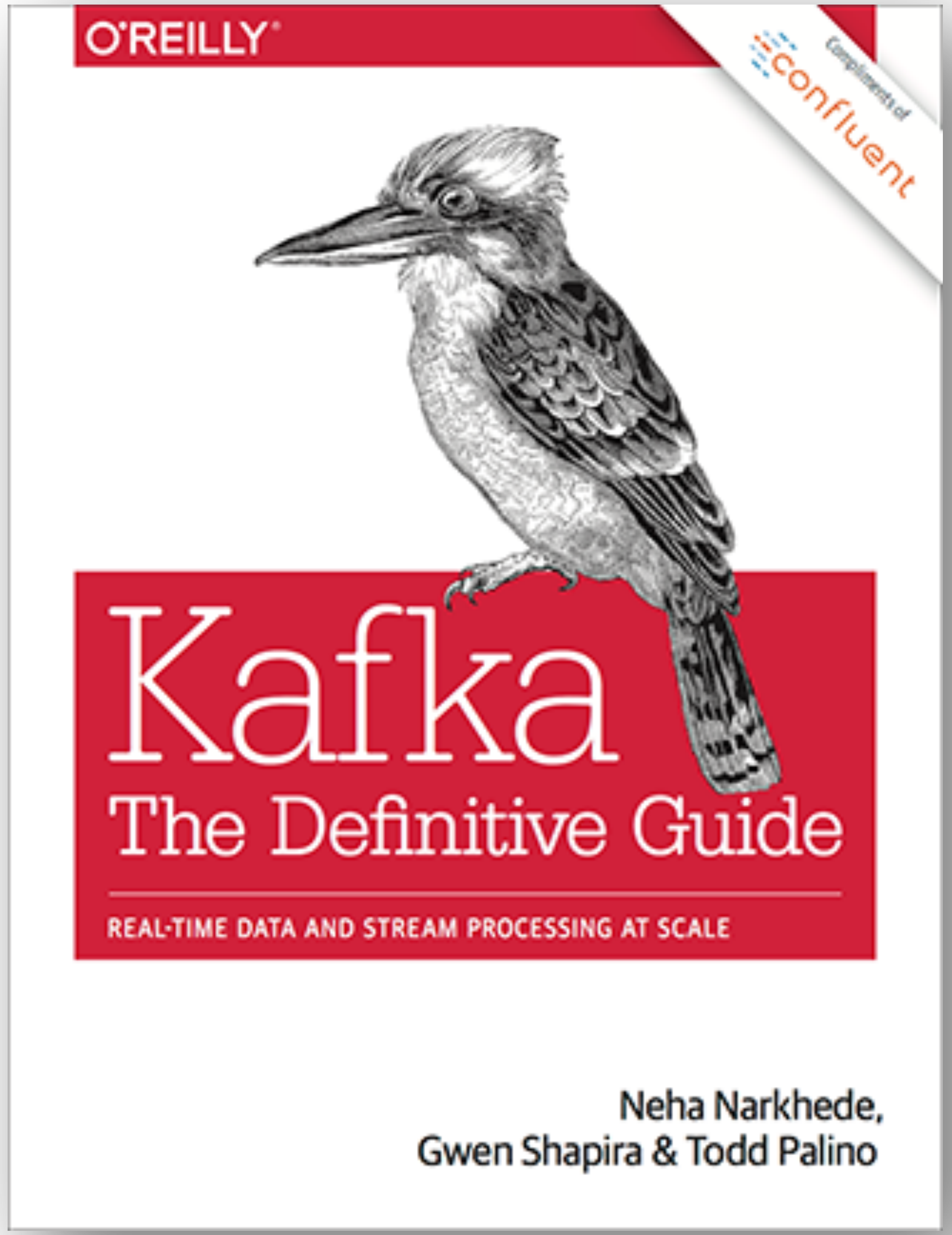
http://confluent.io

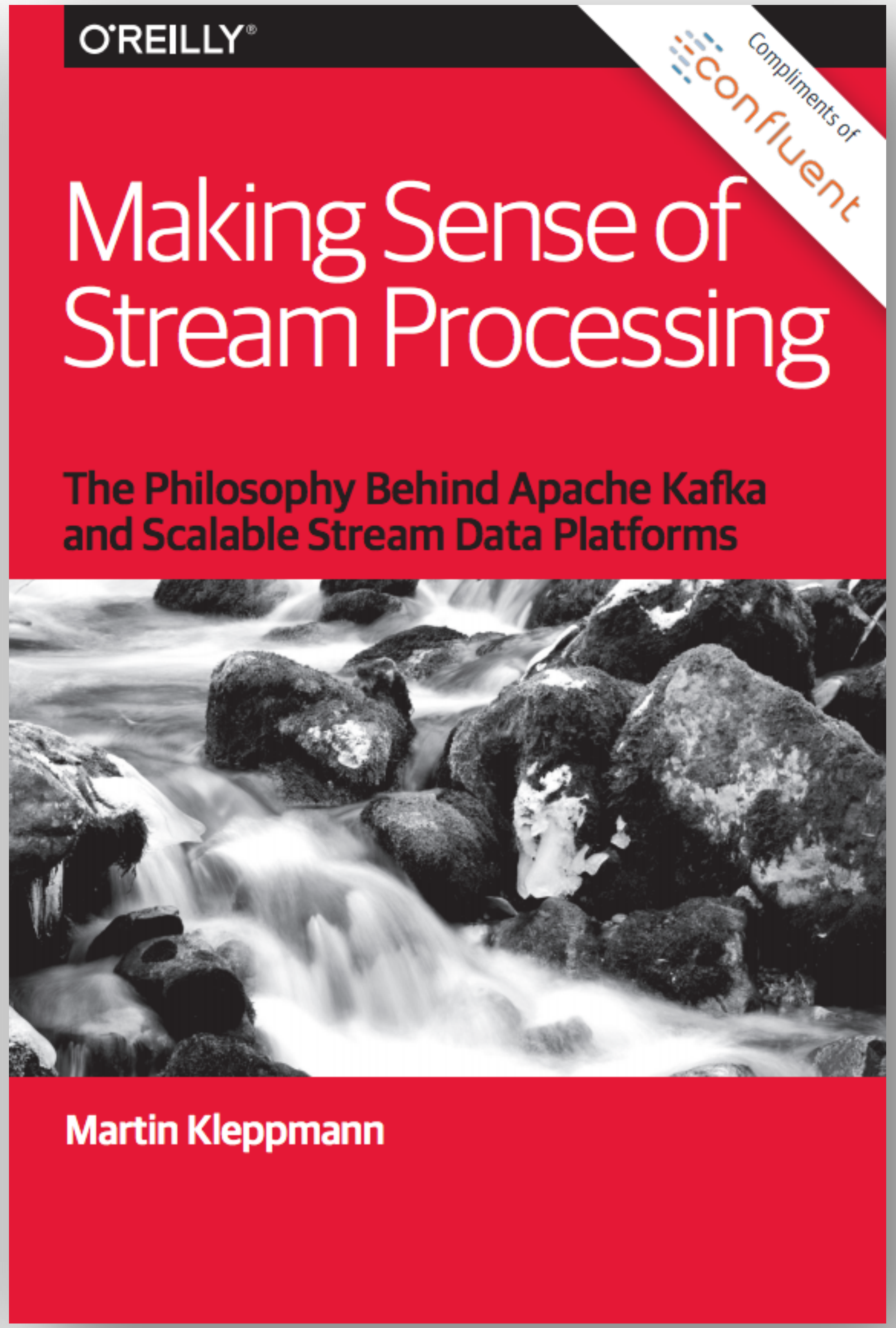
https://slackpass.io/