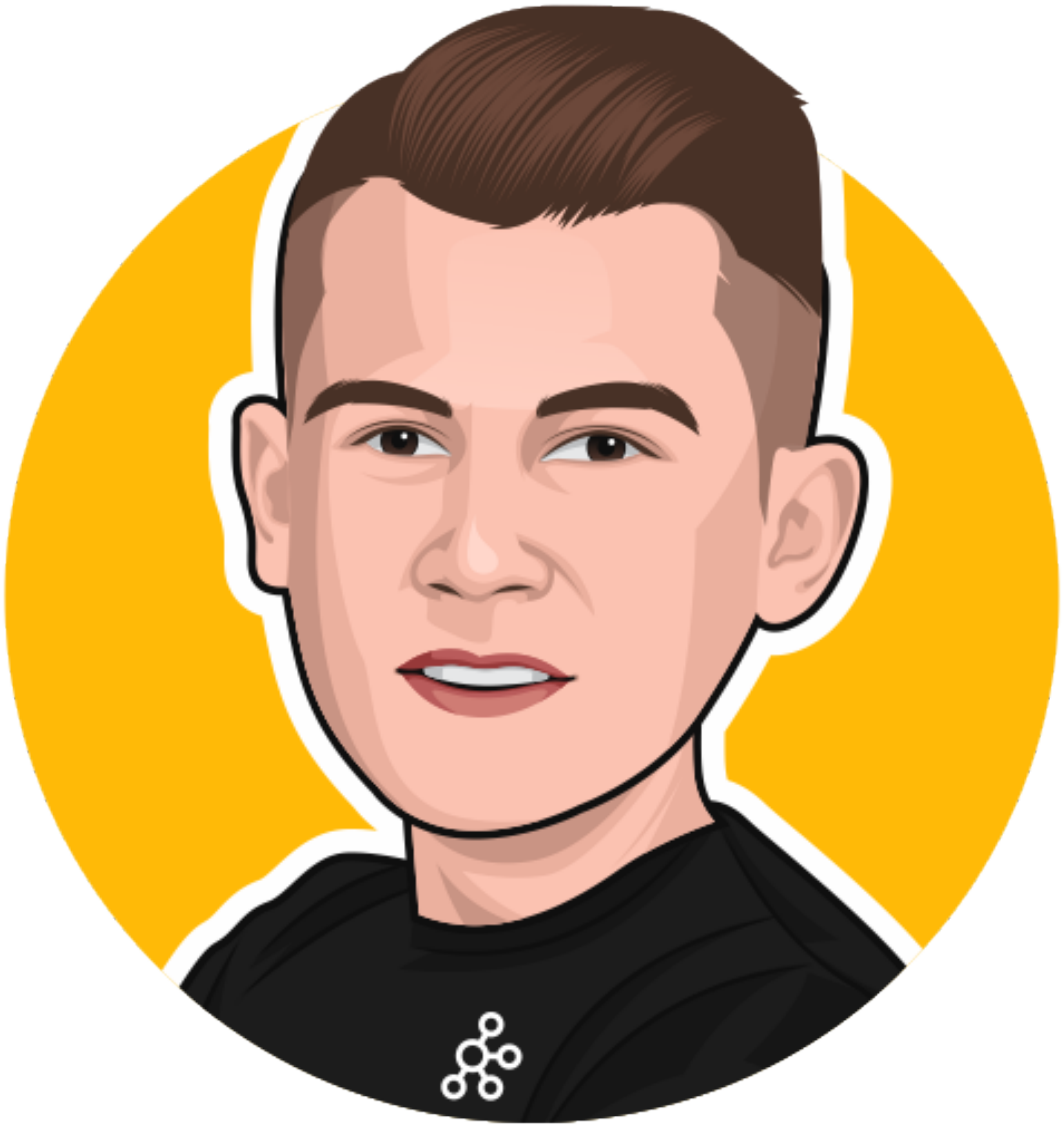
CONFLUENT

WHO'S TWEETING ABOUT #SCALALOVE

@GAMUSSA | #SCALALOVE | @CONFLUENTINC



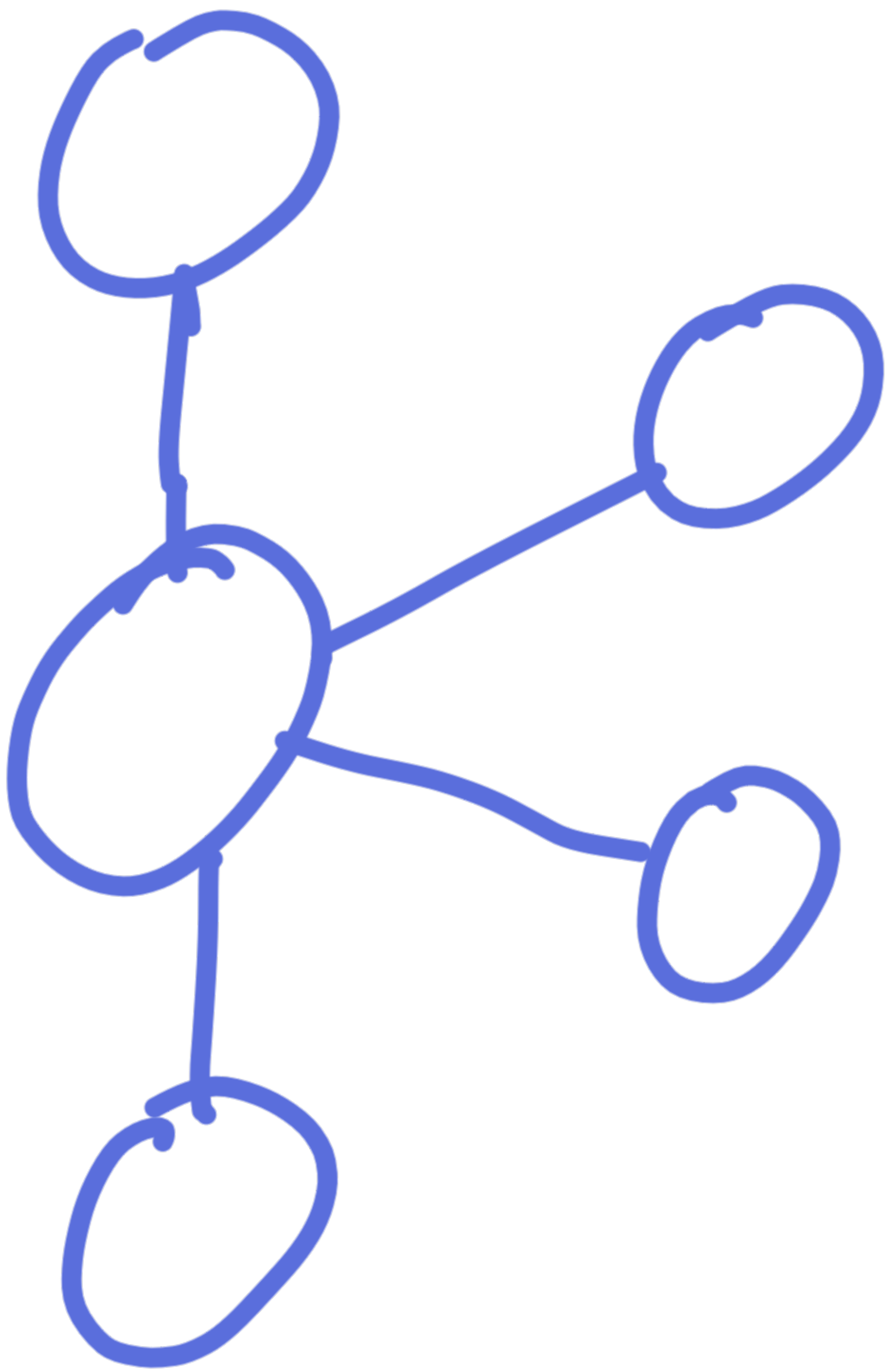
@GAMUSSA | #SCALALOVE | @CONFLUENTINC

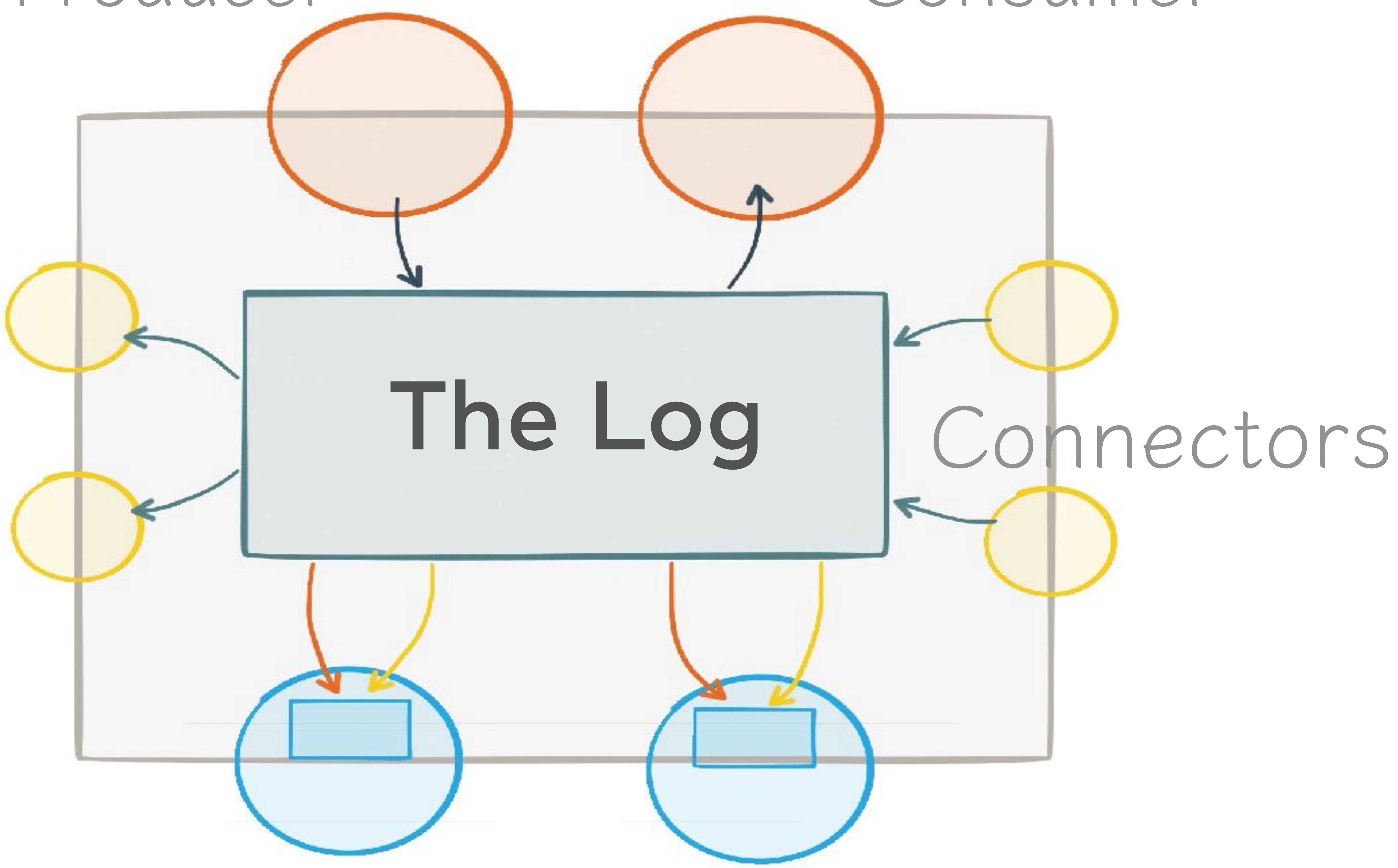
I BUILD HIGHLY SCALABLE

```
Hello World
```

APPS

APACHE KAFKA





```json
{
    "reading_ts": "2020-02-14T12:19:27Z",
    "sensor_id": "aa-101",
    "production_line": "w01",
    "widget_type": "acme94",
    "temp_celcius": 23,
    "widget_weight_g": 100
}
```


Photo by Franck V. on Unsplash

STREAMS OF EVENTS

CONFLUENT

CONFLUENT

STREAM PROCESSING WITH KSQLDB

STREAM: `widgets`

STREAM: `widgets_red`

STREAM PROCESSING WITH KAFKA STREAMS

CONFLUENT

STREAM: widgets

```
final StreamsBuilder builder = new StreamsBuilder()
    .stream("widgets", Consumed.with(stringSerde, widgetsSerde))
    .filter( (key, widget) → widget.getColour().equals("RED") )
    .to("widgets_red", Produced.with(stringSerde, widgetsSerde));
```

STREAM: widgets_red

STREAM PROCESSING WITH KSQLDB

CONFLUENT

STREAM: widgets

KSQLDB

```
CREATE STREAM widgets_red AS
SELECT * FROM widgets
WHERE color='RED';
```

