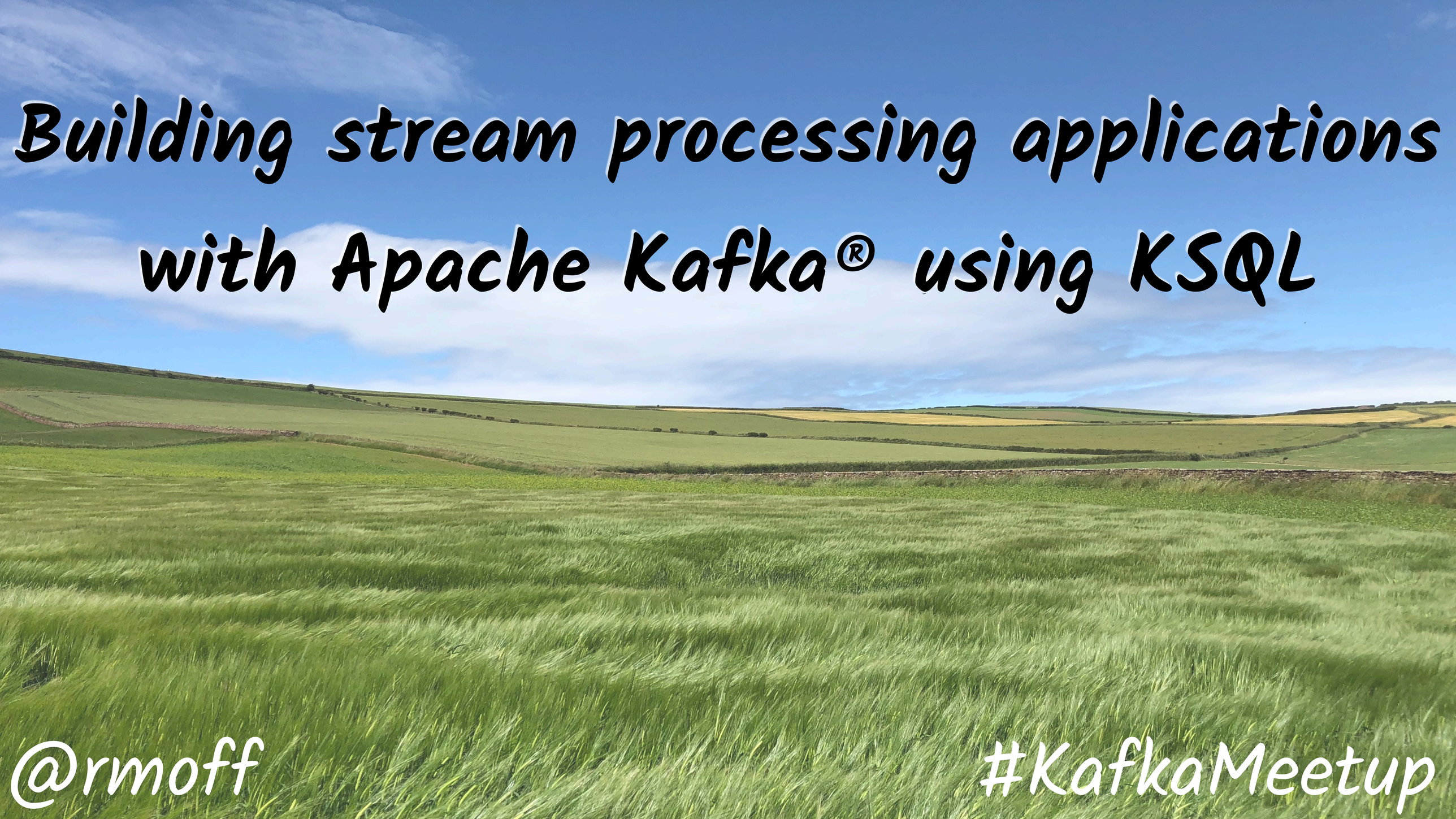

Building stream processing applications with Apache Kafka® using KSQL
@rmoff
#KafkaMeetup

