



A vertical column of decorative orange bars of varying heights, starting from the top right corner and extending downwards towards the center.

ksql

The streaming SQL engine for Apache Kafka®



tom.green@confluent.io

Agenda & Housekeeping

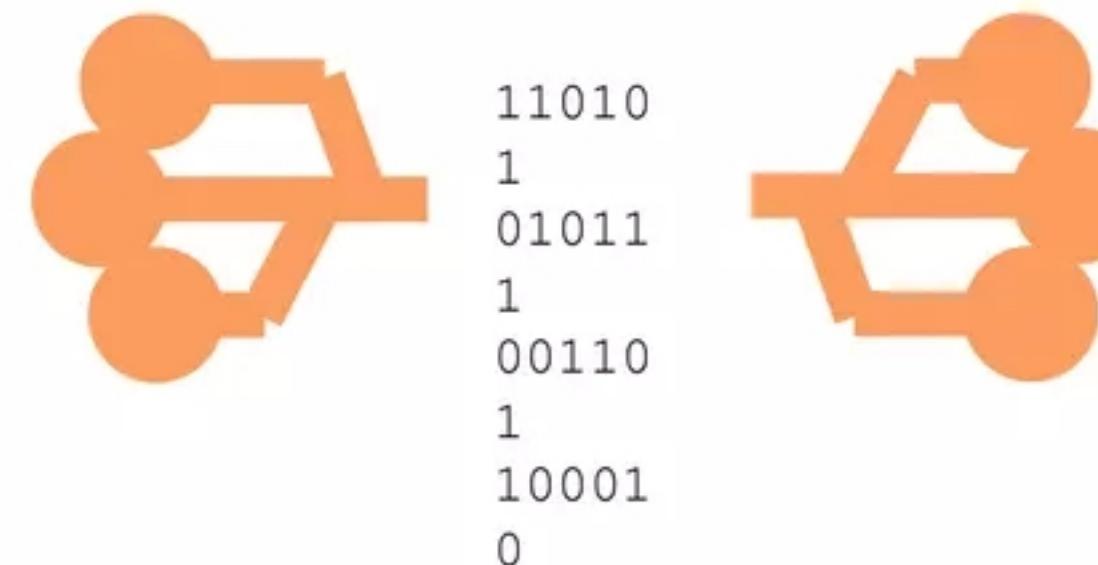
- Introduction to KSQL Presentation
- KSQL Demonstration
- Q & A - please ask your questions via that chat box

Follow up materials

- The slides and recording will be emailed

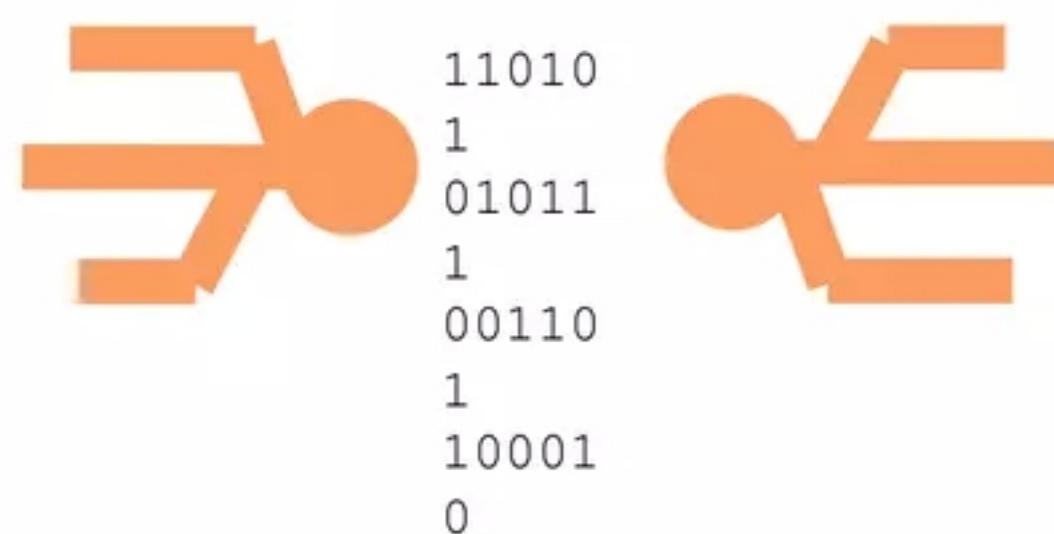
Apache Kafka is a Distributed Streaming Platform

Publish and subscribe to streams of data



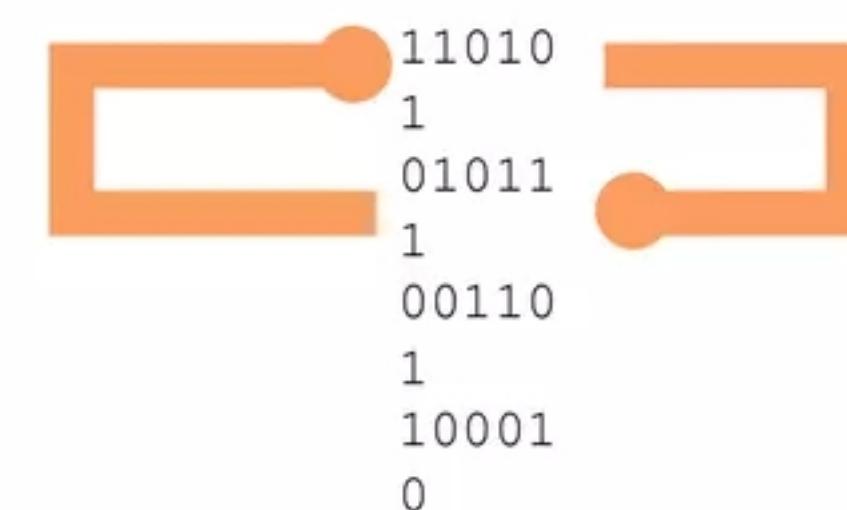
similar to a message queue or enterprise messaging system.