STREAM: widgets_red


{
"reading_ts": "2020-02-14T12:19:27Z",
"sensor_id": "aa-101",
"production_line": "w01",
"widget_type": "acme94",
"temp_celcius": 23,
"widget_weight_g": 100
}
Photo by Franck V. on Unsplash

```sql
SELECT *
FROM WIDGETS
WHERE WEIGHT_G > 120
```

```sql
SELECT COUNT(*)
FROM WIDGETS
GROUP BY PRODUCTION_LINE
```

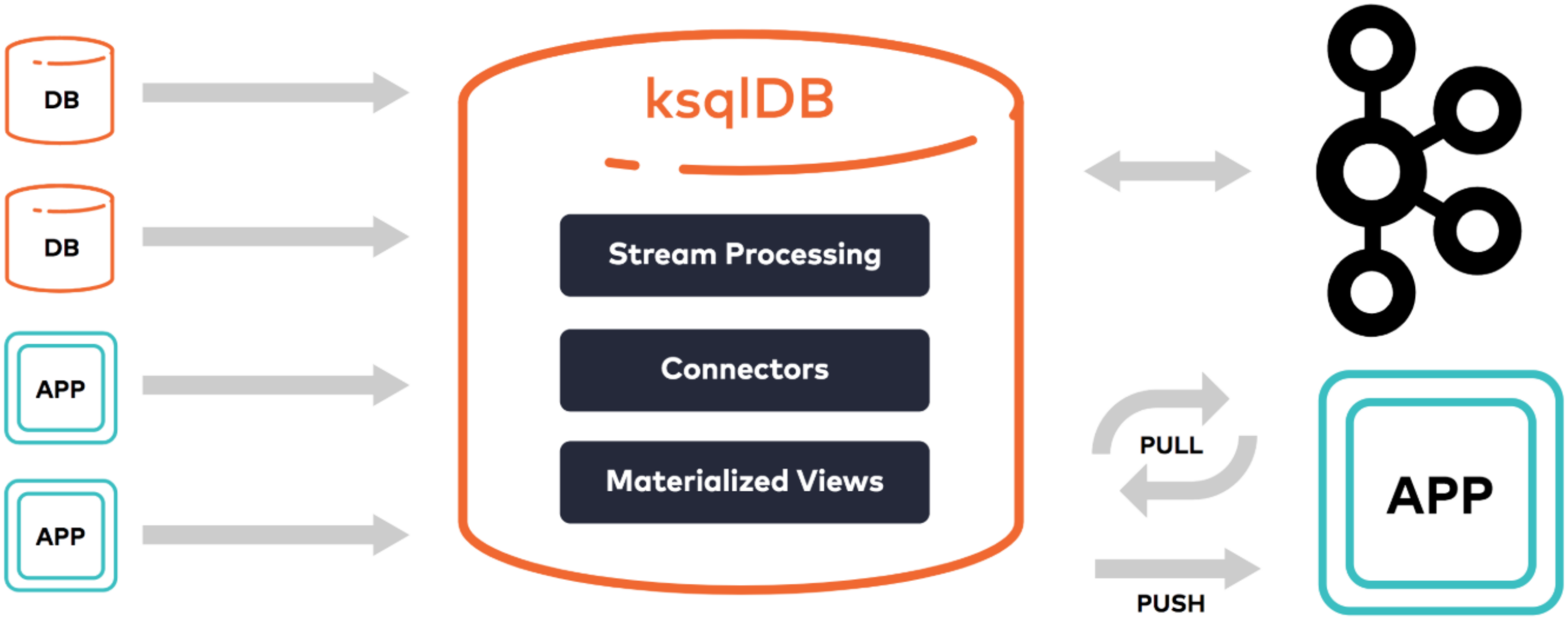
```sql
SELECT AVG(TEMP_CELCIUS) AS TEMP
FROM WIDGETS
GROUP BY SENSOR_ID
HAVING TEMP>20
```


```json
{
    "reading_ts": "2020-02-14T12:19:27Z",
    "sensor_id": "aa-101",
    "production_line": "w01",
    "widget_type": "acme94",
    "temp_celcius": 23,
    "widget_weight_g": 100
}
```


Object store,
data warehouse,
RDBMS

```sql
CREATE SINK CONNECTOR dw WITH (
'connector.class' = 'S3Connector',
'topics' = 'widgets');
```

KSQLDB

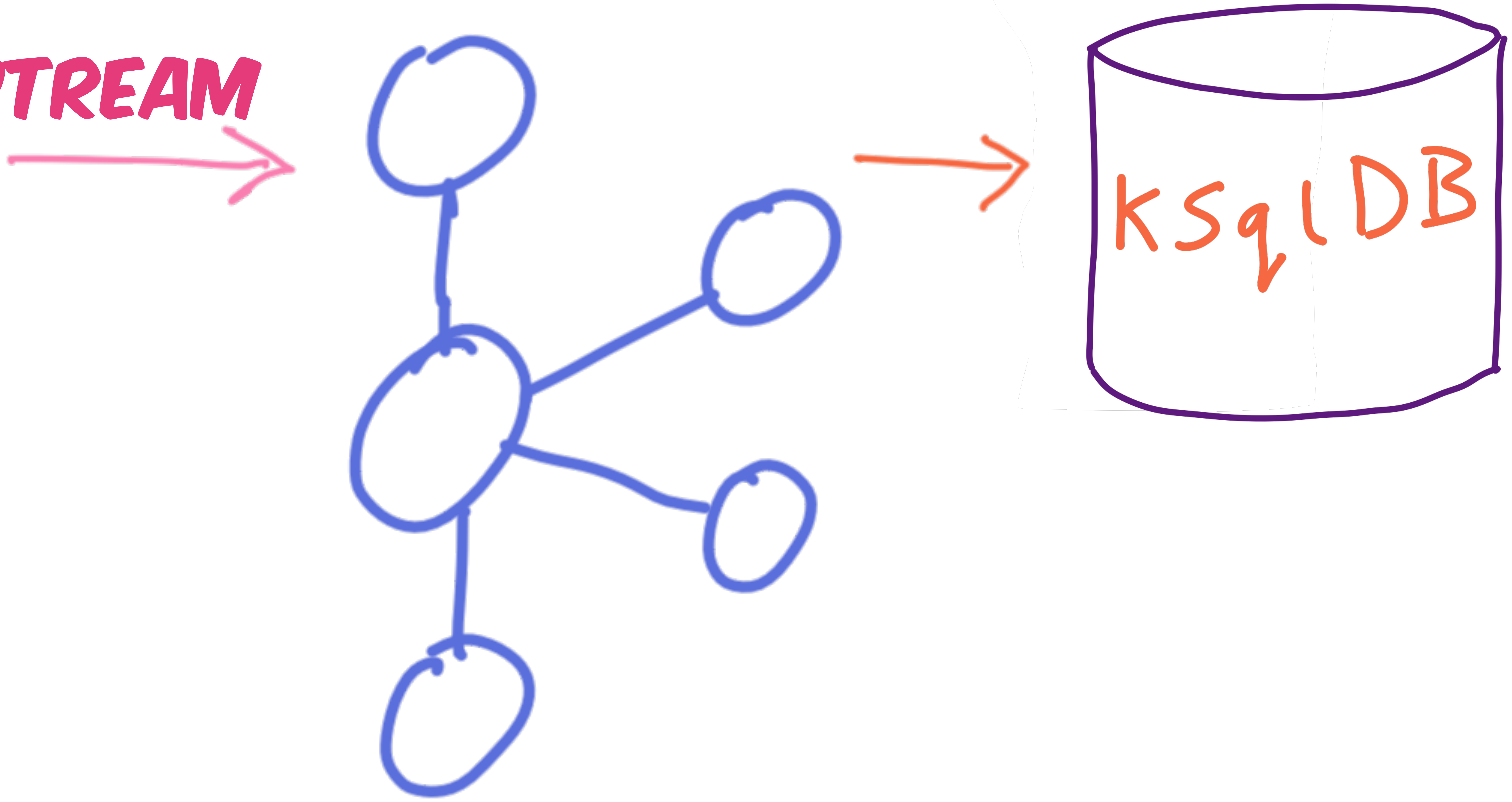


THE EVENT STREAMING DATABASE PURPOSE-BUILT FOR STREAM PROCESSING APPLICATIONS.