STREAM PROCESSING

PROCESSING

STREAM

PROCESSING

a

STREAM

of EVENTS

STREAMS ARE

EVERYWHERE

A Customer Experience

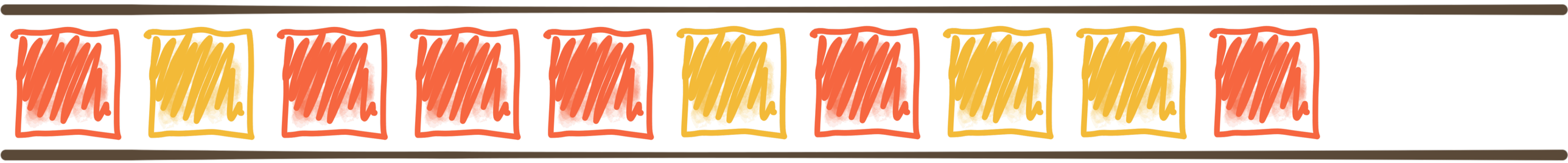
A Sale

A Sensor Reading

An Application Log Entry

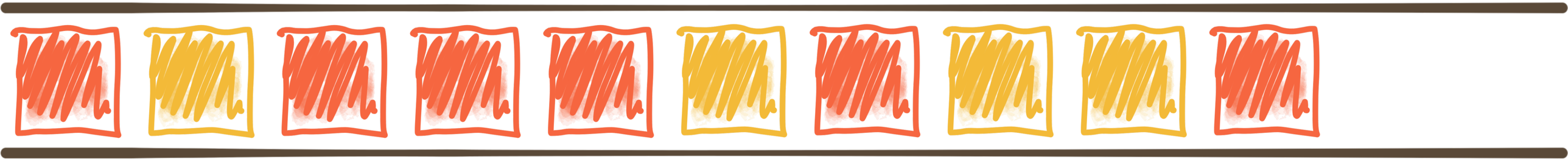
Databases

Streams of events



Time

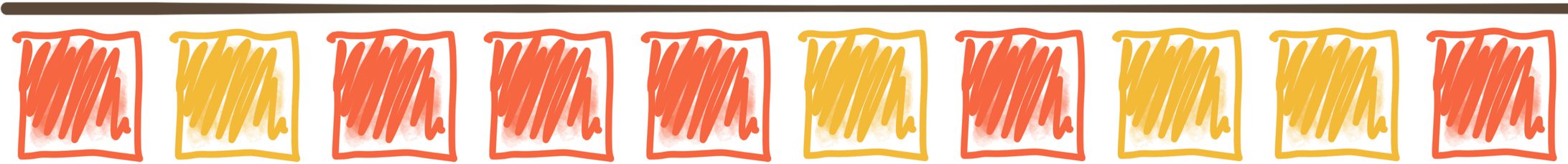
Stream Processing with KSQL



Stream: widgets

Stream: widgets_red

Stream Processing with KSQL



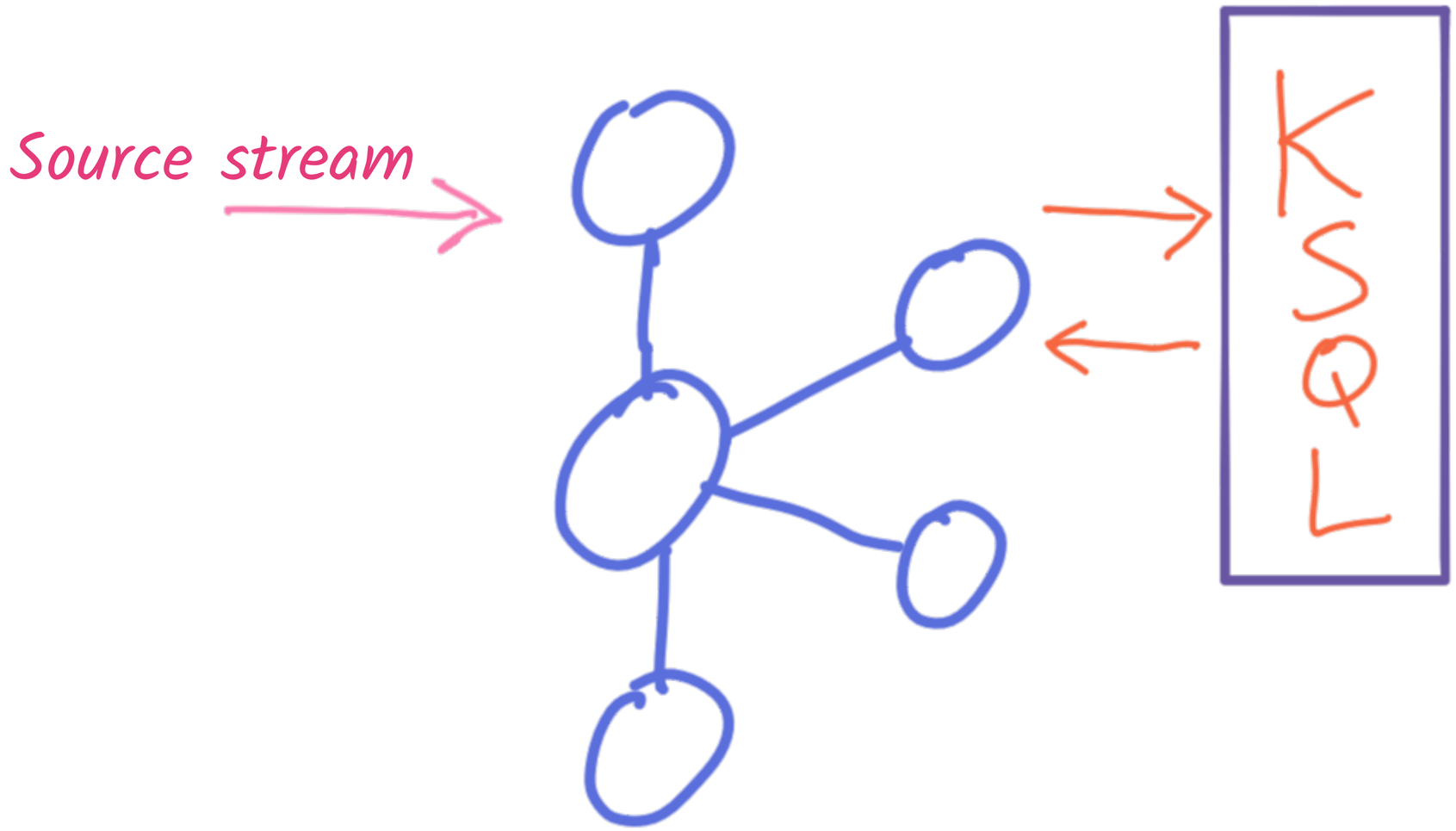
Stream:
widgets

KSQL

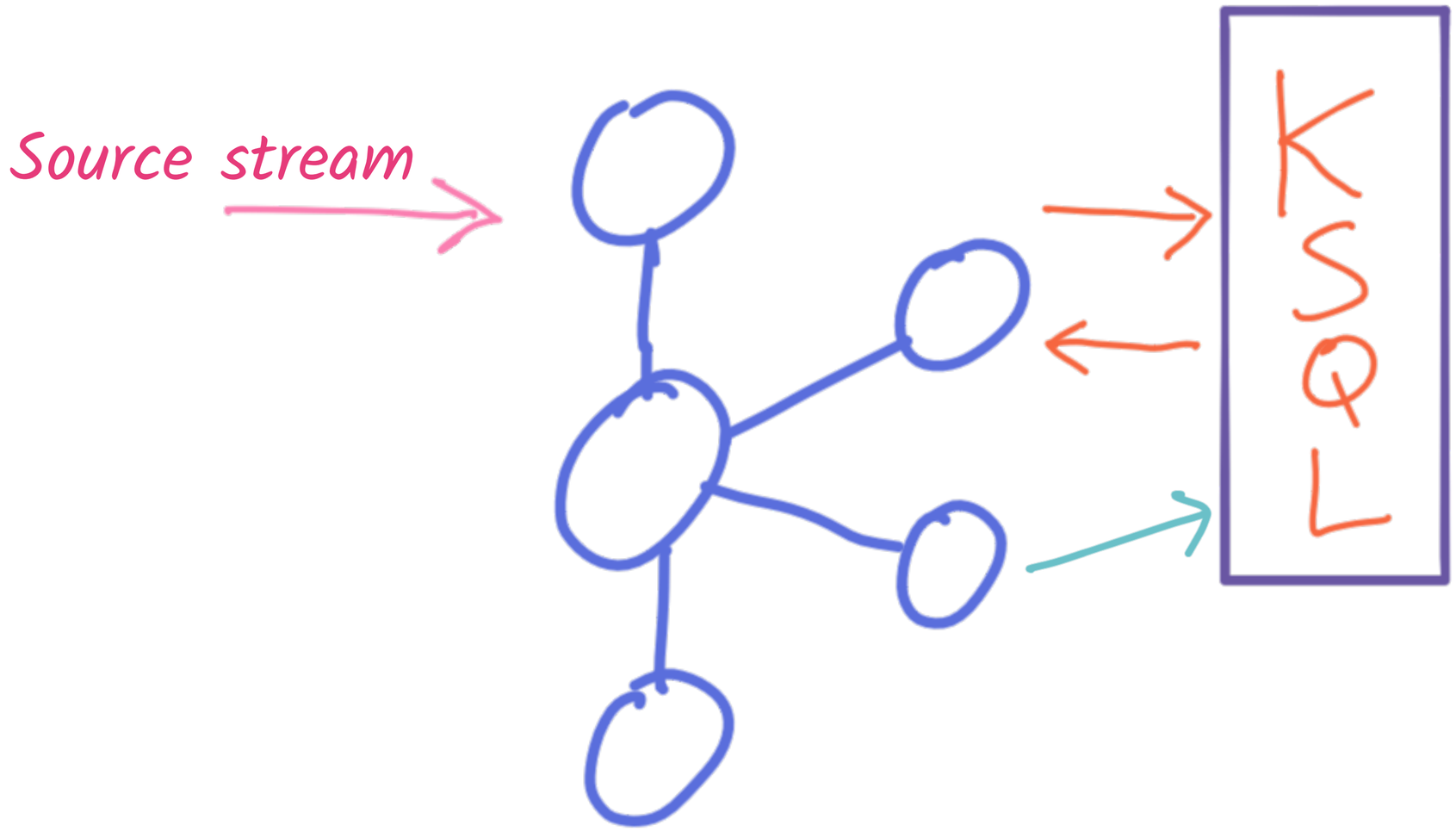
```
CREATE STREAM widgets_red AS
SELECT * FROM widgets
WHERE colour='RED';
```