Store streams of data



in a fault tolerant way.

Process streams of data



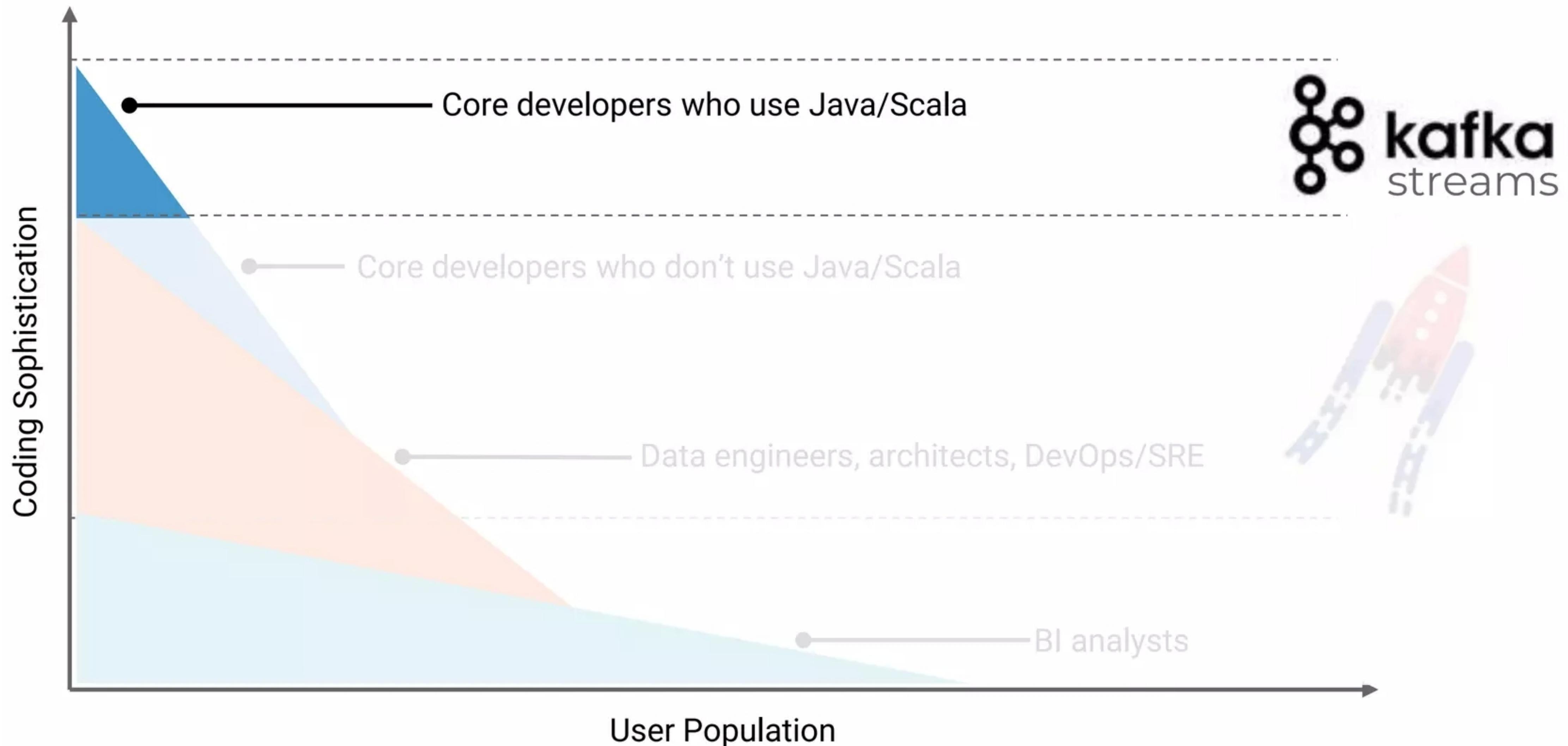
in real time, as they occur.



KSQL

The streaming SQL engine for Apache
Kafka® to write real-time applications in SQL

Lower the bar to enter the world of streaming



Lower the bar to enter the world of streaming



KSQL

```
CREATE STREAM fraudulent_payments AS
  SELECT * FROM payments
  WHERE fraudProbability > 0.8;
```

VS.



```
object FraudFilteringApplication extends App {
    val config = new java.util.Properties()
    config.put(StreamsConfig.APPLICATION_ID_CONFIG, "fraud-filtering")
    config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker")

    val builder: StreamsBuilder = new StreamsBuilder()
    val fraudulentPayments: KStream[String, Payment] = builder
        .stream[String, Payment]("payments-kafka-topic")
        .filter((_, payment) -> payment.fraudProbability > 0.8)

    val streams: KafkaStreams = new KafkaStreams(builder.build(), co
    streams.start()
}
```

KSQL

- You write **only SQL**.
No Java, Python, or
other boilerplate to
wrap around it!
- Create KSQL **user
defined functions** in
Java when needed.

```
CREATE STREAM fraudulent_payments AS  
SELECT * FROM payments  
WHERE fraudProbability > 0.8;
```