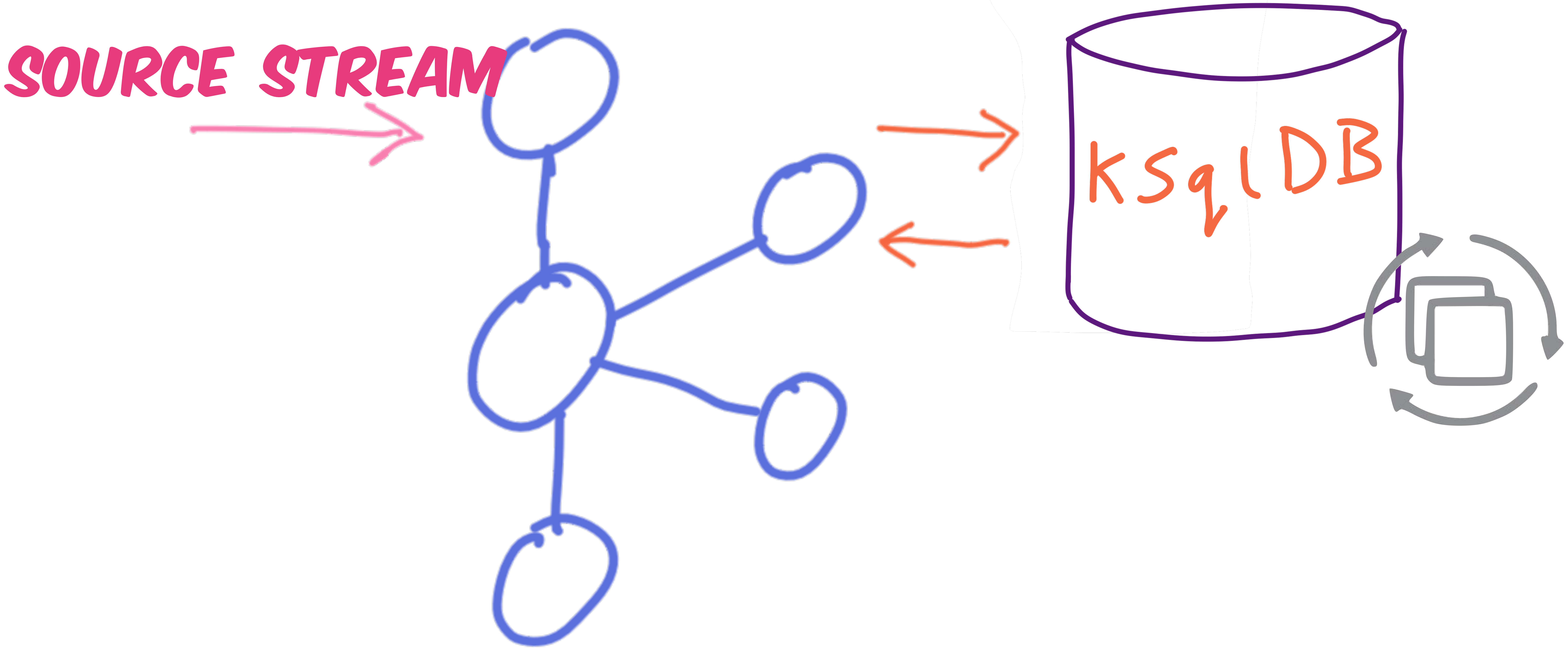
STREAM PROCESSING WITH KSQLDB

CONFLUENT

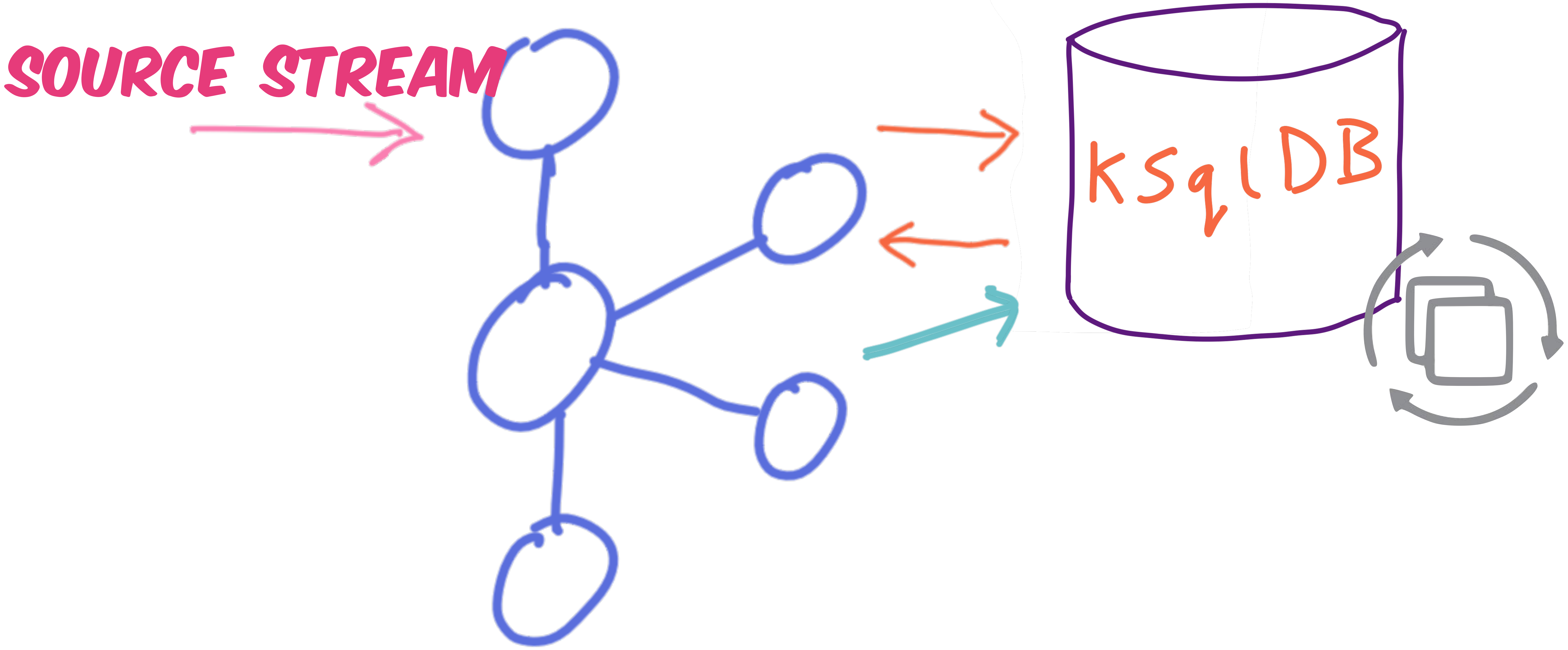
SOURCE STREAM



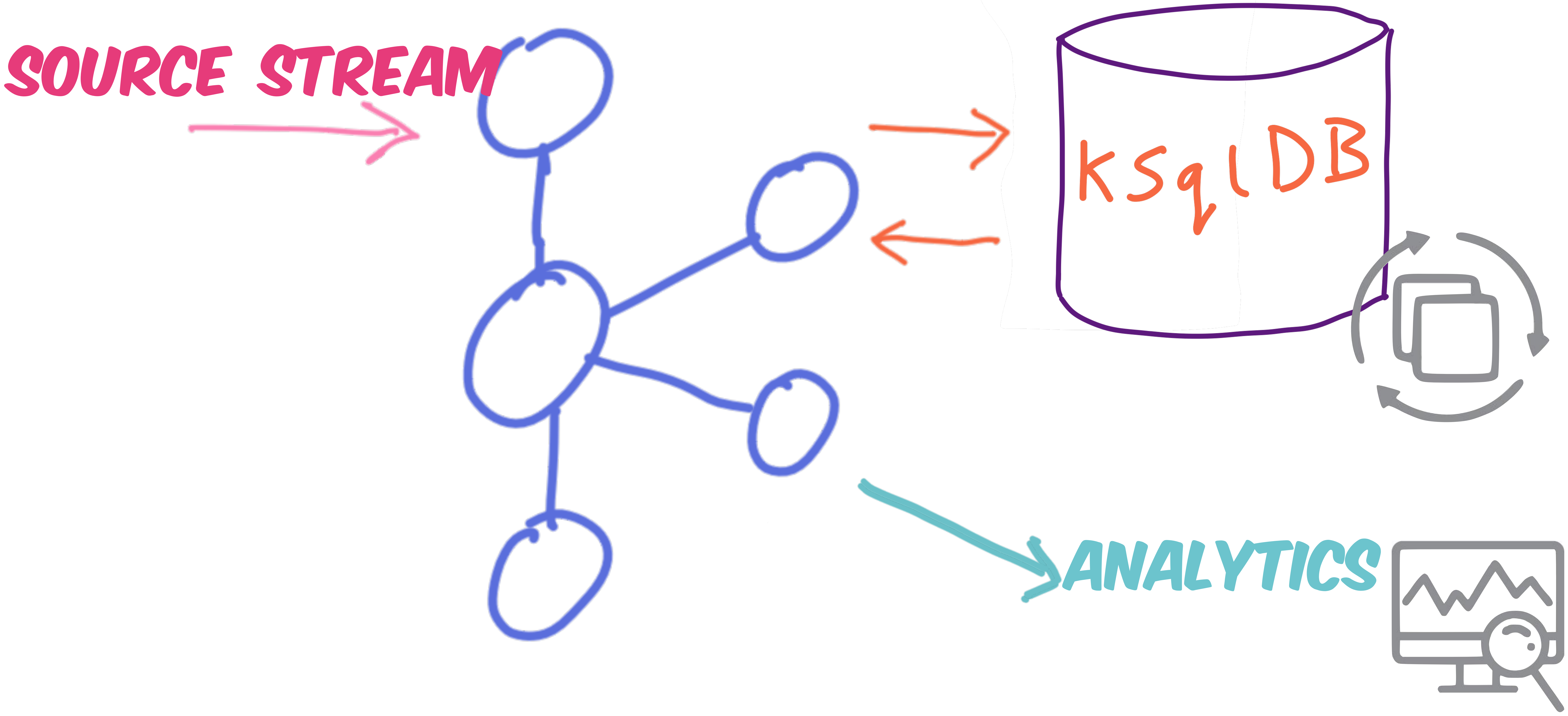
STREAM PROCESSING WITH KSQLDB



STREAM PROCESSING WITH KSQLDB

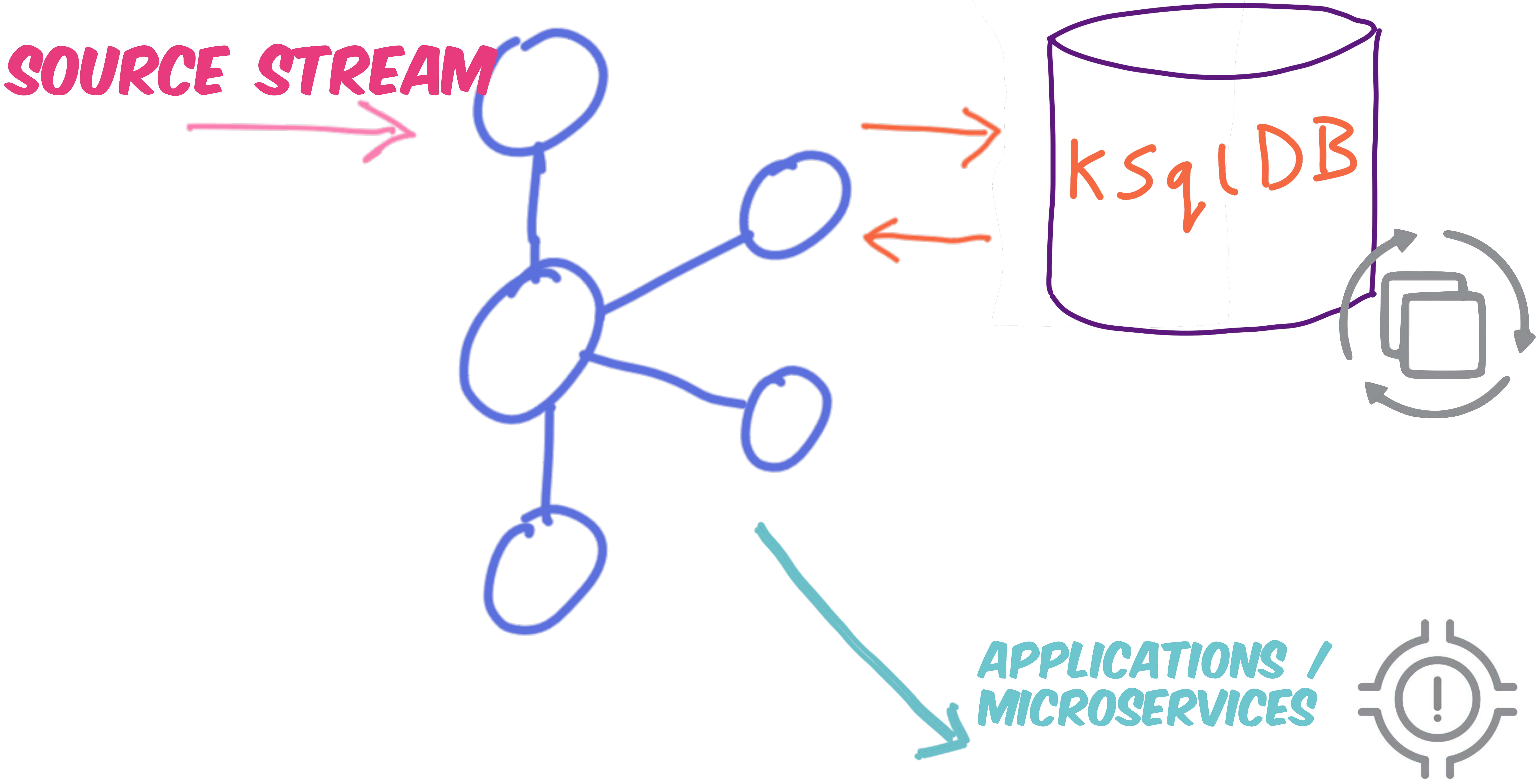


STREAM PROCESSING WITH KSQLDB



STREAM PROCESSING WITH KSQLDB

CONFLUENT



STREAM PROCESSING WITH KSQLDB

CONFLUENT

CONFLUENT

DEMO

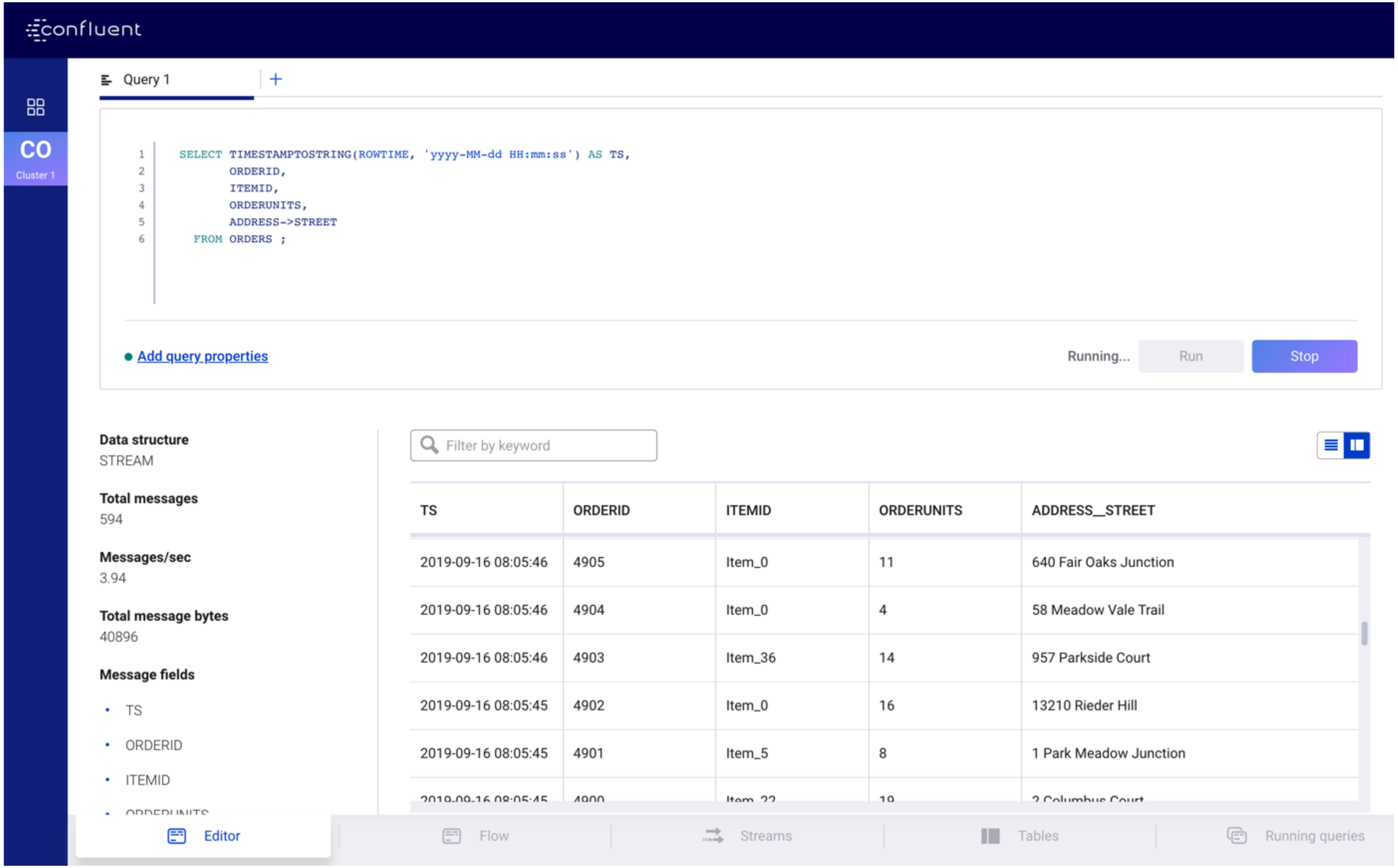
https://gamov.dev/scala-love-demo

INTERACTING WITH KSQLDB

Photo by Tim Mossholder on Unsplash

KSQLDB - CONFLUENT CONTROL CENTER

CONFLUENT



@GAMUSSA | #SCALALOVE | @CONFLUENTINC

KSQLDB - REST API

CONFLUENT

```
$ echo '{"ksql":"SELECT ORDERID, ITEMID, ADDRESS FROM ORDERS LIMIT 5;", "streamsProperties":
        "ksql.streams.auto.offset.reset": "earliest"}}' | \
  http http://localhost:8088/query
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Type: application/json
Date: Tue, 02 Jul 2019 12:46:25 GMT
Server: Jetty(9.4.14.v20181114)
Transfer-Encoding: chunked
Vary: Accept-Encoding, User-Agent
```


```
{"row":{"columns":[0,"Item_0",{"STREET":"377 Maryland Place","CITY":"Beaumont","STATE":"Texas
l,"terminal":false}
{"row":{"columns":[0,"Item_0",{"STREET":"072 Butternut Lane","CITY":"Grand Junction","STATE":
sage":null,"terminal":false}
{"row":{"columns":[1,"Item_0",{"STREET":"703 Hoffman Place","CITY":"Mountain View","STATE":"C
sage":null,"terminal":false}
{"row":{"columns":[2,"Item_0",{"STREET":"0 Dorton Circle","CITY":"Brooklyn","STATE":"New York
l,"terminal":false}
{"row":{"columns":[3,"Item_0",{"STREET":"404 Mayer Park","CITY":"Lubbock","STATE":"Texas"}]},
rminal":false}
{"row":null,"errorMessage":null,"finalMessage":"Limit Reached","terminal":true}
```

KSQLDB - NATIVE CLIENT (COMING SOON)

CONFLUENT

Branch: master ▾ ksql / design-proposals / klip-15-new-api-and-client.md Find file Copy path

purplefox docs: Bring KLIP-15 up to date (#4499) ddf0701 7 hours ago

1 contributor

331 lines (235 sloc) | 14.9 KB Raw Blame History

KLIP 15 - ksqlDB Client and New Server API

Author: Tim Fox (purplefox) | **Release Target**: ? | **Status**: *In Discussion* | **Discussion**:

Please read the initial discussion on the ksqlDB developer group first:

https://groups.google.com/forum/#!topic/ksql-dev/yxcRlsOsNmo

And there is a partially working prototype for this work described here:

https://groups.google.com/forum/#!topic/ksql-dev/5mLKvtZFs4Y