Stream:
widgets_red

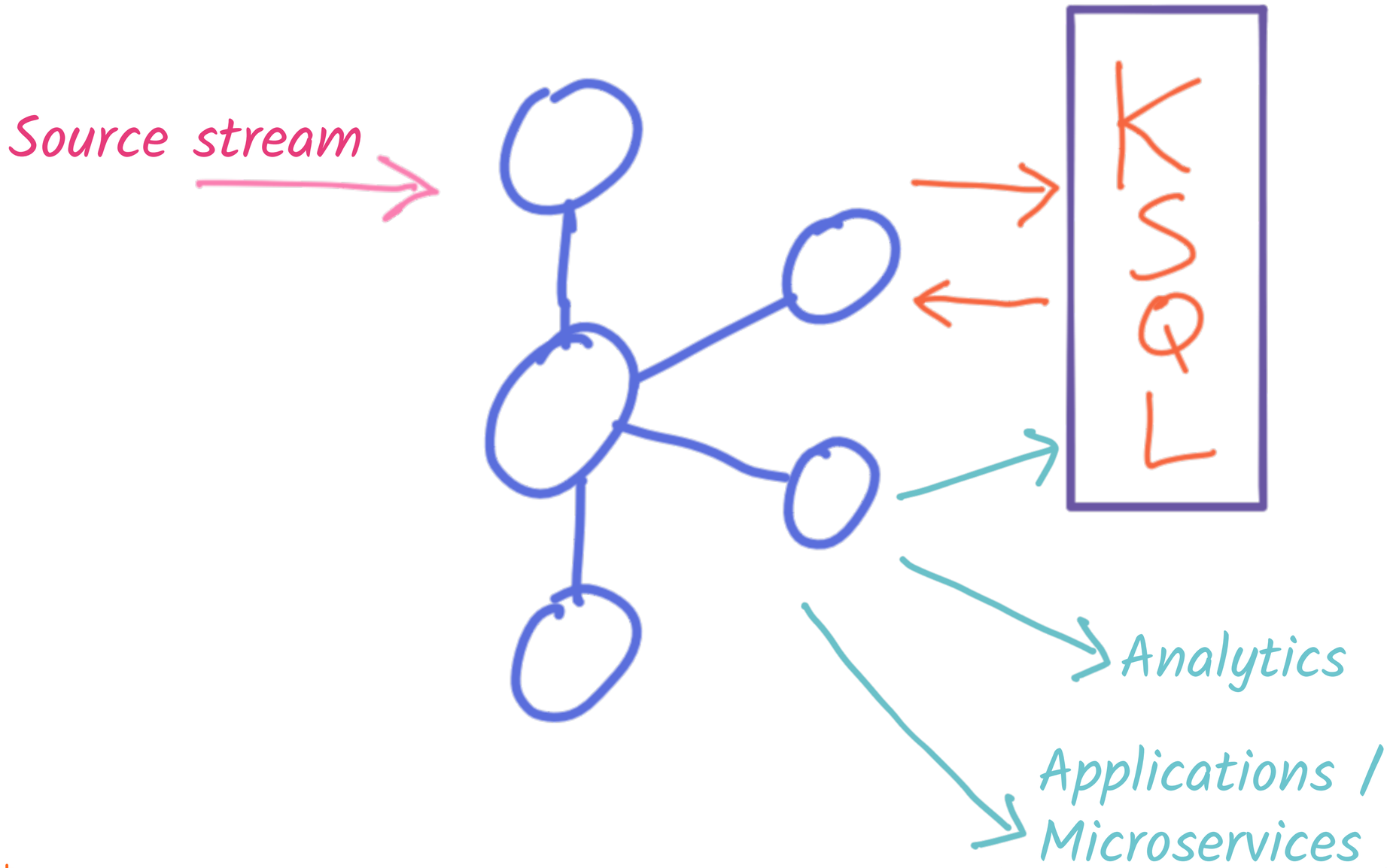
Stream Processing with KSQL



Stream Processing with KSQL

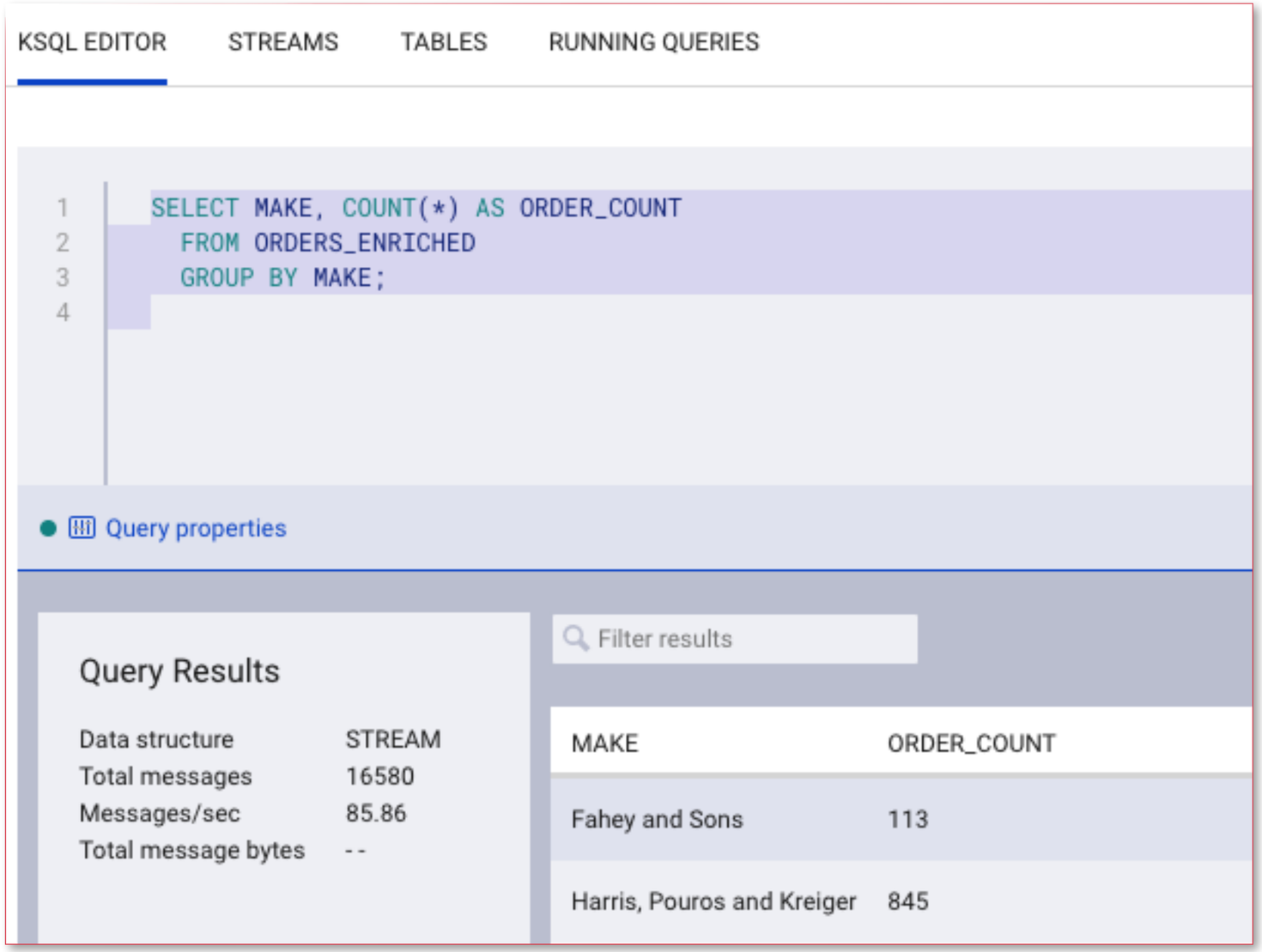


Stream Processing with KSQL



Interacting with KSQL

KSQL - Confluent Control Center



KSQL - CLI

```
CLI v5.2.2, Server v5.2.2 located at http://ksql-server:8088

Having trouble? Type 'help' (case-insensitive) for a rundown of how things work!

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

 Message
----------------------------
 Stream created and running
----------------------------
ksql> select * from ORDERS_NO_ADDRESS_DATA;
1562059702636 | 0 | 2019-07-02 09:28:22 | 0 | Item_48 | 16
1562059703535 | 0 | 2019-07-02 09:28:23 | 0 | Item_45 | 16
1562059703638 | 1 | 2019-07-02 09:28:23 | 1 | Item_18 | 15
1562059703804 | 2 | 2019-07-02 09:28:23 | 2 | Item_11 | 2
1562059703836 | 3 | 2019-07-02 09:28:23 | 3 | Item_0 | 6
```

KSQL - REST API

```
$ 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}
```

KSQL in action

http://rmoff.dev/ksql-intro_code

Filtering with KSQL

Filtering with KSQL

ORDERS


```
CREATE STREAM ORDERS_NY AS
  SELECT *
    FROM ORDERS
   WHERE ADDRESS->STATE='New York';
```

Filtering with KSQL

ORDERS

KSQL

```
CREATE STREAM ORDERS_NY AS
  SELECT *
    FROM ORDERS
   WHERE ADDRESS->STATE='New York';
```

ORDERS_NY

Schema manipulation with KSQL

ORDERS

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

Schema manipulation with KSQL

ORDERS

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

KSQL

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

Schema manipulation with KSQL

ORDERS

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

KSQL

```
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": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5
}
```

Schema manipulation with KSQL

ORDERS

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