New user experience: interactive stream processing

The screenshot shows the KSQL Query Editor within the Confluent Platform UI. The left sidebar has sections for MONITORING (System health, Data streams, Consumer lag) and MANAGEMENT (Kafka Connect, Clusters, Topics). The PROCESSING section is active, showing KSQL. The main area shows a KSQL query to create a stream:

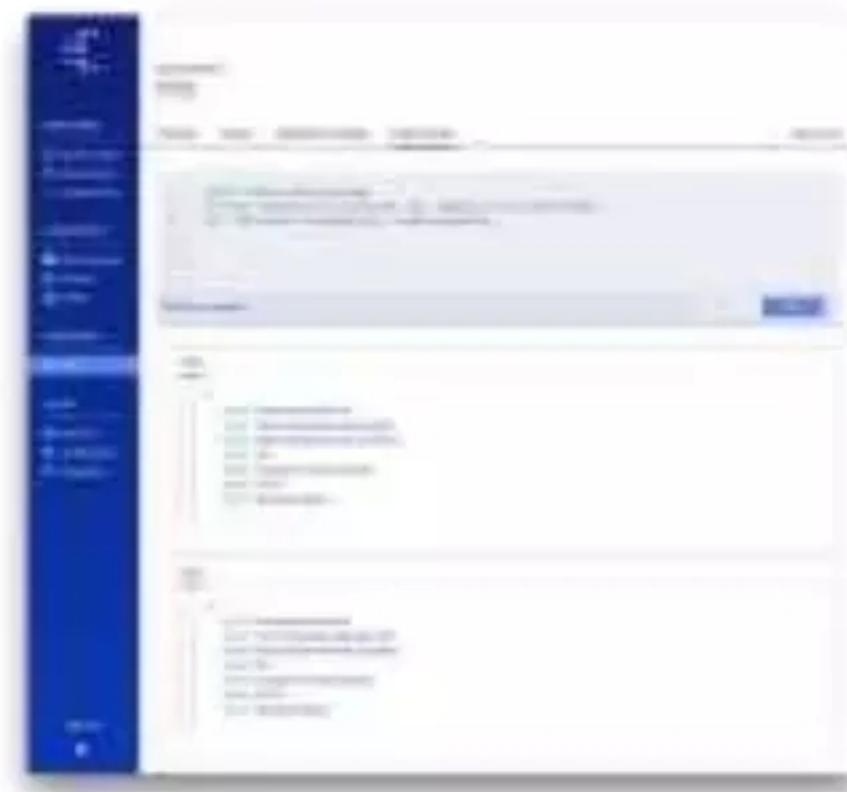
```
1 CREATE STREAM orders-enriched
2 AS SELECT products.id AS productid, sku, regionid, price FROM products
3 LEFT JOIN orders ON products.id = orders.productid;
```

Below the query is a "Query properties" section with "Run" and "Stop" buttons. At the bottom, there is a table with a single row labeled "Value".

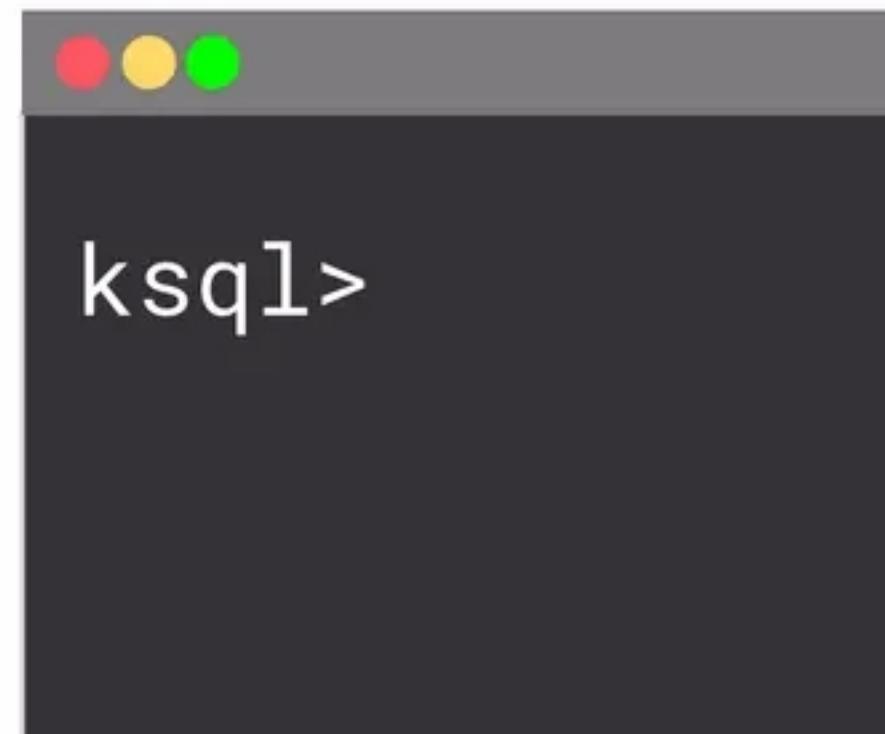
Value
1 { 2 "key1": "Lorem ipsum dolor sit", 3 "key2": "amet, consectetur adipiscing elit", 4 "key3": "Etiam ultricies sed odio in pretium", 5 "key4": 100, 6 "key5": "Curabitur non dolor facilisis", 7 "key6": 020101, 8 "key7": "Sed odio pretium",