For ksqlDB to be a successful project we need to provide an awesome out of the box developer experience and it should be super easy and fun to write powerful event streaming applications. Currently this is somewhat difficult to do, and it requires the use of multiple clients and multiple moving parts in order to build a simple event sourcing / CQRS style application. We should provide a new ksqlDB client that provides that currently missing single interaction point with the event streaming platform so that we can improve the application development process. To support the client we should provide an updated HTTP2 based API that allows streaming use cases to be handled better.


WHAT ELSE CAN KSQLDB DO?
Photo by Sereja Ris on Unsplash

MESSAGE TRANSFORMATION WITH KSQLDB

CONFLUENT

ORDERS

```
{ "ordertime": 1560070133853,
  "orderid": 7,
  "itemid": "Item_9",
  "orderunits": 5,
  "address": {
    "street": "243 Utah Way",
    "city": "Orange",
    "state": "California"
  }
}
```

CONVERT THIS TIMESTAMP TO HUMAN-READABLE FORMAT

DROP THESE ADDRESS FIELDS

MESSAGE TRANSFORMATION WITH KSQLDB

CONFLUENT

```
{ "ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5,
  "address": {
    "street": "243 Utah Way",
    "city": "Orange",
    "state": "California"
```

ORDERS

KSQLDB

```
CREATE STREAM ORDERS_NO_ADDRESS_DATA AS
 SELECT ORDERTIME, ORDERID, ITEMID, ORDERUNITS
  FROM ORDERS;
```

MESSAGE TRANSFORMATION WITH KSQLDB

CONFLUENT

ORDERS


```
{ "ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5,
  "address": {
    "street": "243 Utah Way",
    "city": "Orange",
```


KSQLDB

```
CREATE STREAM ORDERS_NO_ADDRESS_DATA AS
  SELECT TIMESTAMPTOSTRING(ROWTIME, 'yyyy-MM-dd HH:mm:ss')
         AS ORDER_TIMESTAMP,
       ORDERID, ITEMID, ORDERUNITS
   FROM ORDERS;
```

ORDERS_NO_ADDRESS_DATA


```
{ "order_ts": "2020-02-14 15:10:58",
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5
}
```

LOOKUPS AND JOINS WITH KSQLDB

CONFLUENT

ORDERS

{"ordertime": 1560070133853,
"orderid": 67,
"itemid": "Item_9",
"orderunits": 5}

CONFLUENT

LOOKUPS AND JOINS WITH KSQLDB

Lookup

ITEMS

ORDERS

{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}

LOOKUPS AND JOINS WITH KSQLDB

CONFLUENT

ITEMS

ORDERS

```sql
CREATE STREAM ORDERS_ENRICHED
AS
SELECT O.*, I.*,
 O.ORDERUNITS * I.UNIT_COST
  AS TOTAL_ORDER_VALUE,
 FROM ORDERS O
   INNER JOIN ITEMS I
   ON O.ITEMID = I.ID ;
```


```json
{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}
```

LOOKUPS AND JOINS WITH KSQLDB

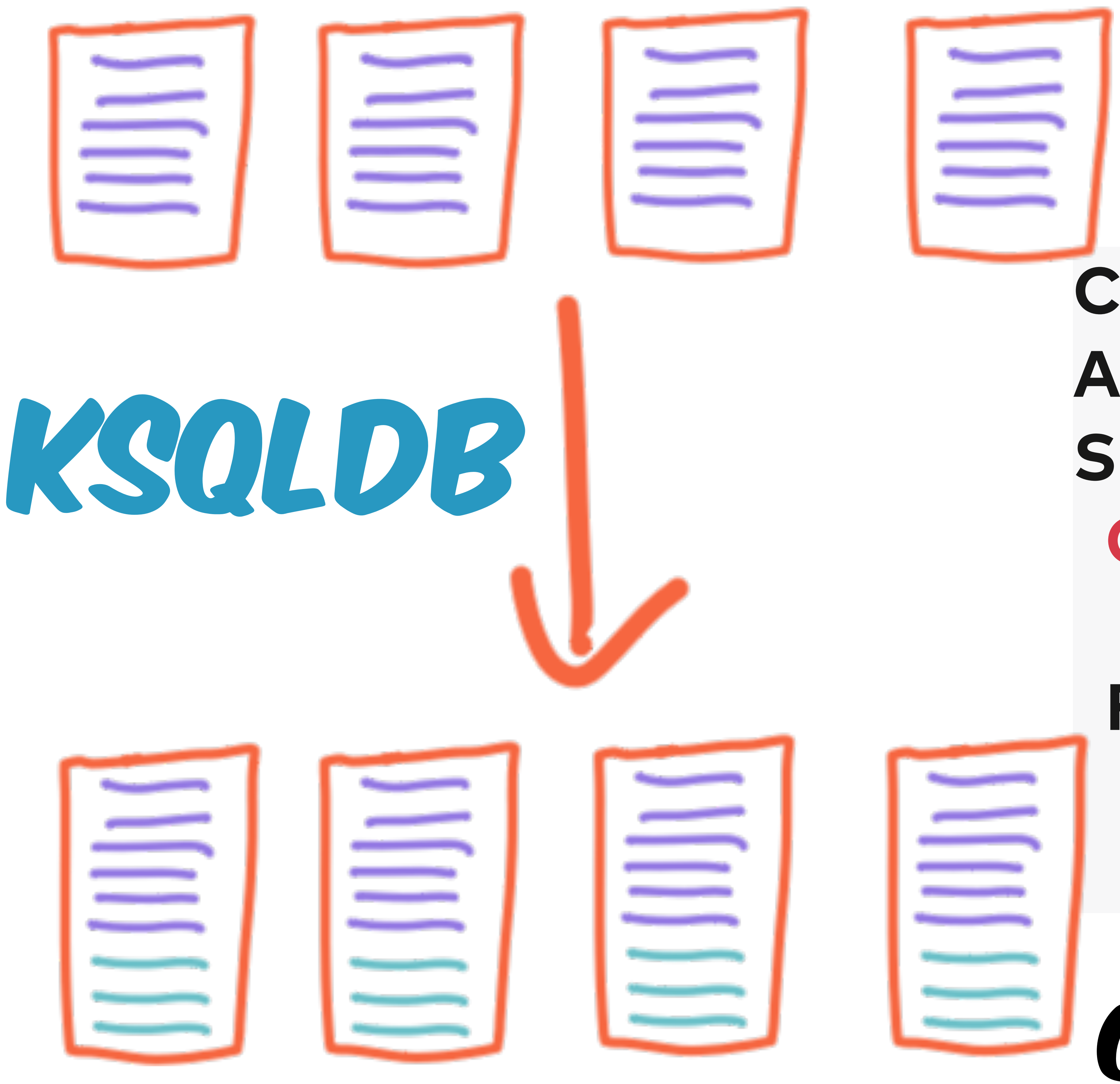
CONFLUENT

Lookup

ITEMS


```
{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
{"ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5}
```


ORDERS




```
CREATE STREAM ORDERS_ENRICH
AS
SELECT O.*, I.*,
  O.ORDERUNITS * I.UNIT_COST
   AS TOTAL_ORDER_VALUE,
 FROM ORDERS O
    INNER JOIN ITEMS I
    ON O.ITEMID = I.ID ;
```

ORDERS_ENRICHED

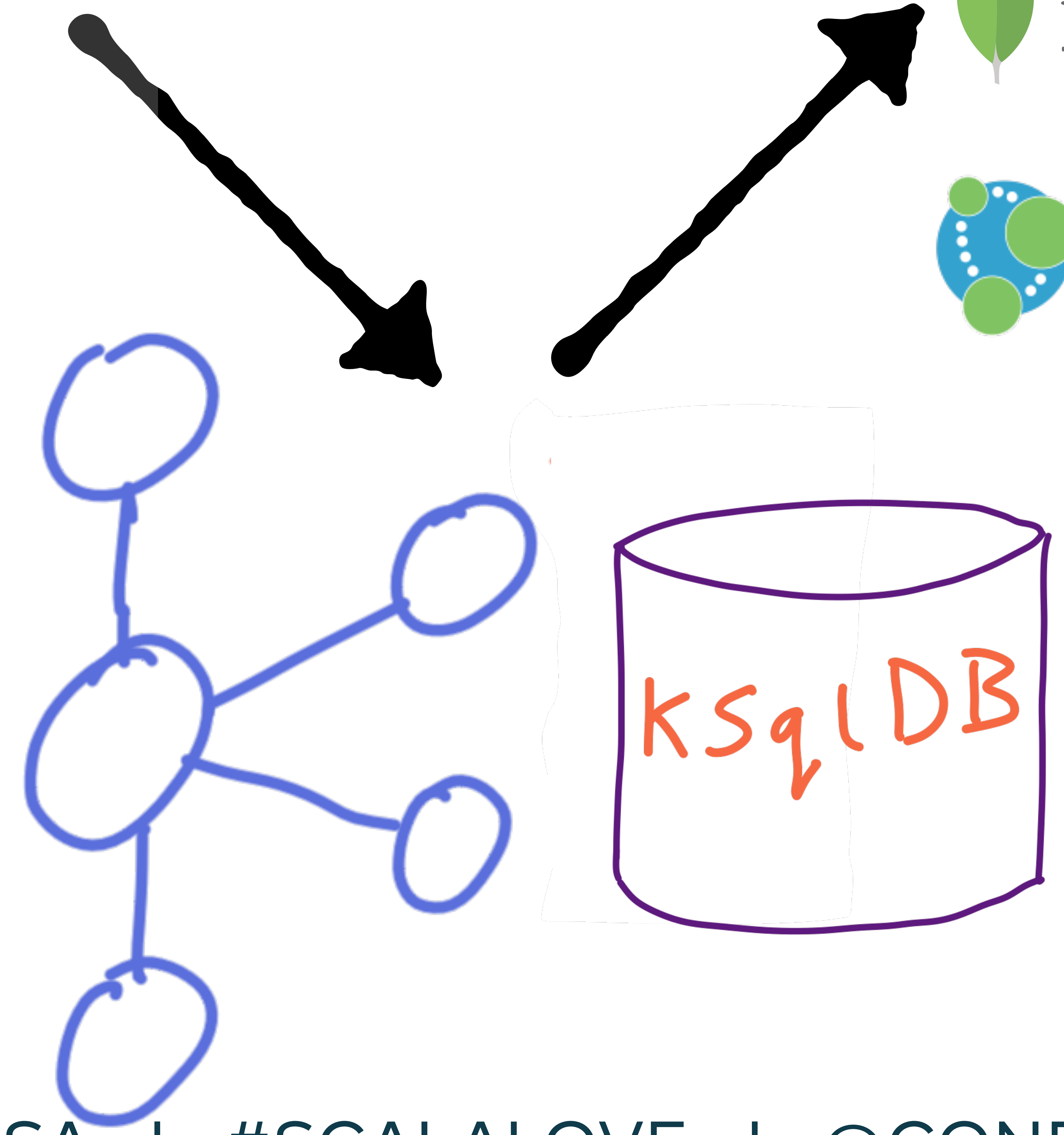

```
{
  "ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5,
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
  "unit_cost": 19.9,
  "total_order_value": 99.5
}
```