Schema manipulation with KSQL

ORDERS

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

KSQL

```
CREATE STREAM ORDERS_FLAT AS
  SELECT […]
         ADDRESS->STREET AS ADDRESS_STREET,
          ADDRESS->CITY AS ADDRESS_CITY,
          ADDRESS->STATE AS ADDRESS_STATE
    FROM ORDERS;
```

Schema manipulation with KSQL

ORDERS

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

KSQL

```
CREATE STREAM ORDERS_FLAT AS
  SELECT […]
         ADDRESS->STREET AS ADDRESS_STREET,
         ADDRESS->CITY AS ADDRESS_CITY,
         ADDRESS->STATE AS ADDRESS_STATE
    FROM ORDERS;
```

ORDERS_FLAT

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

Reserialising data with KSQL

ORDERS

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

Reserialising data with KSQL

ORDERS

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

KSQL

```
CREATE STREAM ORDERS_CSV
  WITH (VALUE_FORMAT='DELIMITED) AS
  SELECT * FROM ORDERS_FLAT;
```

Reserialising data with KSQL

ORDERS


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

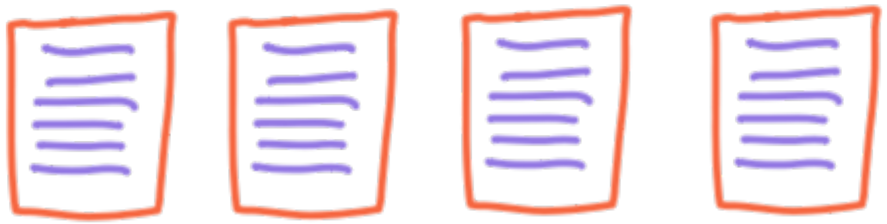

KSQL

```
CREATE STREAM ORDERS_CSV
  WITH (VALUE_FORMAT='DELIMITED) AS
  SELECT * FROM ORDERS_FLAT;
```

ORDERS_CSV


```
1560045914101,24644,Item_0,1,43078 De
1560047305664,24643,Item_29,3,209 Mon
1560057079799,24642,Item_38,18,3 Autu
1560088652051,24647,Item_6,6,82893 Ar
1560105559145,24648,Item_0,12,45896 W
1560108336441,24646,Item_33,4,272 Hef
1560123862235,24641,Item_15,16,0 Dort
1560124799053,24645,Item_12,1,71 Knut
```

Lookups and Joins with KSQL

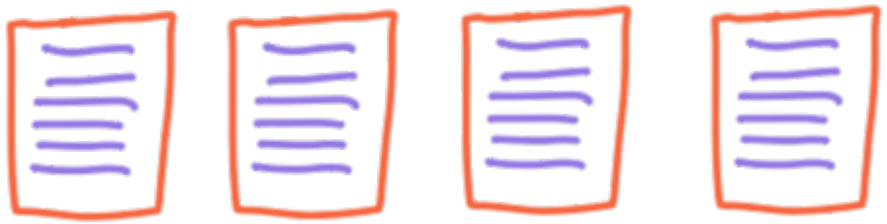


ORDERS

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

Lookups and Joins with KSQL

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 KSQL

ITEMS

ORDERS

KSQL

```
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 ;
```

```
{
  "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 KSQL

Lookup

ITEMS

ORDERS

KSQL