KSQL can be used interactively + programmatically

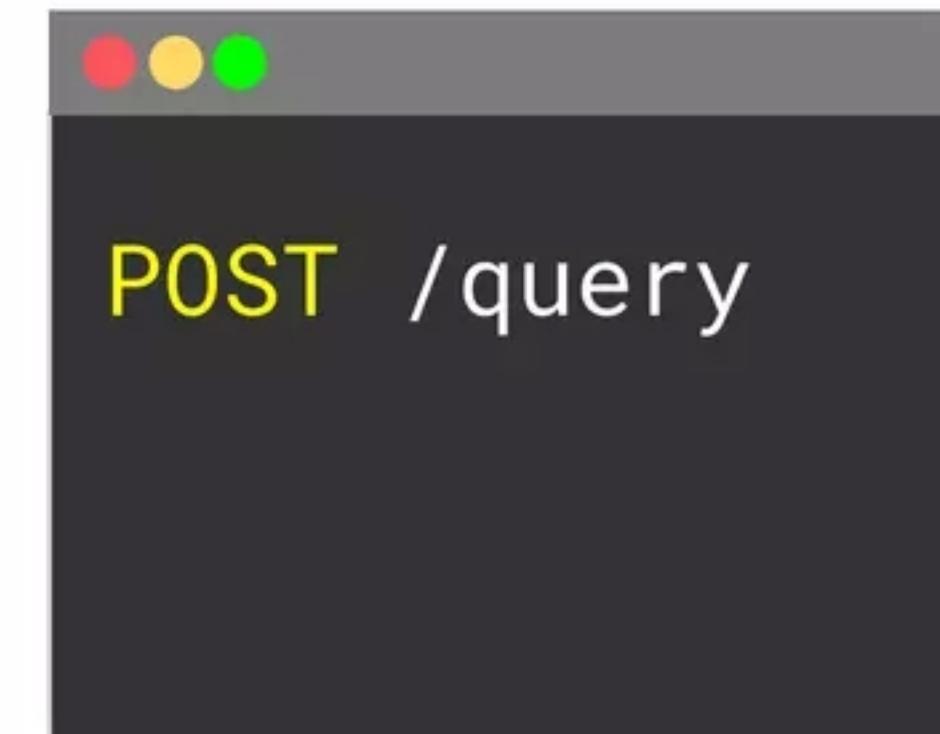
1 UI



2 CLI



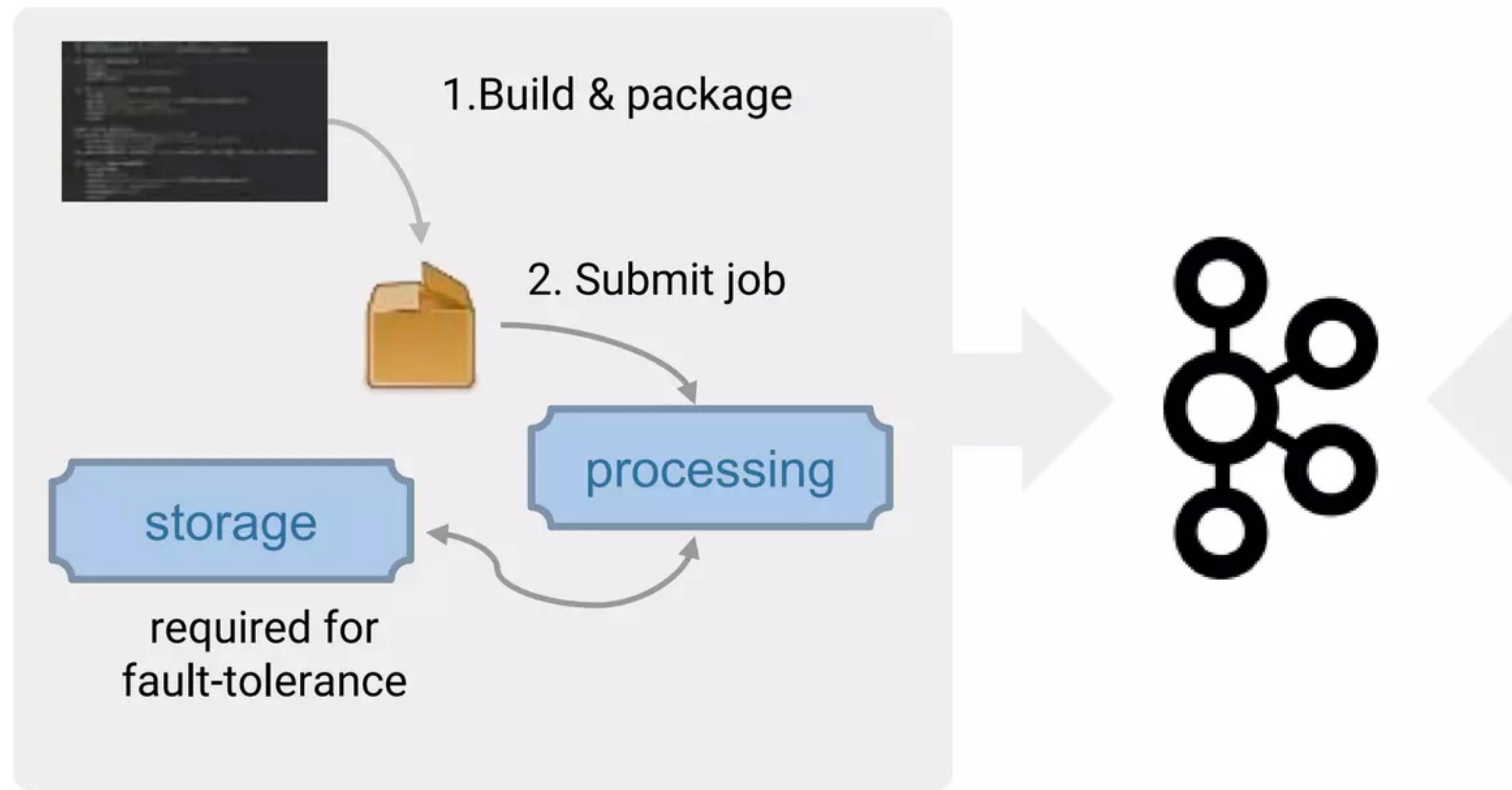
3 REST



4 Headless



All you need is Kafka and KSQL

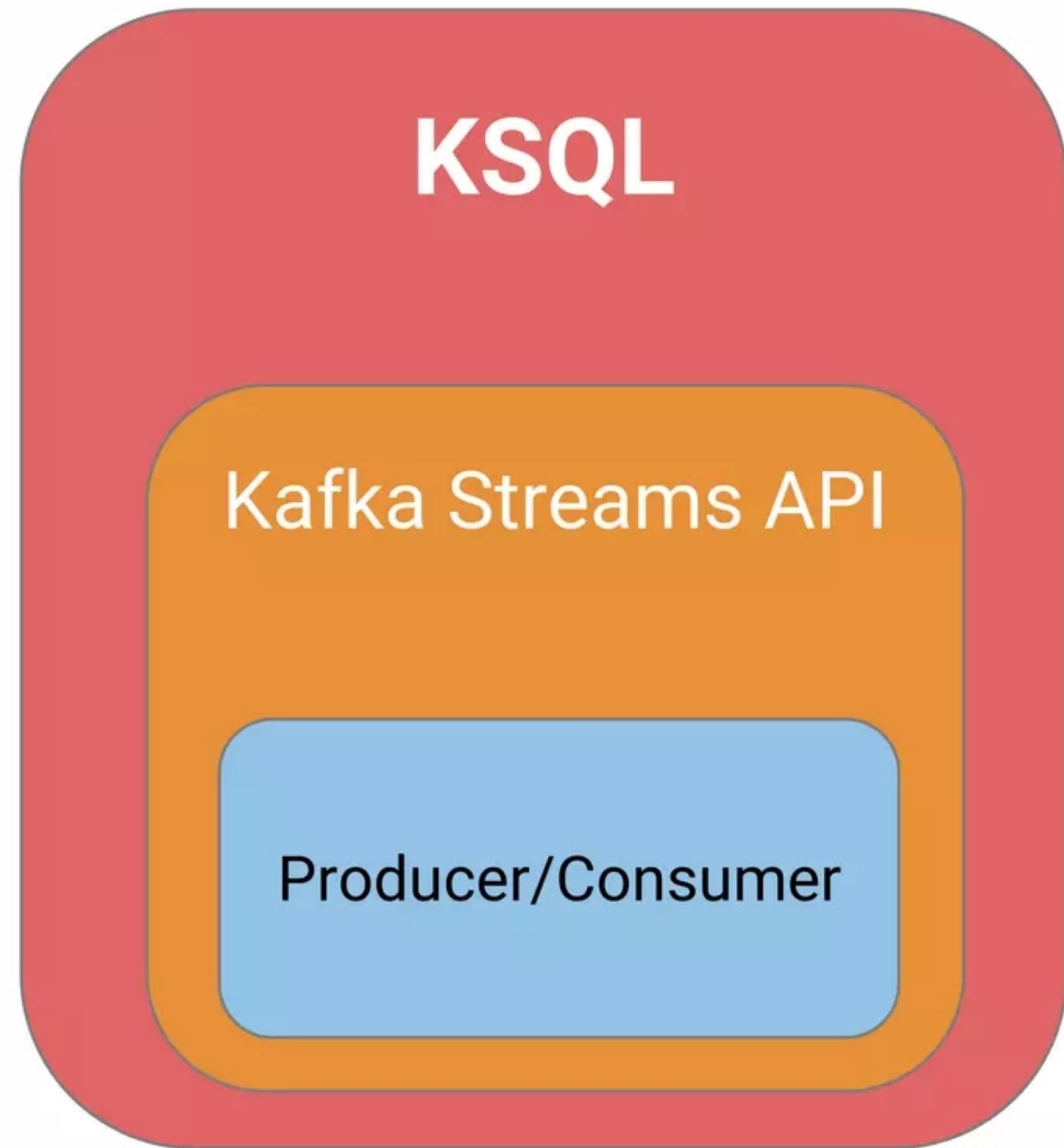


Without KSQL

A terminal window with three colored status indicators (red, yellow, green) at the top. The command `ksql> SELECT * FROM myStream` is being typed into the window.

With KSQL

Stream Processing in Kafka



- SQL
 - Interactive service
 - ...
-
- Stateful stream processing
 - Window operations
 - DSL
 - `map()`, `filter()`,
`join()`, `aggregate()`, ...
- ...

- Elasticity
 - Fault Tolerance
- ...