CONNECTING KSQLDB TO OTHER SYSTEMS

CONNECTING KSQLDB TO OTHER SYSTEMS

CONFLUENT

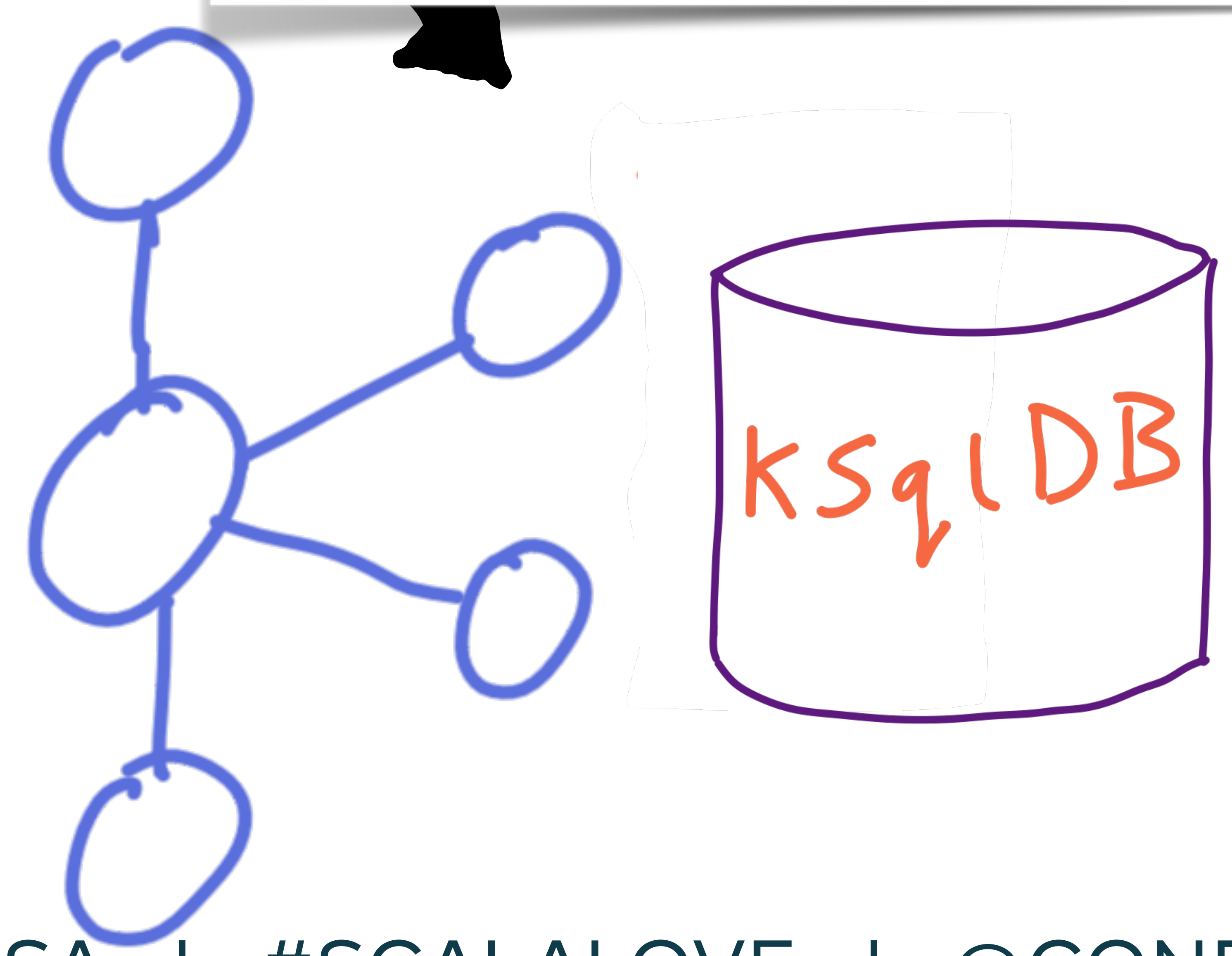
JSON
MQTT.ORG
MySQL
ORACLE
syslog
CSV
salesforce
http://
Microsoft SQL Server



Java JDBC
salesforce
snowflake
influxdb
IBM MQ
Google BigQuery
mongoDB
elasticsearch
http://
hadoop HDFS
neo4j
splunk>
MQTT.ORG
Amazon S3

CONNECTING KSQLDB TO OTHER SYSTEMS

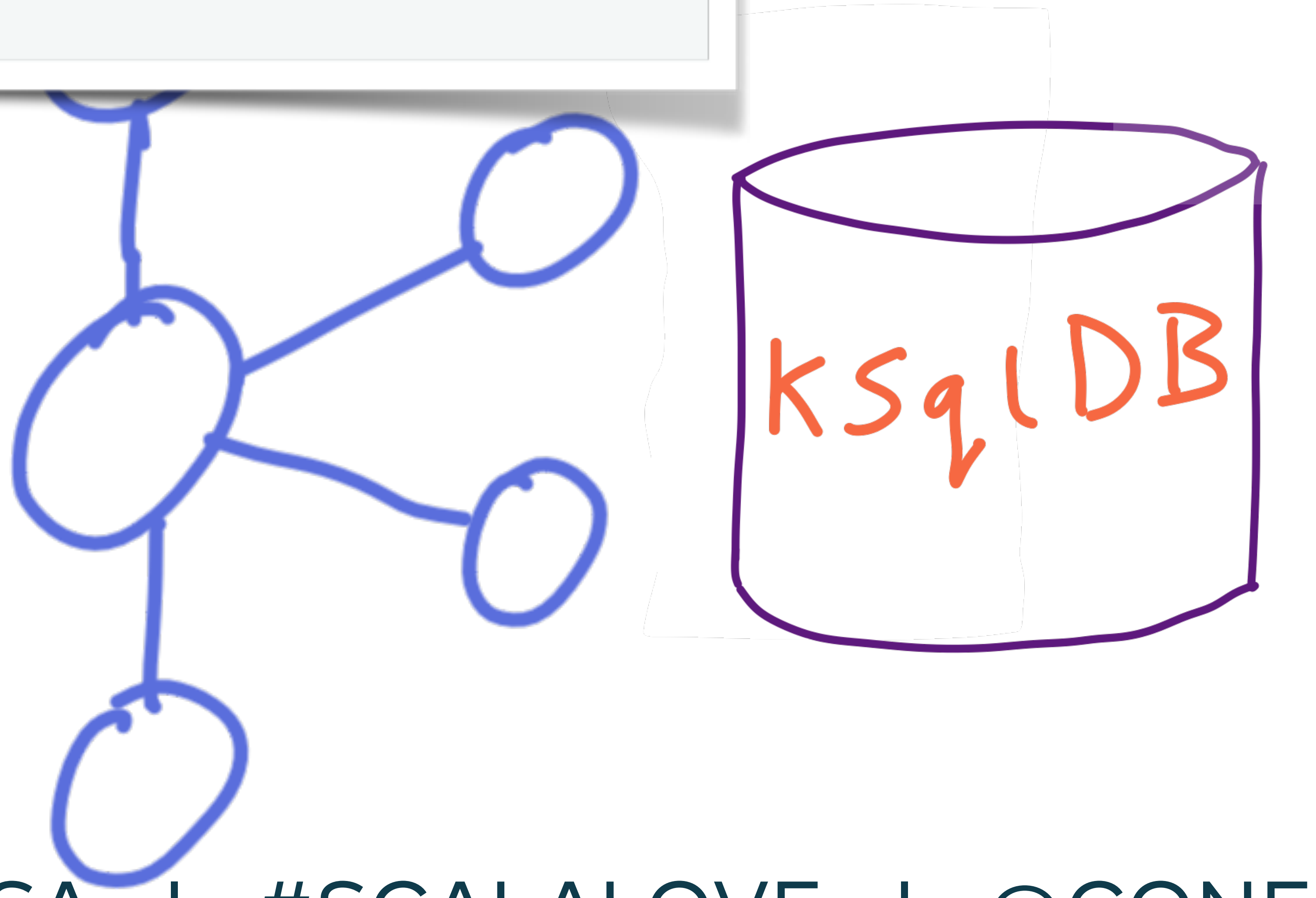
```sql
CREATE SOURCE CONNECTOR
SOURCE_MYSQL_01 WITH (
  'connector.class' =
'i.d.c.mysql.MySqlConnector',
  'database.hostname' = 'mysql',
  'table.whitelist' = 'demo.customers');
```



CONNECTING KSQLDB TO OTHER SYSTEMS

CONFLUENT

```sql
CREATE SINK CONNECTOR
SINK_ELASTIC_01 WITH (
 'connector.class' =
'…ElasticsearchSinkConnector',
 'connection.url' =
'http://elasticsearch:9200',
 'topics' = 'orders');
```



CONFLUENT

STREAMS & TABLES

@GAMUSSA | #SCALALOVE | @CONFLUENTINC

STREAMS AND TABLES

KAFKA TOPIC (K/V BYTES)

```
{ "event_ts": "2020-02-17T15:22:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
{ "event_ts": "2020-02-17T17:23:00Z",
  "person"  : "robin",
  "location": "London"
}
```

```
{ "event_ts": "2020-02-17T22:23:00Z",
  "person"  : "robin",
  "location": "Wakefield"
}
```

```
{ "event_ts": "2020-02-18T09:00:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

KSQLDB STREAM

EVENT_TS	PERSON	LOCATION
2020-02-17 15:22:00	robin	Leeds
2020-02-17 17:23:00	robin	London
2020-02-17 22:23:00	robin	Wakefield
2020-02-18 09:00:00	robin	Leeds

STREAM: TOPIC + SCHEMA

KSQLDB TABLE

PERSON	LOCATION
robin	Leeds

TABLE: STATE FOR GIVEN KEY TOPIC + SCHEMA

CONFLUENT

STATEFUL AGGREGATIONS IN KSQLDB

CONFLUENT

KAFKA TOPIC

```
{ "event_ts": "2020-02-17T15:22:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
{ "event_ts": "2020-02-17T17:23:00Z",
  "person"  : "robin",
  "location": "London"
}
```

```
{ "event_ts": "2020-02-17T22:23:00Z",
  "person"  : "robin",
  "location": "Wakefield"
}
```

```
{ "event_ts": "2020-02-18T09:00:00Z",
  "person"  : "robin",
  "location": "Leeds"
}
```

```
SELECT PERSON,
       COUNT(*)
FROM   MOVEMENTS
GROUP BY PERSON;
```

```
+-------+------------------+
|PERSON | LOCATION_CHANGES |
+-------+------------------+
|robin  | 4                |
```