```
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 ;
```

ORDERS_ENRICHED


```
{
  "id": "Item_9",
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
 "unit_cost": 19.9
}
```

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

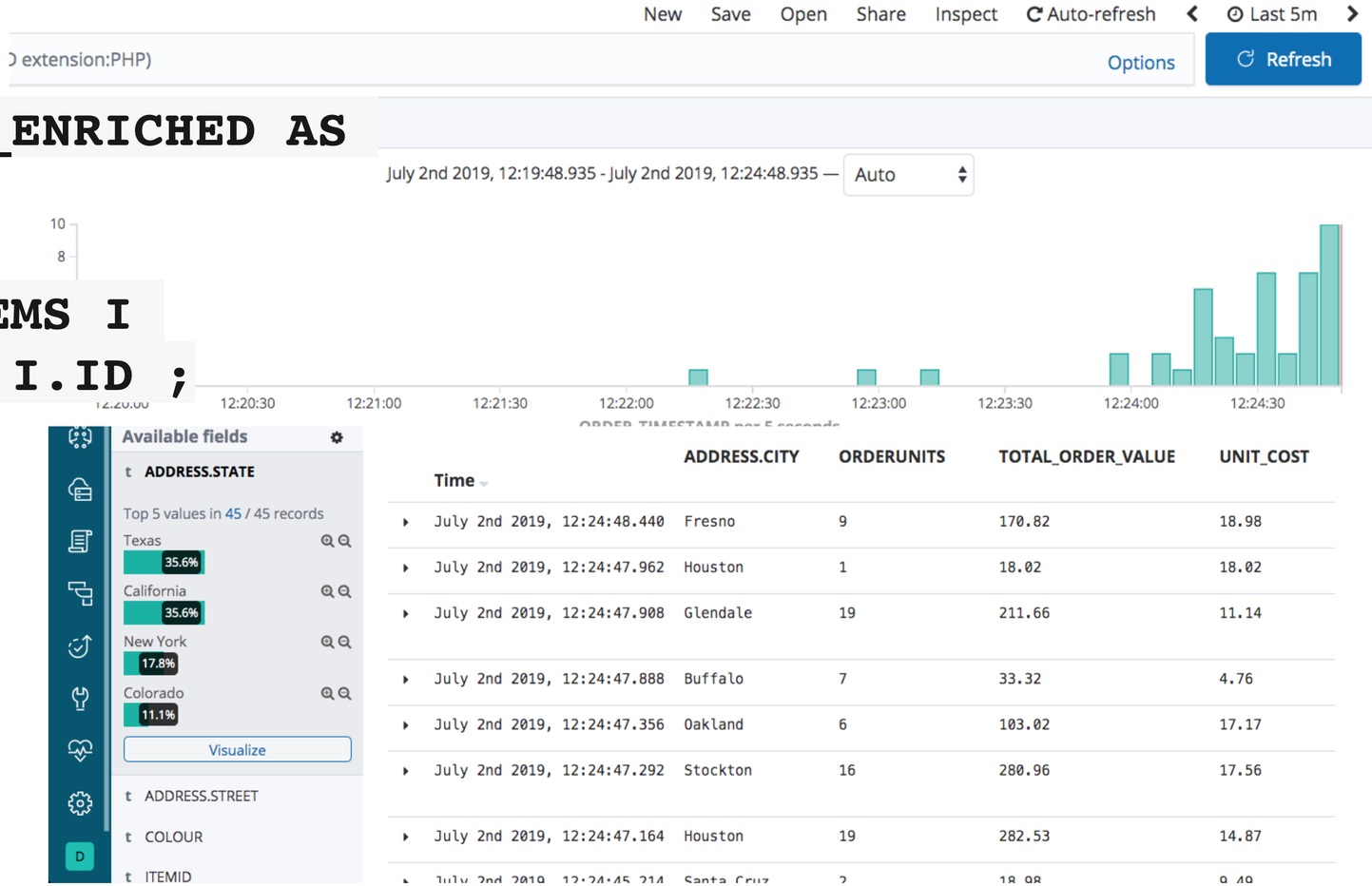
```
{
  "ordertime": 1560070133853,
  "orderid": 67,
  "itemid": "Item_9",
  "orderunits": 5,
  "make": "Boyle-McDermott",
  "model": "Apiaceae",
  "unit_cost": 19.9,
  "total_order_value": 99.5
}
```

Connecting to other systems with Kafka Connect

KSQL

```
CREATE STREAM ORDERS_ENRICHED AS
SELECT […]
  FROM ORDERS O
       INNER JOIN ITEMS I
       ON O.ITEMID = I.ID ;
```

Kafka Connect



Stateful Aggregation with KSQL

ORDERS

Stateful Aggregation with KSQL

ORDERS

```
SELECT MAKE, COUNT(*) AS ORDER_COUNT
  FROM ORDERS_ENRICHED
  GROUP BY MAKE;
```

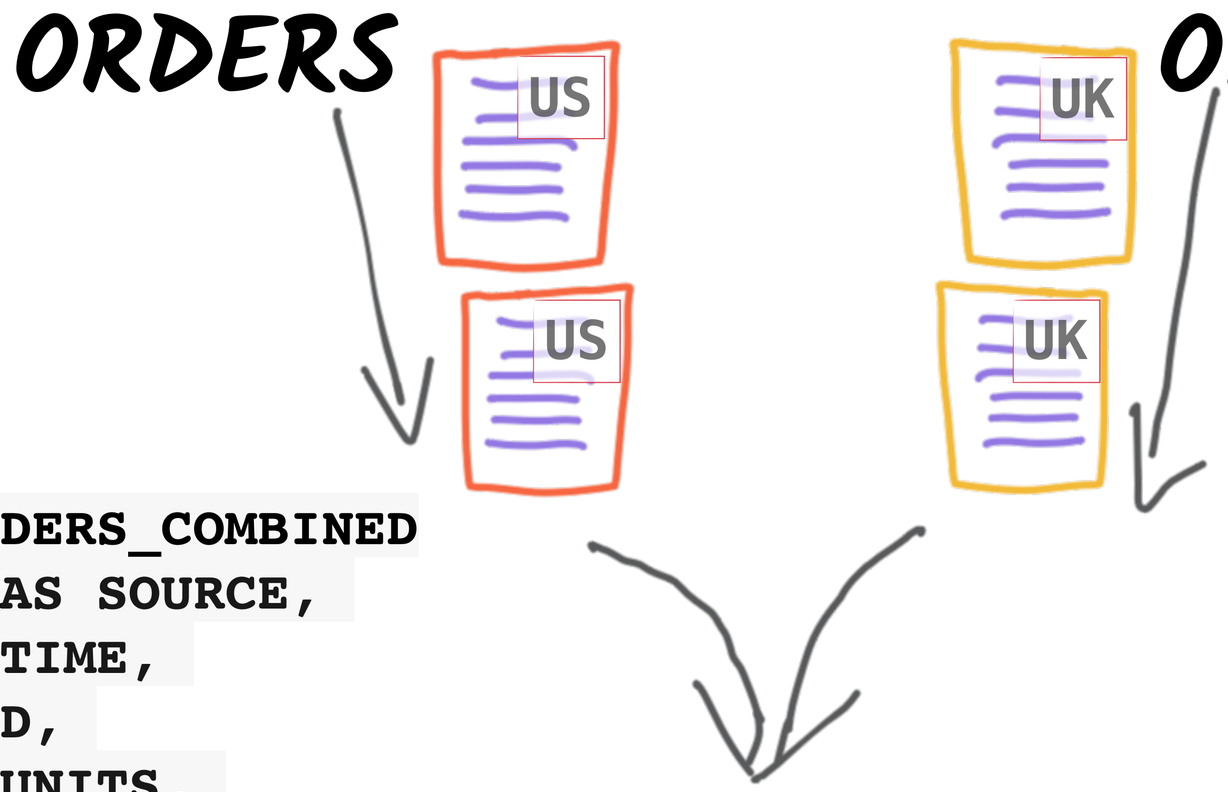
Stateful Aggregation with KSQL

ORDERS