Stream/Table Duality

The Stream-Table Duality



KSQL Examples

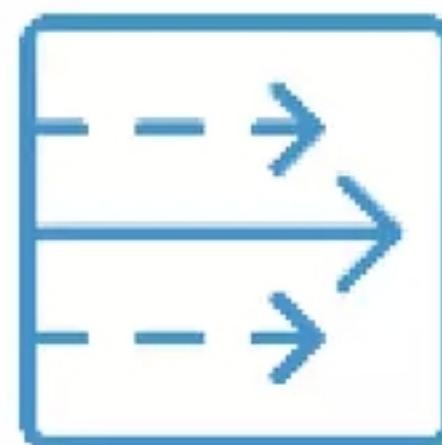
KSQL example use cases



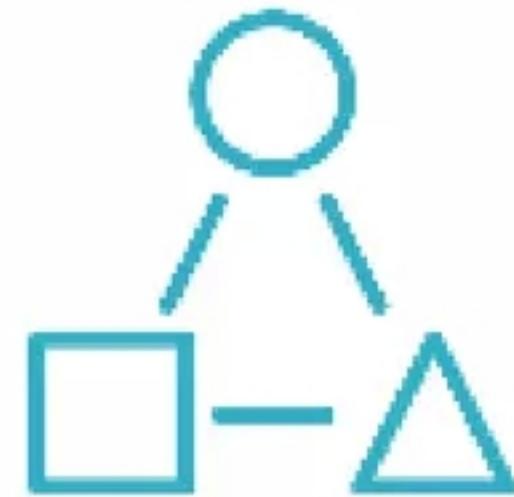
Data exploration



Data enrichment



Streaming ETL



Filter, cleanse, mask

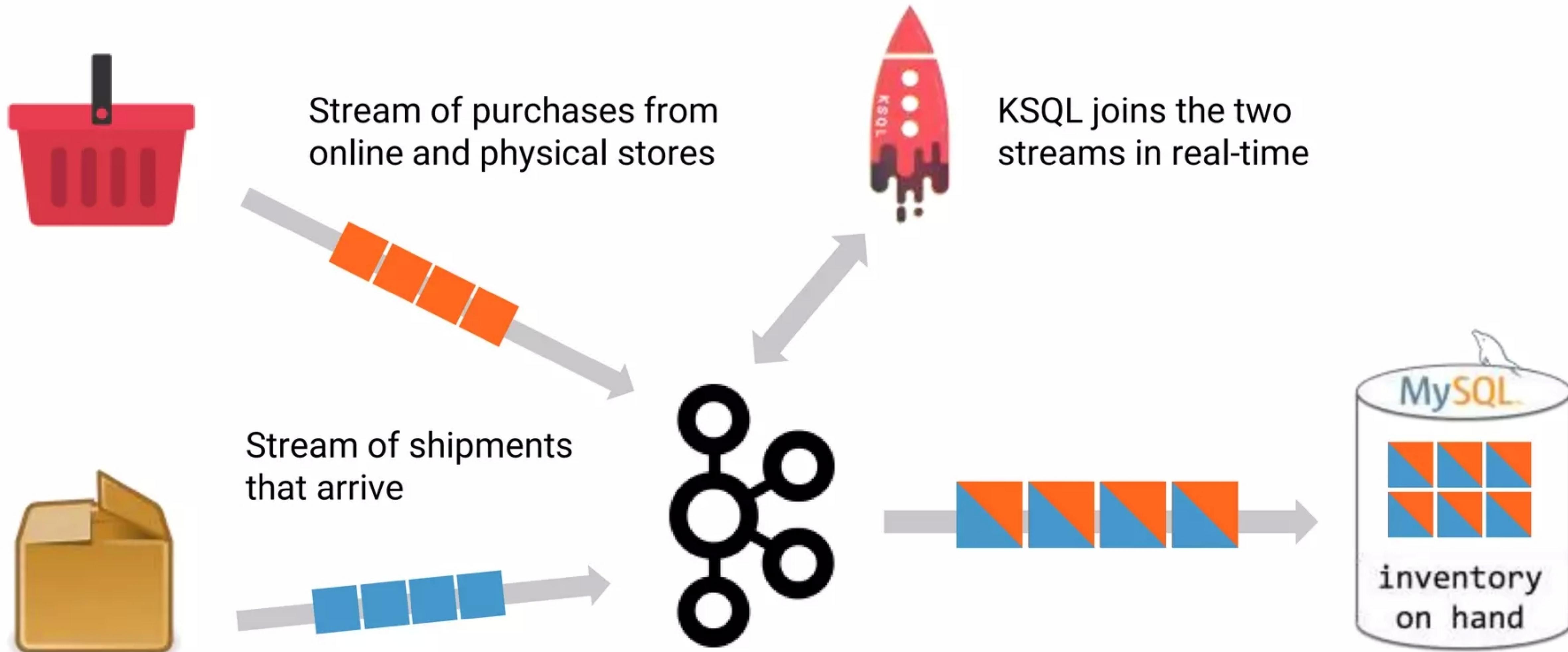


Real-time monitoring

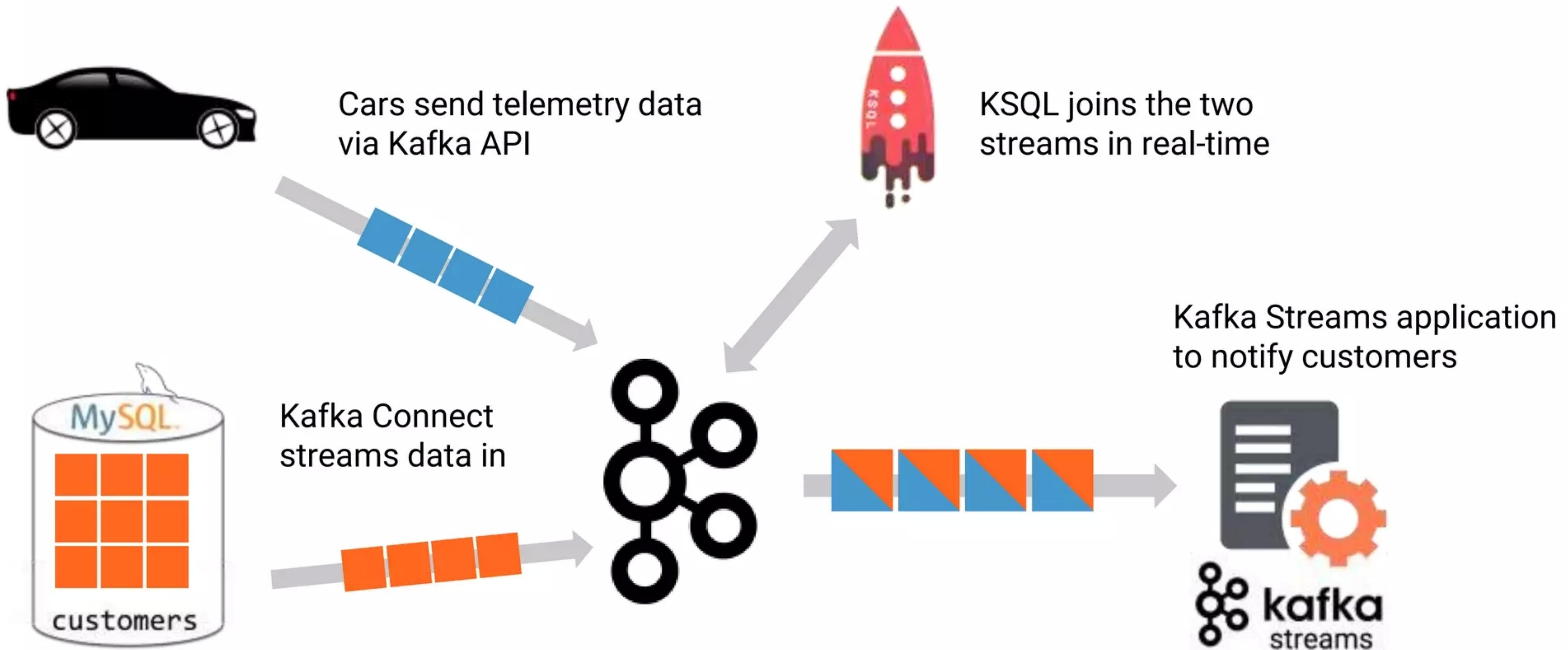


Anomaly detection

Example: Retail



Example: IoT, Automotive, Connected Cars



KSQL for Real-Time Monitoring

- Log data monitoring
- Tracking and alerting
- Syslog data
- Sensor / IoT data
- Application metrics

```
CREATE STREAM syslog_invalid_users AS  