```
SELECT PERSON,
       COUNT_DISTINCT(LOCATION)
FROM   MOVEMENTS
GROUP BY PERSON;
```

```
+-------+------------------+
|PERSON | UNIQUE_LOCATIONS |
+-------+------------------+
|robin  | 3                |
```

AGGREGATIONS CAN BE ACROSS THE ENTIRE INPUT, OR WINDOWED (TUMBLING, HOPPING, SESSION)

PULL AND PUSH QUERIES IN KSQLDB

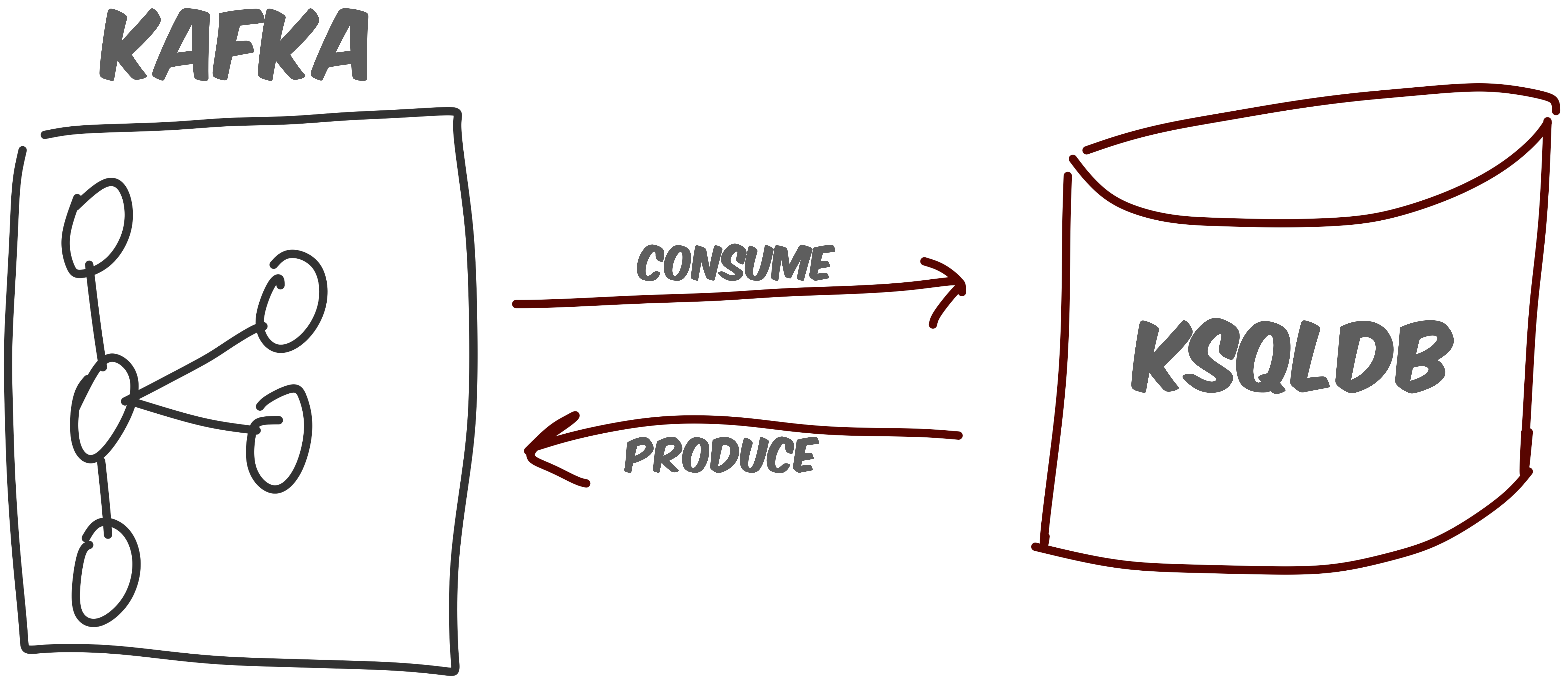
CONFLUENT

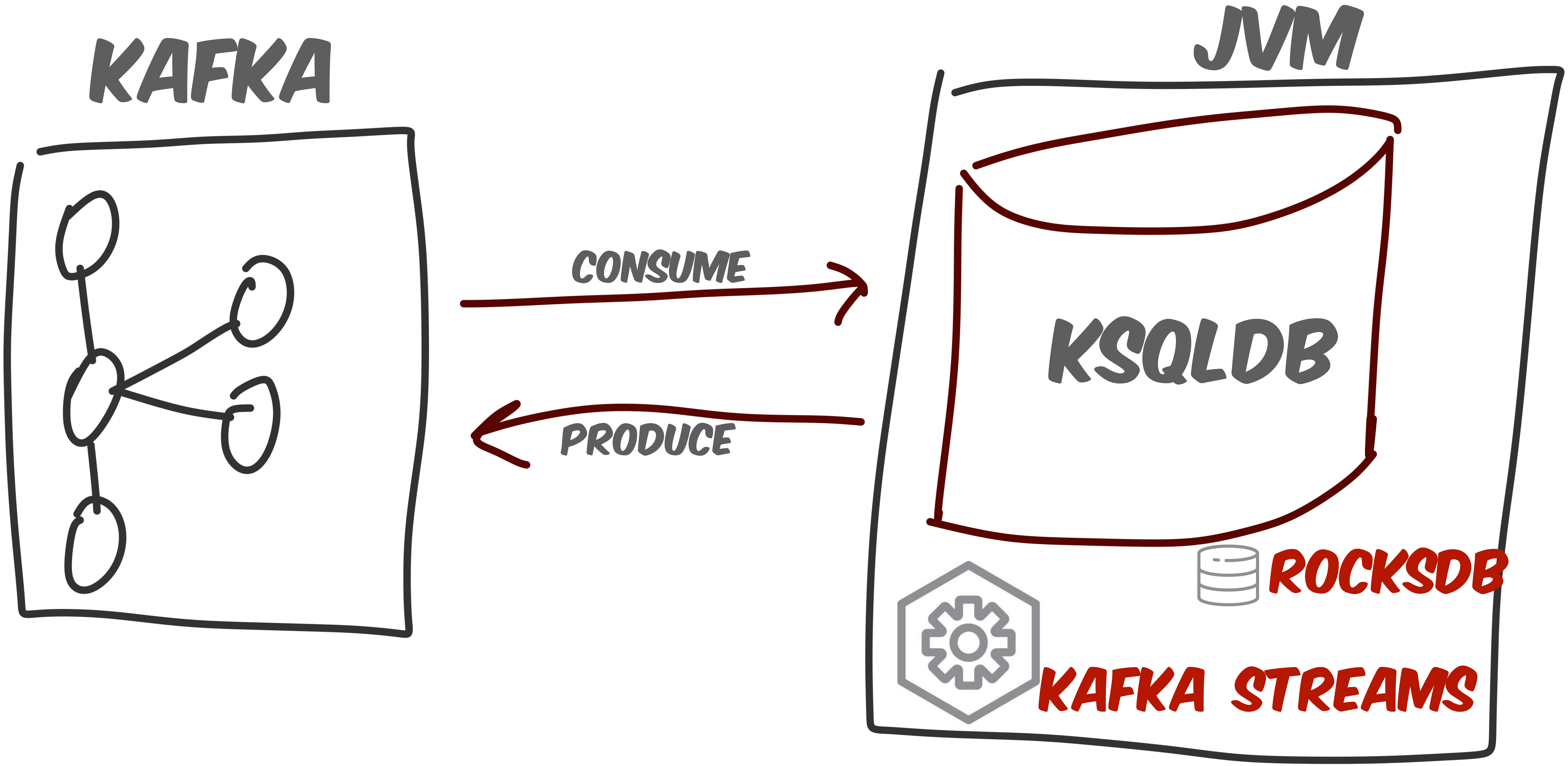
PULL QUERY	PUSH QUERY
	
TELLS YOU: POINT IN TIME VALUE	ALL VALUE CHANGES
EXITS: IMMEDIATELY	NEVER

UNDER THE COVERS OF KSQLDB

Photo by Vinicius de Moraes on Unsplas

CONFLUENT




KAFKA
CONSUME
PRODUCE
JVM
KSQLDB
ROCKSDB
KAFKA STREAMS


WHERE CAN YOU RUN KSQLDB?
ANYWHERE!