confluentcommunity

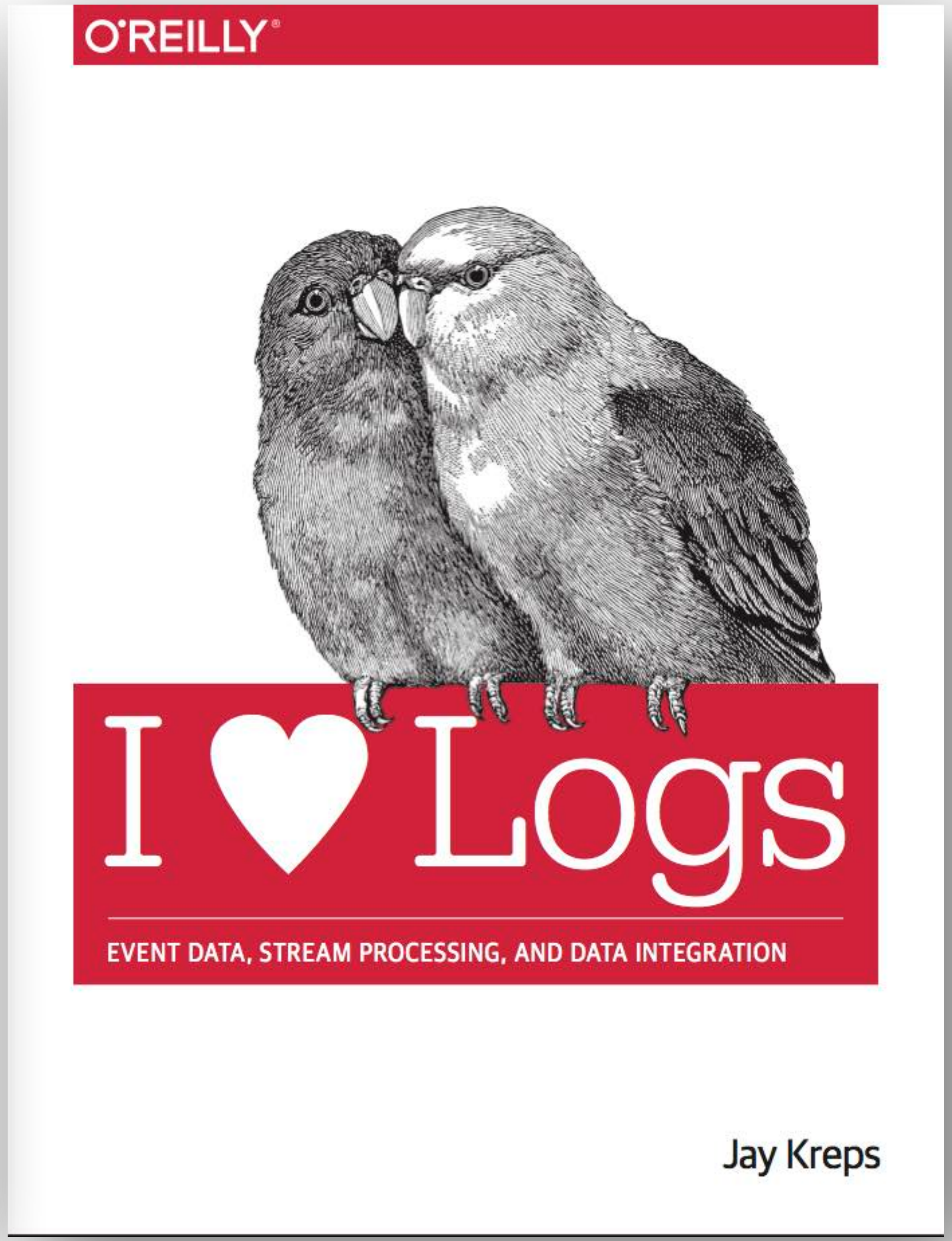
#ksql #connect

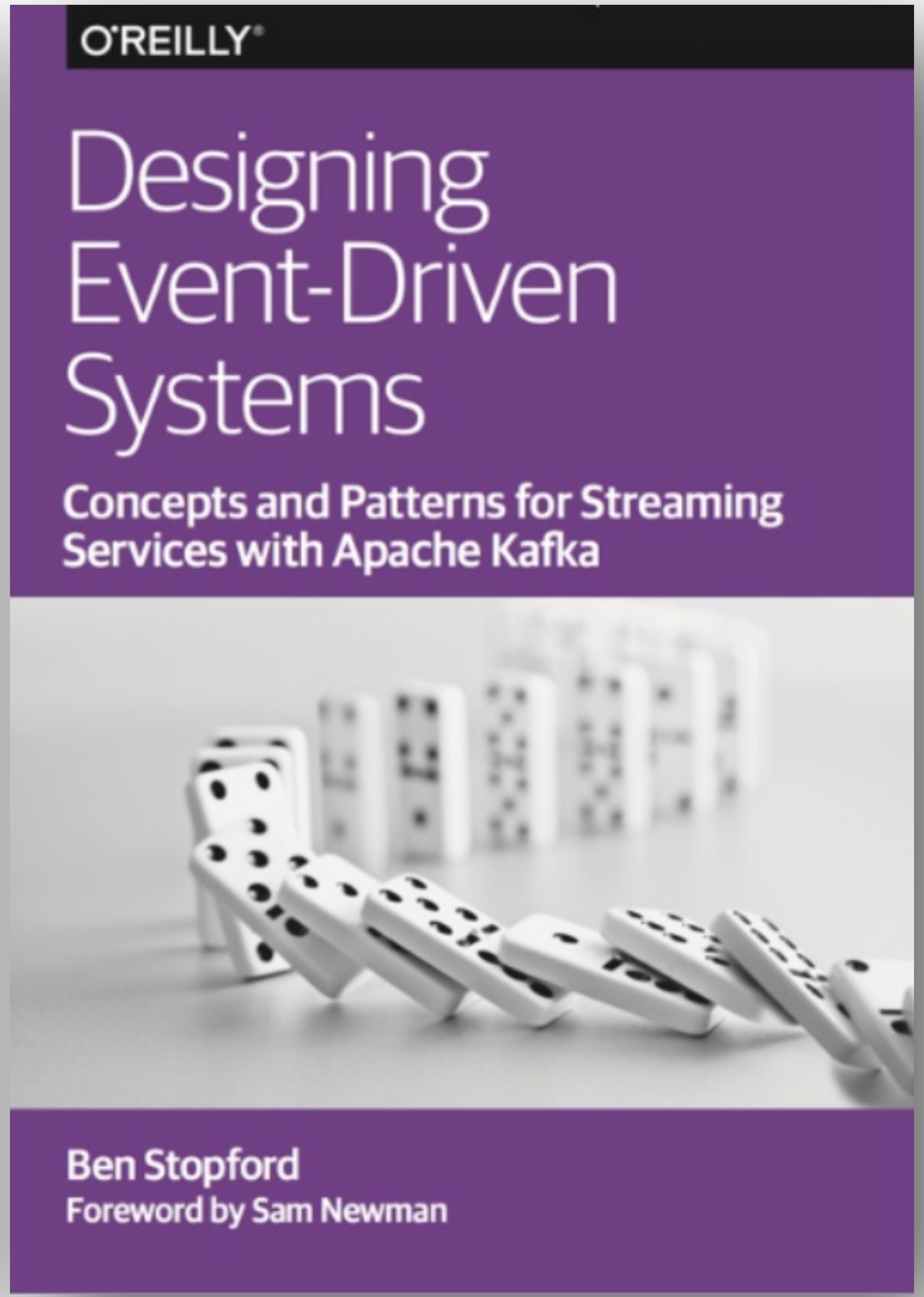
Free Books!

https://www.confluent.io/apache-kafka-stream-processing-book-bundle









THANKS!

@gamussa
viktor@confluent.io

confluent

One last thing…

kafka summit

https://kafka-summit.org

Gamov30

APRIL 2, 2019

NEW YORK CITY

REGISTER

LEARN MORE