SELECT host, message  
FROM syslog  
WHERE message LIKE '%Invalid user%';
```

<http://cnfl.io/syslogs-filtering> / <http://cnfl.io/syslog-alerting>

KSQL for Anomaly Detection

- Identify 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 Streaming ETL

- Joining, filtering, and aggregating streams of event data

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

KSQL for Data Transformation

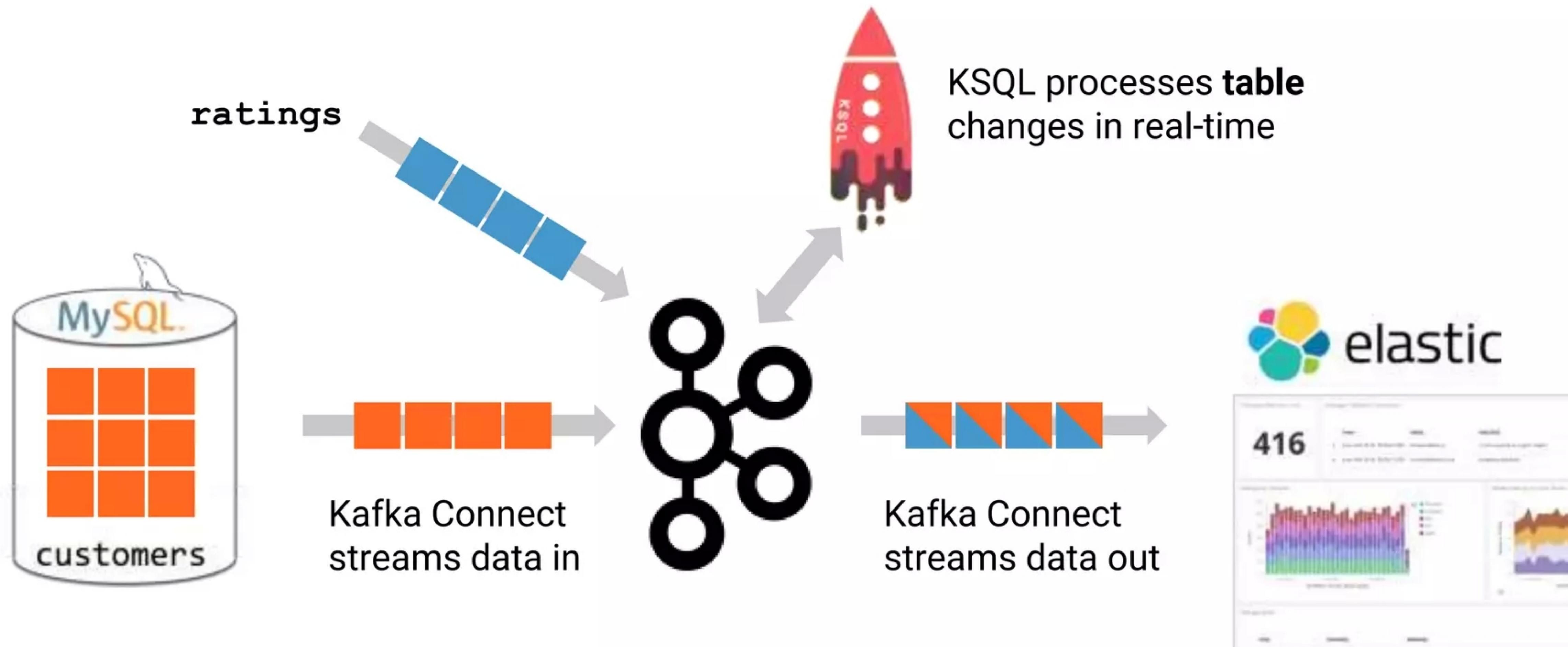
- Easily make derivations of existing topics

```
CREATE STREAM pageviews_avro
WITH (PARTITIONS=6,
      VALUE_FORMAT='AVRO') AS
SELECT * FROM pageviews_json
PARTITION BY user_id;
```