```
SELECT MAKE, COUNT(*) AS ORDER_COUNT
  FROM ORDERS_ENRICHED
  GROUP BY MAKE;
```

Transform data with KSQL - merge streams

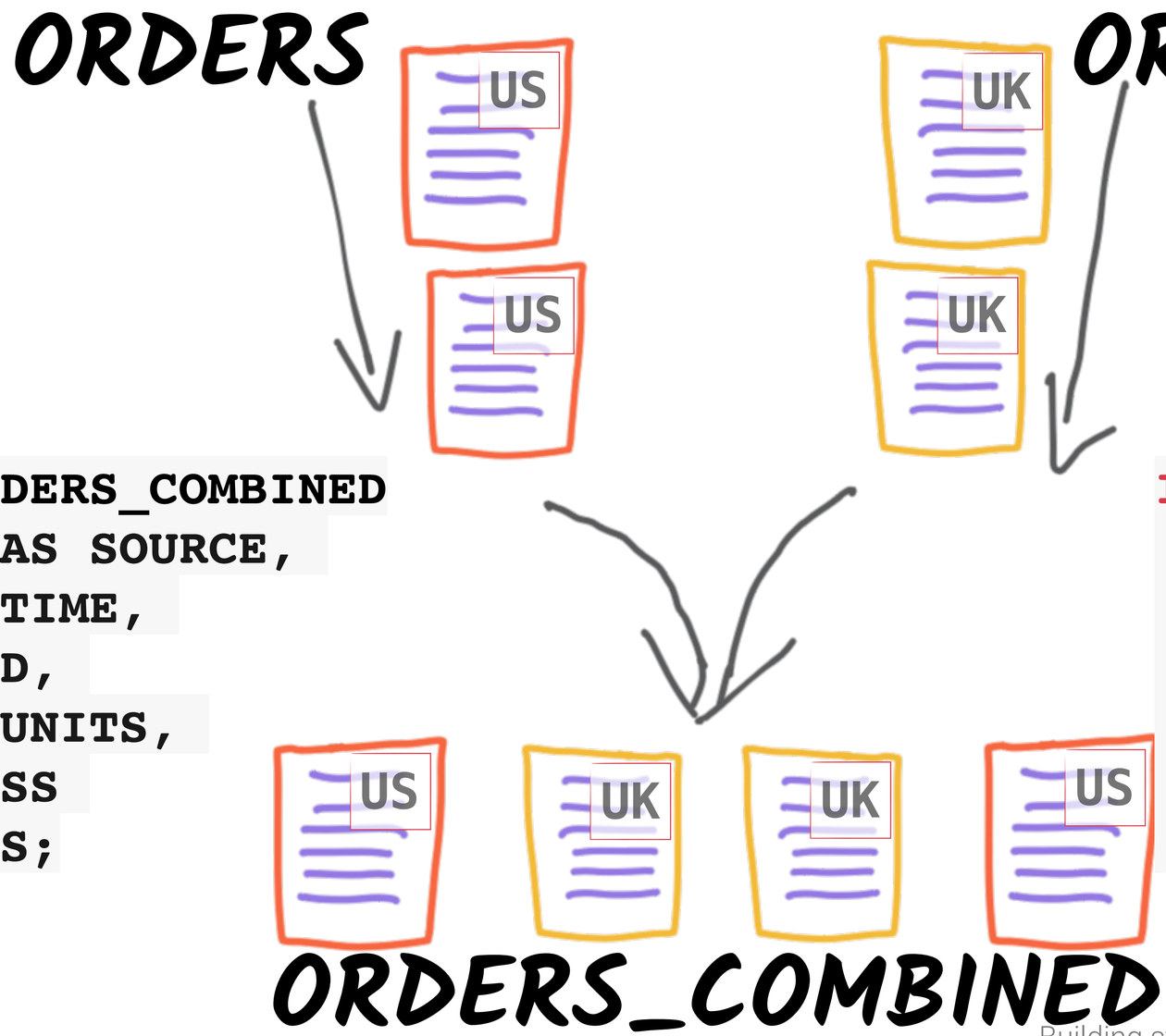
Transform data with KSQL - merge streams




```
INSERT INTO ORDERS_COMBINED
  SELECT 'US' AS SOURCE,
         ORDERTIME,
         ITEMID,
         ORDERUNITS,
         ADDRESS
    FROM ORDERS;
```

```
INSERT INTO ORDERS_COMBINED
  SELECT 'UK' AS SOURCE,
         ORDERTIME,
         ITEMID,
         ORDERUNITS,
         ADDRESS
    FROM ORDERS_UK;
```

Transform data with KSQL - merge streams




```
INSERT INTO ORDERS_COMBINED
  SELECT 'US' AS SOURCE,
         ORDERTIME,
         ITEMID,
         ORDERUNITS,
         ADDRESS
    FROM ORDERS;
```

```
INSERT INTO ORDERS_COMBINED
  SELECT 'UK' AS SOURCE,
         ORDERTIME,
         ITEMID,
         ORDERUNITS,
         ADDRESS
    FROM ORDERS_UK;
```

Transform data with KSQL - merge streams

ORDERS_COMBINED

Transform data with KSQL - merge streams

ORDERS_COMBINED

```
CREATE STREAM ORDERS_US AS
  SELECT *
    FROM ORDERS_COMBINED
   WHERE SOURCE ='US';
```

```
CREATE STREAM ORDERS_UK AS
  SELECT *
    FROM ORDERS_COMBINED
   WHERE SOURCE ='UK';
```

Transform data with KSQL - merge streams

ORDERS_COMBINED