imgflip.com

confluent cloud

Fully Managed Kafka & KSQL
as a Service

RUNNING KSQLDB - SELF-MANAGED

CONFLUENT

ksqlDB Server
(JVM process)

DEB, RPM, ZIP, TAR downloads
http://confluent.io/download

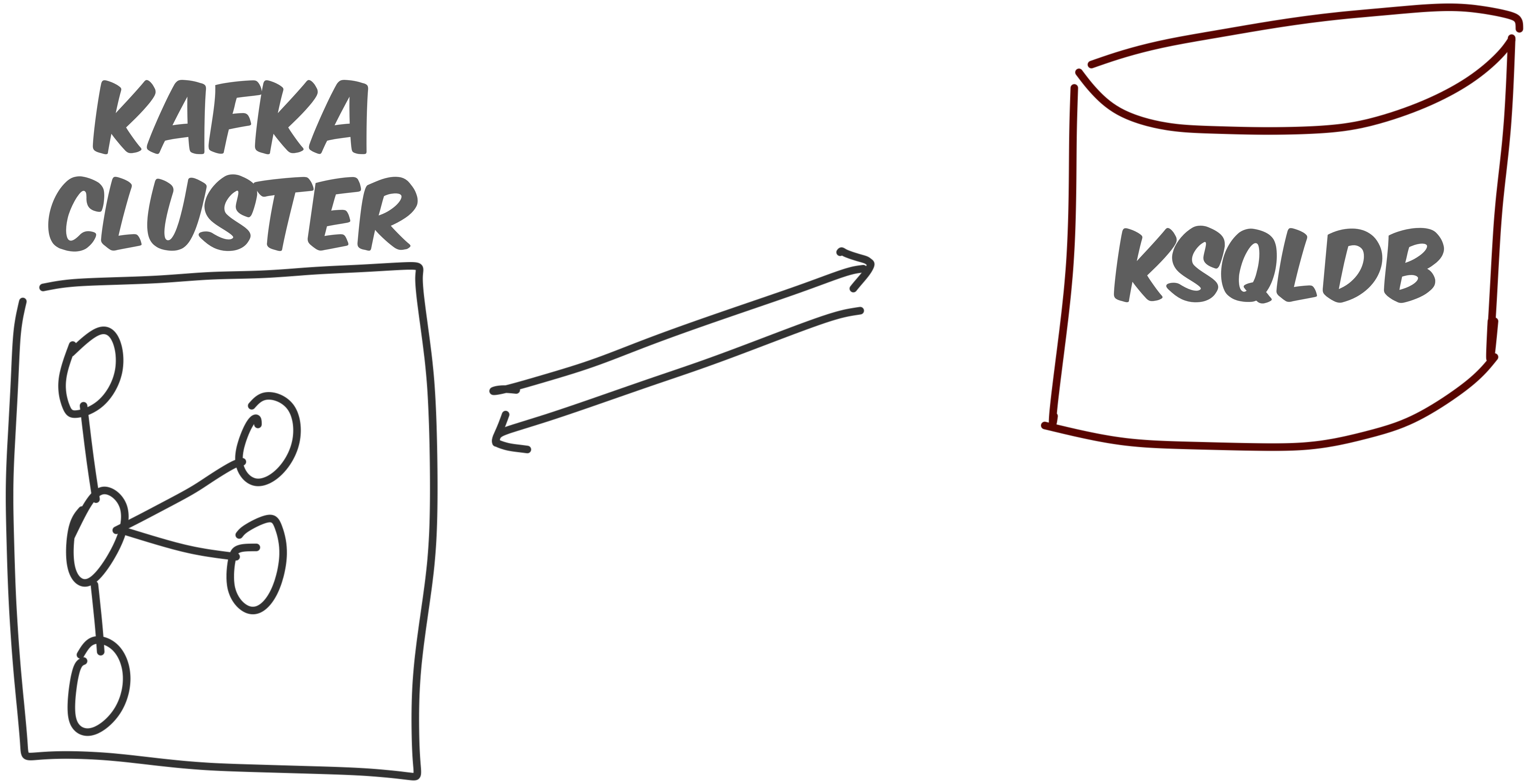
Docker images
confluentinc/ksqldb-server

Physical
docker
kubernetes
openstack
vmware
Microsoft Azure
Google Cloud Platform
amazon web services

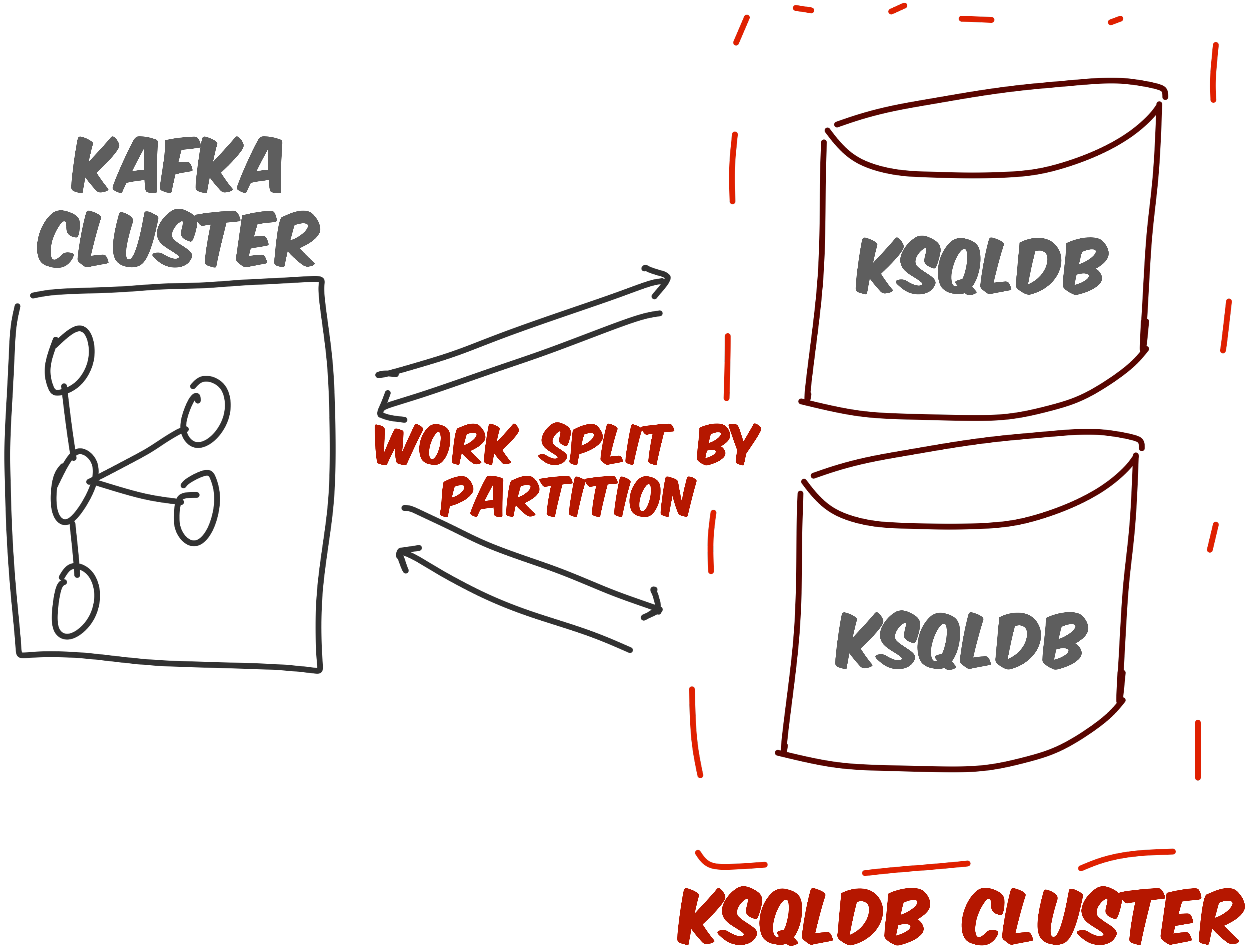
...and many more...

SCALING KSQLDB

CONFLUENT



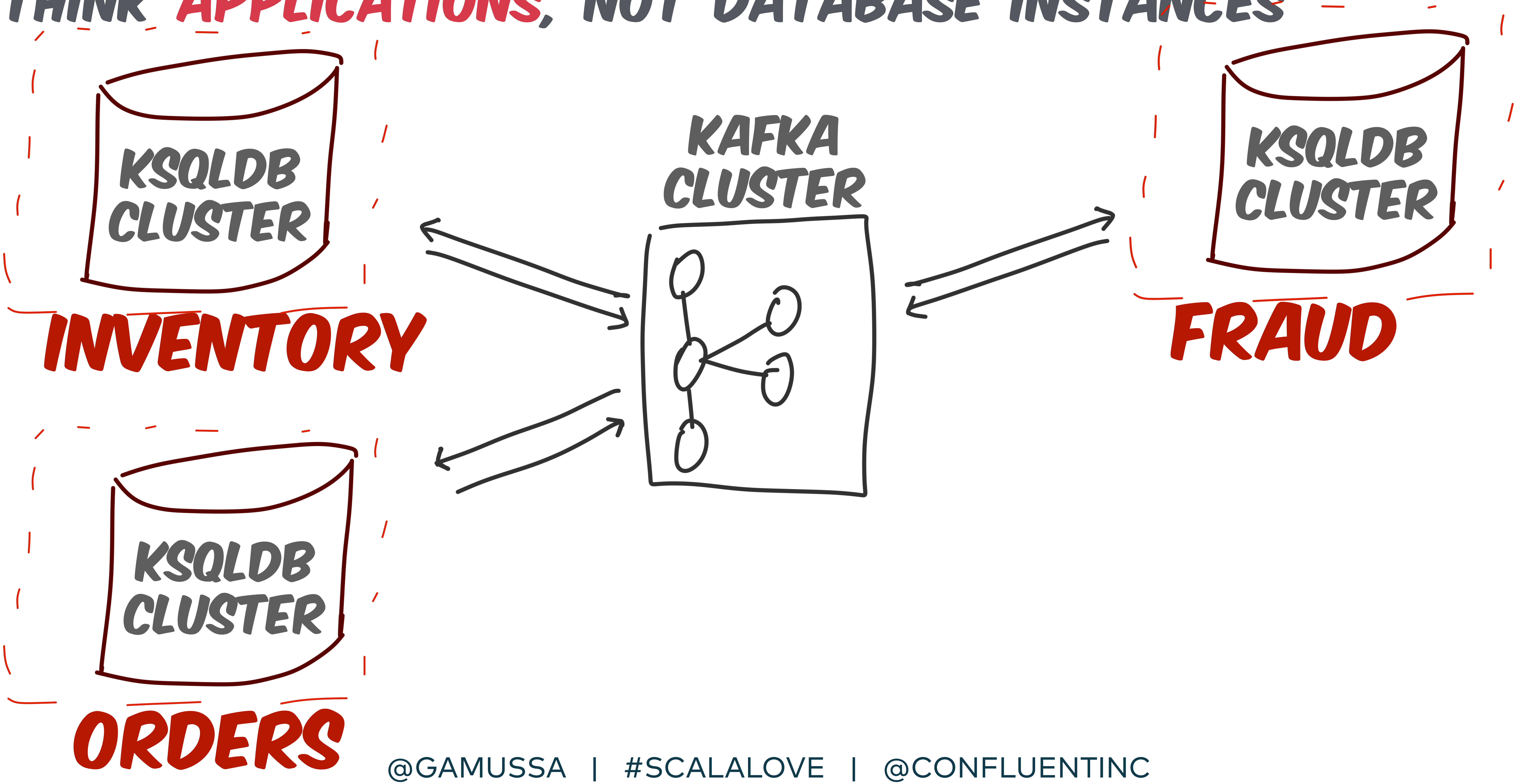
SCALING KSQLDB




ONE DOES NOT SIMPLY RUN
ONE SINGLE KSQLDB CLUSTER

THINK APPLICATIONS, NOT DATABASE INSTANCES

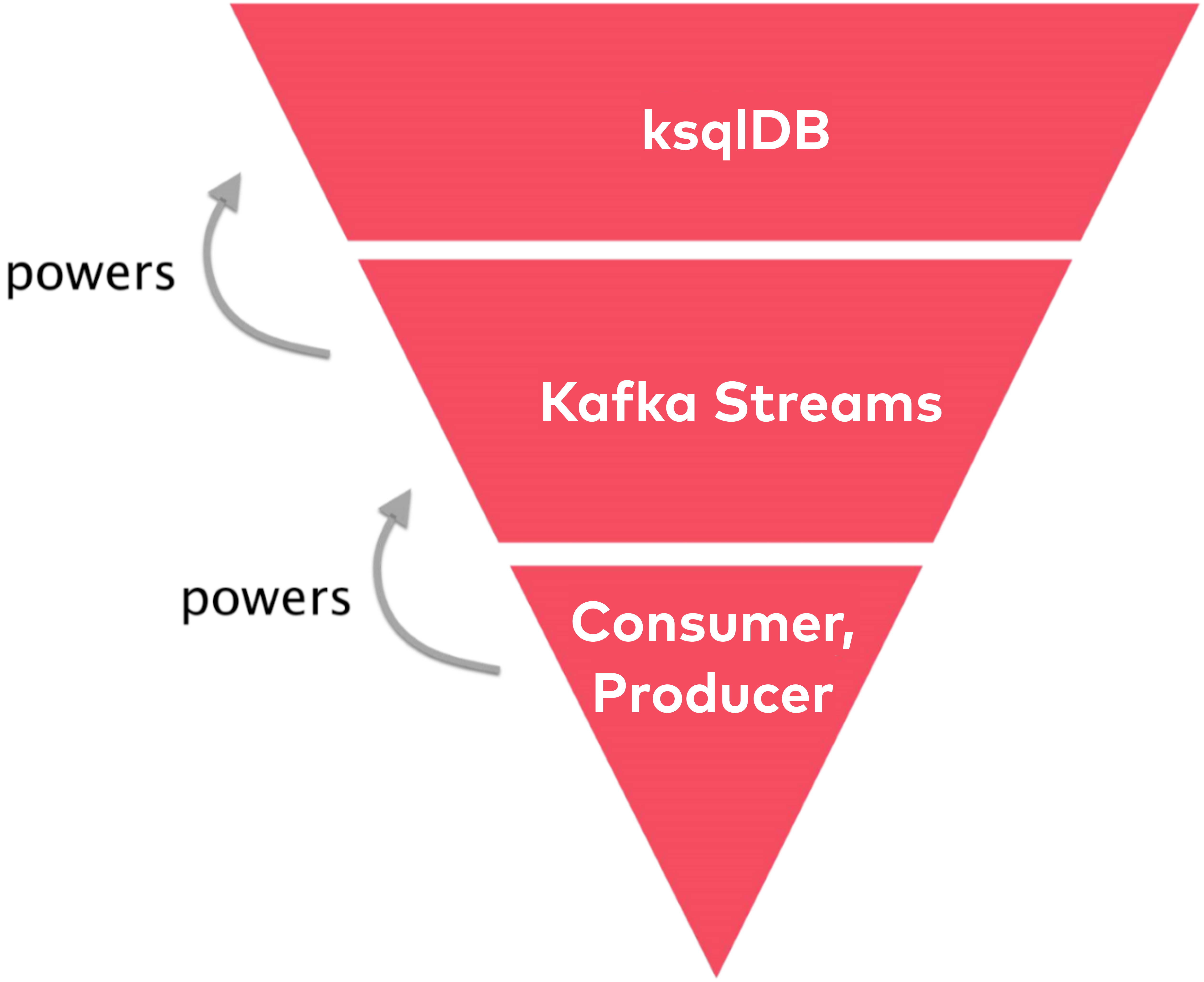
CONFLUENT