```
CREATE STREAM ORDERS_US AS
  SELECT *
    FROM ORDERS_COMBINED
   WHERE SOURCE ='US';
```

```
CREATE STREAM ORDERS_UK AS
  SELECT *
    FROM ORDERS_COMBINED
   WHERE SOURCE ='UK';
```

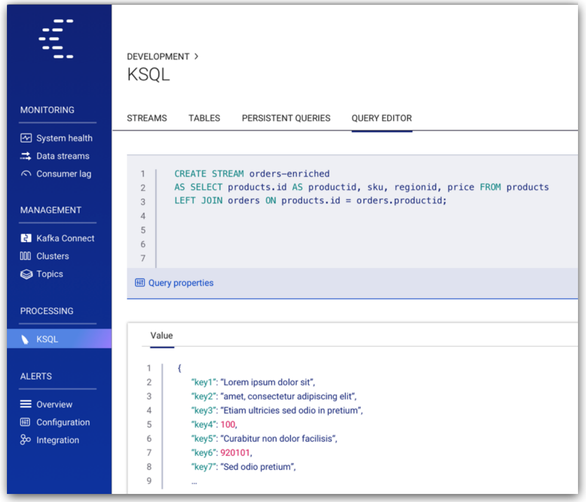
ORDERS_US ORDERS_UK

KSQL operations and deployment

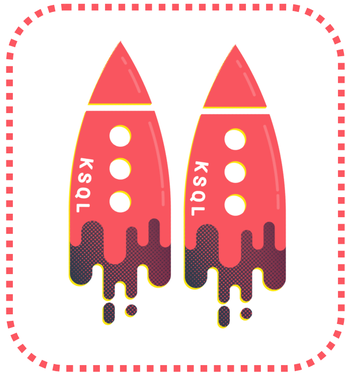
KSQL in Development and Production

Interactive KSQL
for development and testing

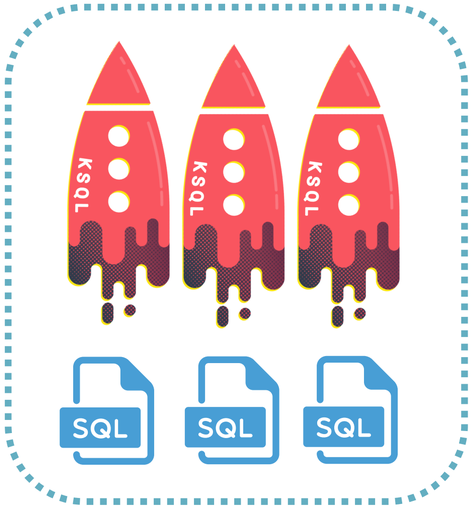
Headless KSQL
for Production



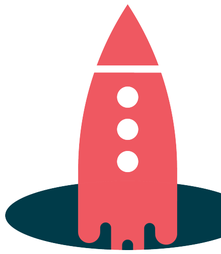
REST



Desired KSQL queries have been identified



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



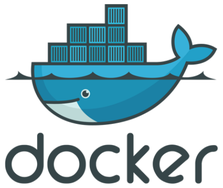
How to run KSQL

KSQL Server
(JVM process)

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

Docker images
confluentinc/cp-ksql-server
confluentinc/cp-ksql-cli

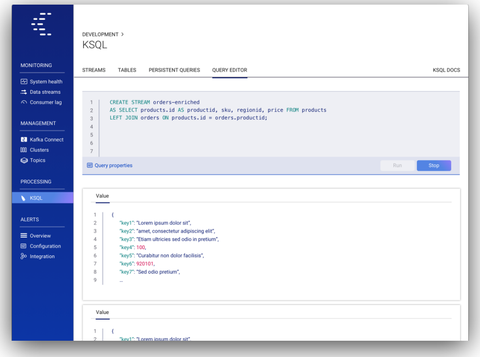

Physical

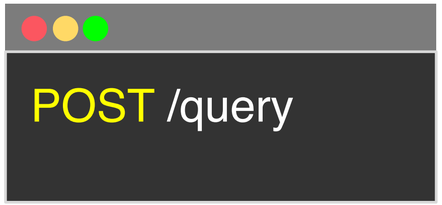


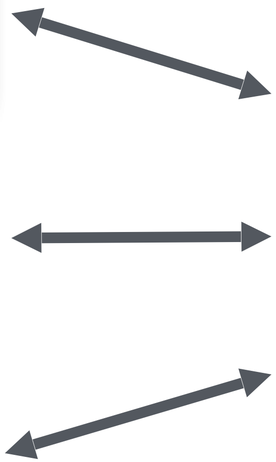
…and many more…

How to run KSQL

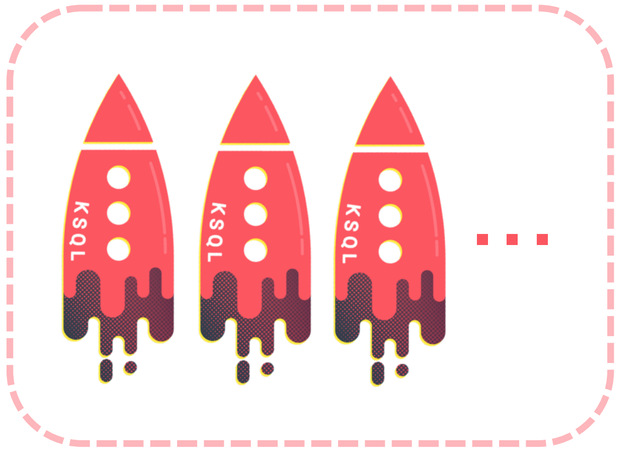
#1 Interactive KSQL, for development & testing



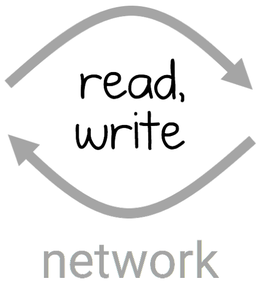




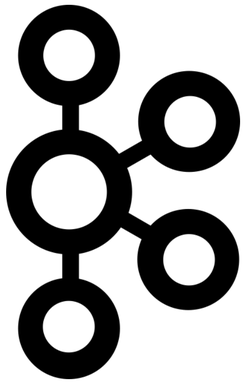
KSQL Cluster
(processing)



KSQL does not run
on Kafka brokers!

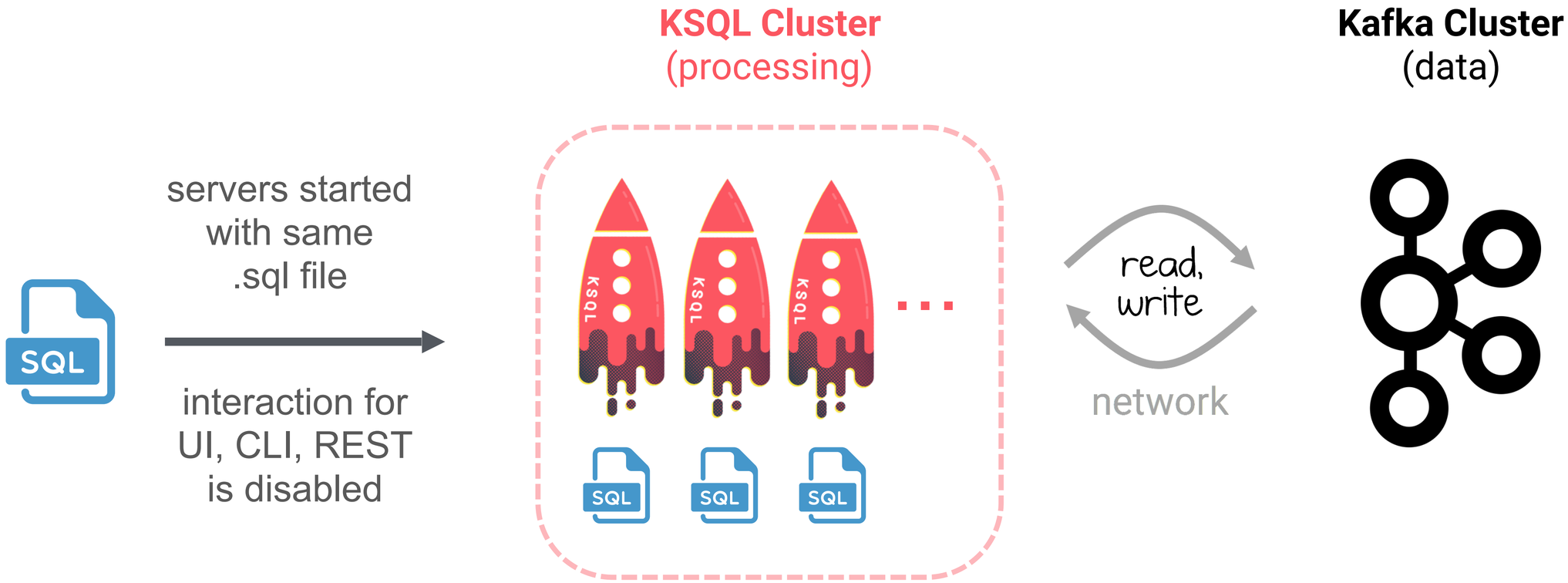


Kafka Cluster
(data)



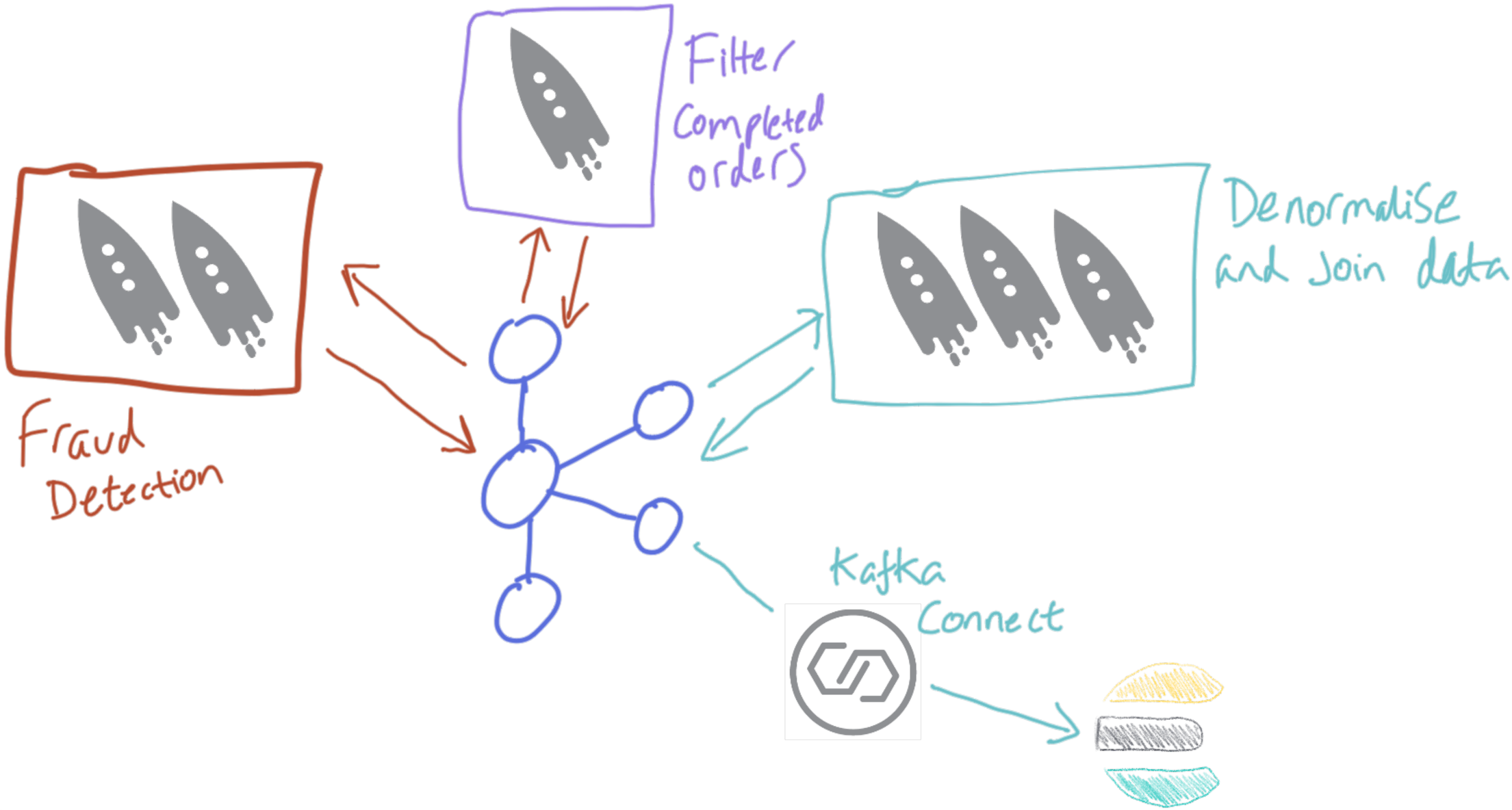
How to run KSQL

#2 Headless KSQL, for production



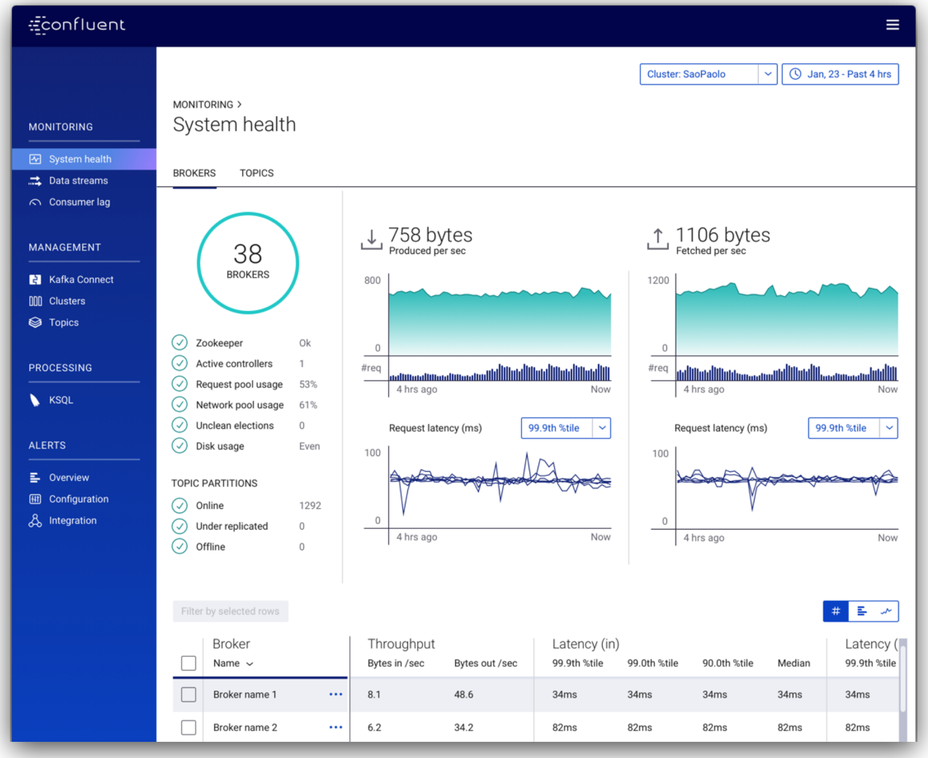

ONE DOES NOT SIMPLY RUN
A SINGLE KSQL CLUSTER

Think Applications, not database instances



Monitoring KSQL

Confluent Control Center



JMX

https://www.confluent.io/blog/troubleshooting-ksql-part-2


NEXT STOP
SAN FRANCISCO
SEPT 30-OCT 1
CONFLUENT COMMUNITY DISCOUNT CODE
KS19Meetup
25% OFF*
kafka summit
*Standard Priced Conference pass


http://cnfl.io/demo-scene
http://cnfl.io/book-bundle
http://cnfl.io/slack
@rmoff
robin@confluent.io

Related Talks

- The Changing Face of ETL: Event-Driven Architectures for Data Engineers
  - 📖 Slides
  - 📽️ Recording
- ATM Fraud detection with Kafka and KSQL
  - 📖 Slides
  - 👾 Code
  - 📽️ Recording
- Embrace the Anarchy: Apache Kafka's Role in Modern Data Architectures
  - 📖 Slides
  - 📽️ Recording
- Apache Kafka and KSQL in Action : Let's Build a Streaming Data Pipeline!
  - 📖 Slides
  - 👾 Code
  - 📽️ Recording
- No More Silos: Integrating Databases and Apache Kafka
  - 📖 Slides
  - 👾 Code (MySQL)
  - 👾 Code (Oracle)
  - 📽️ Recording

#EOF