KSQLDB OR KAFKA STREAMS?

Photo by Ramiz Dedaković on Unsplash

KSQLDB BUILDS ON STREAMS



ease of use

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

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

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

flexibility

WANT TO LEARN MORE?

CTAS, NOT CATS
(SORRY, NOT SORRY)

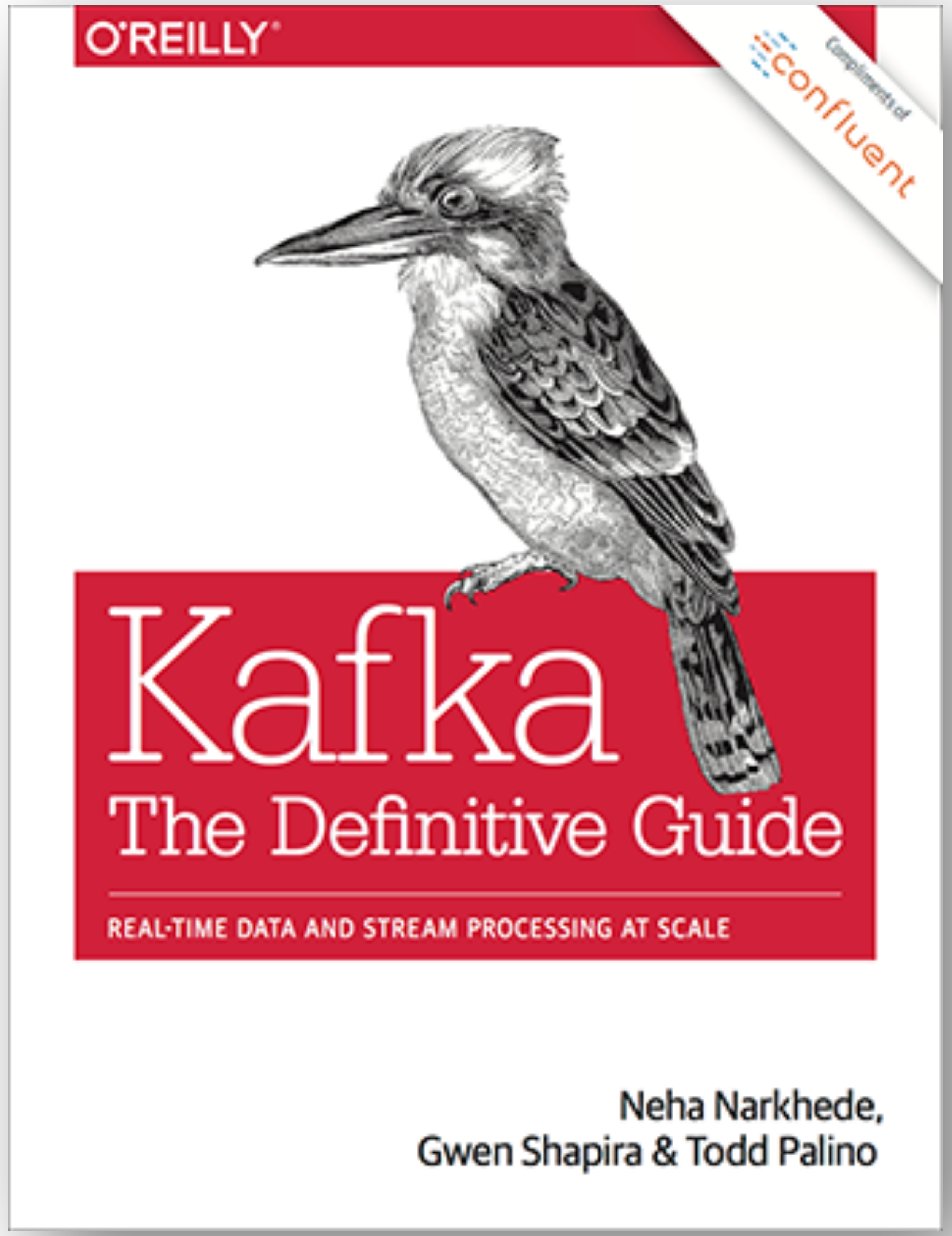

Learn Kafka.
Start building with Apache Kafka at Confluent Developer.
developer.confluent.io

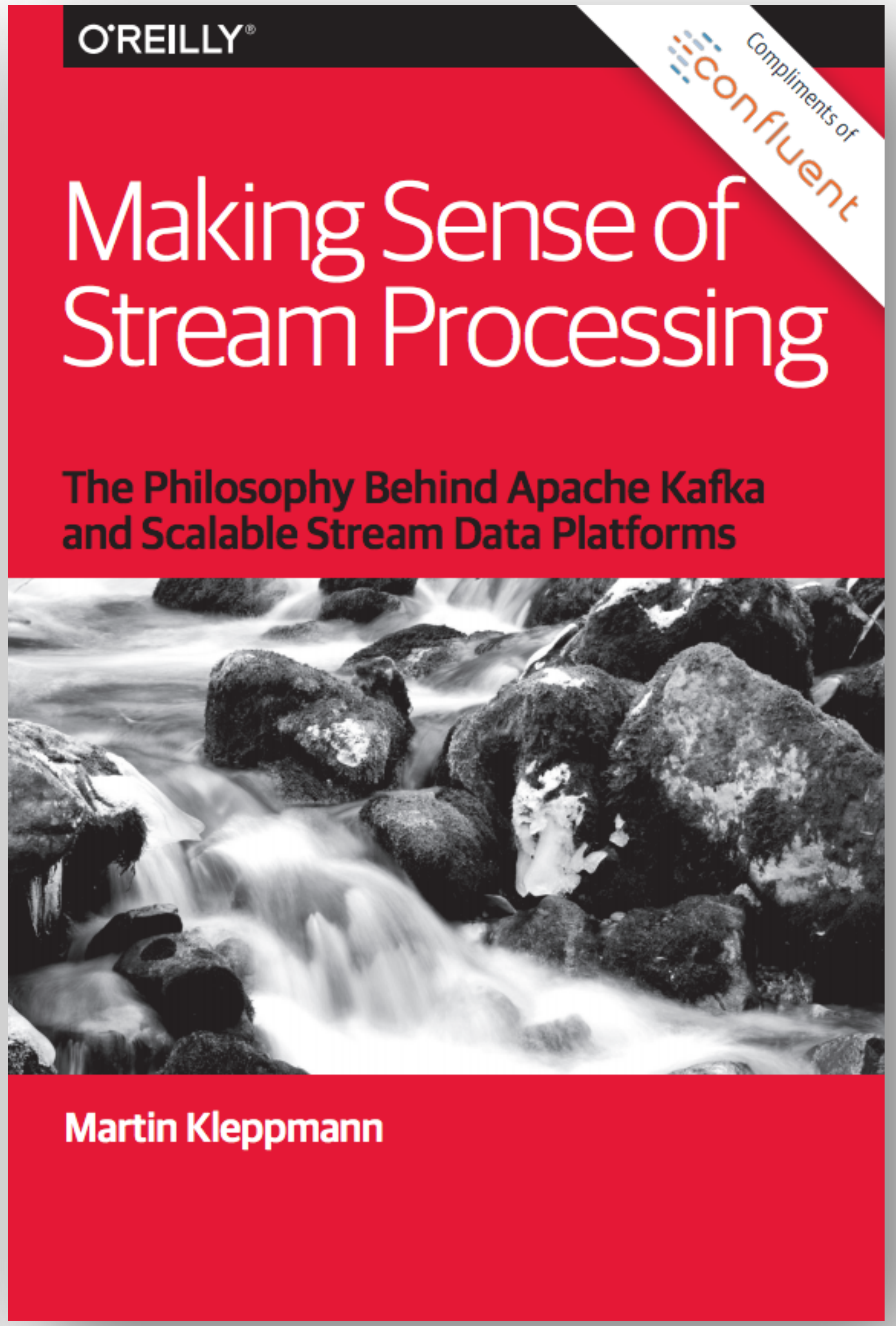
CONFLUENT COMMUNITY SLACK GROUP

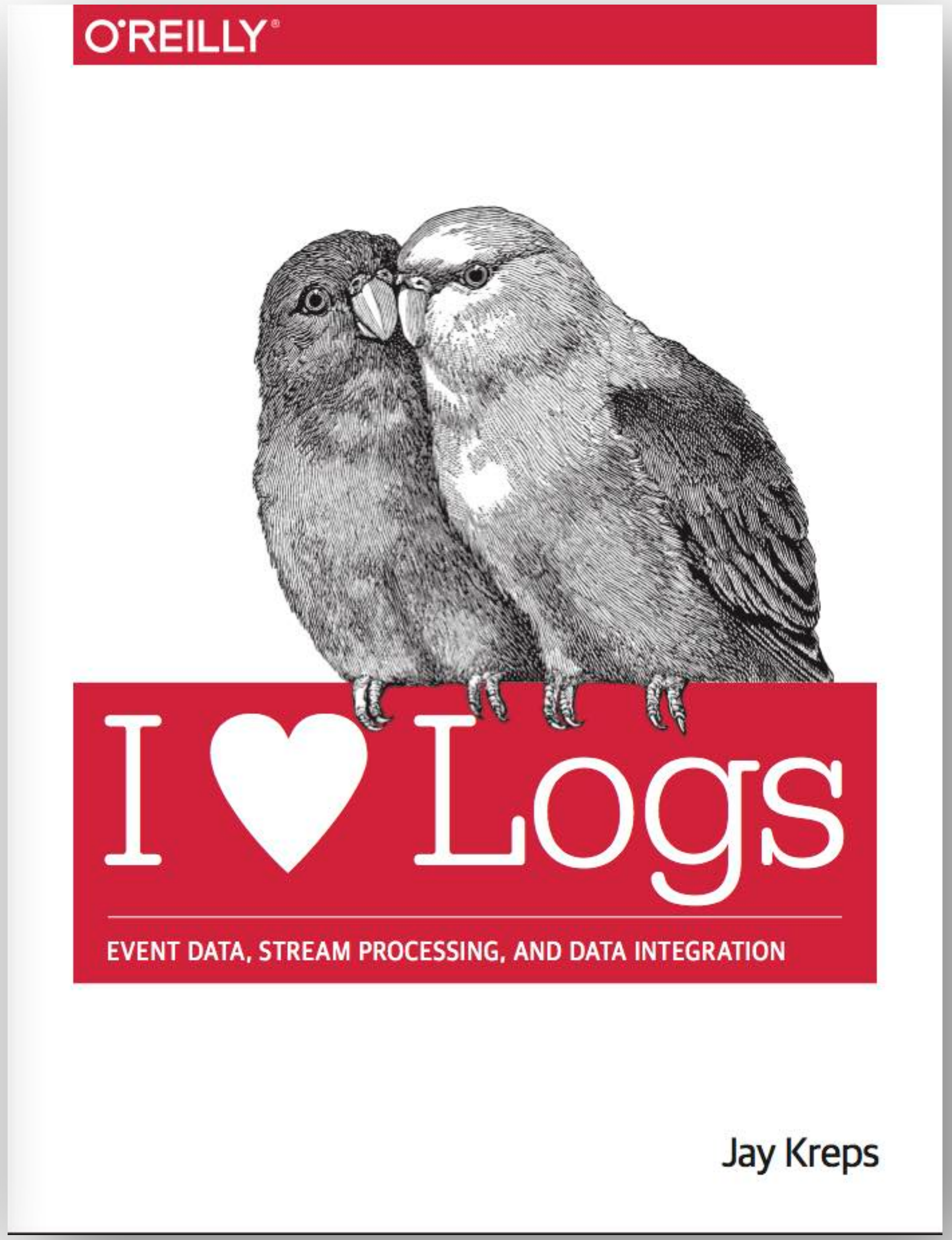
cnfl.io/slack

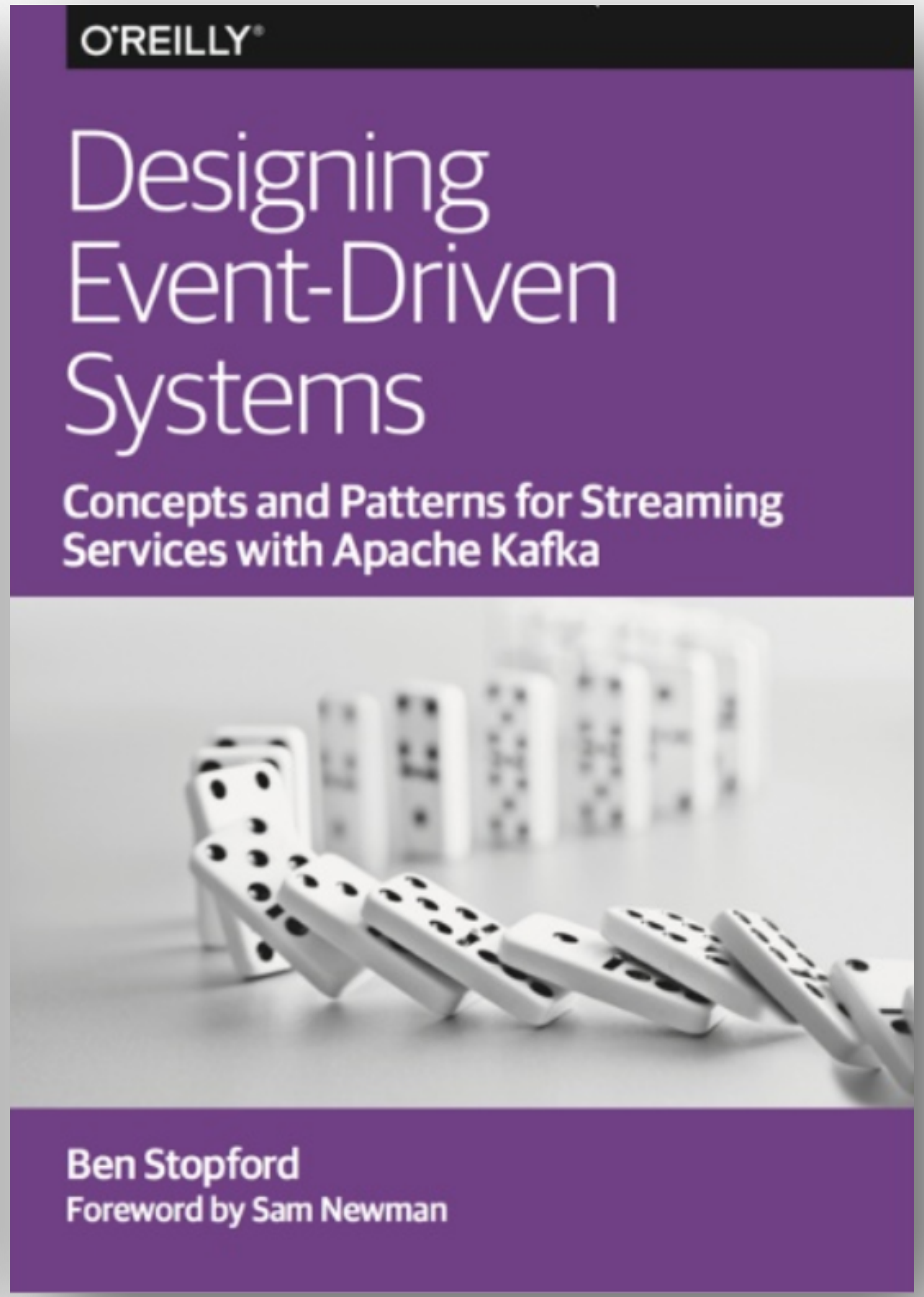
FREE BOOKS!

https://cnfl.io/book-bundle









CONFLUENT