Demo

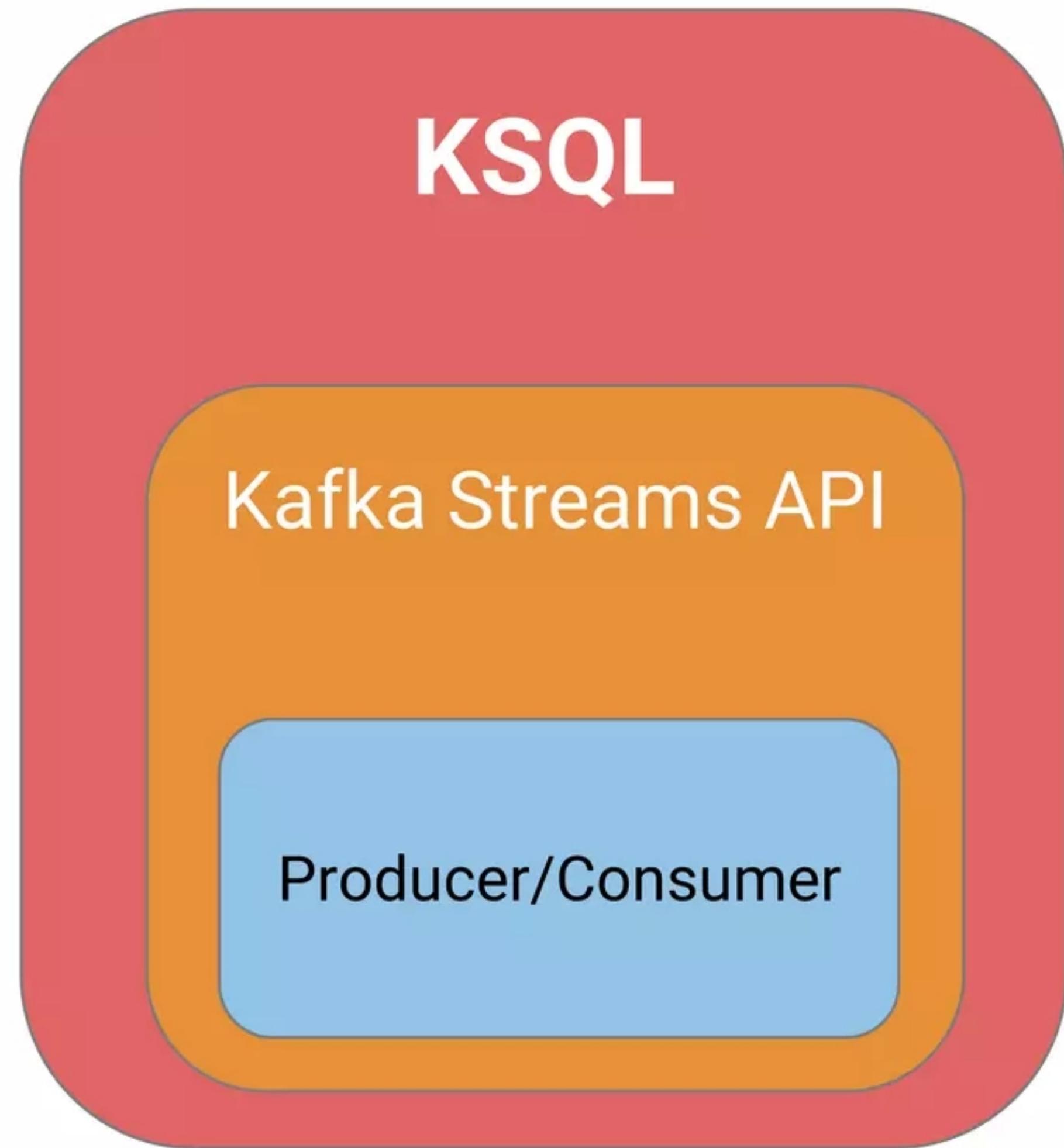
Example: CDC from DB via Kafka to Elastic





Deployment Scalability and HA

Stream Processing in Kafka

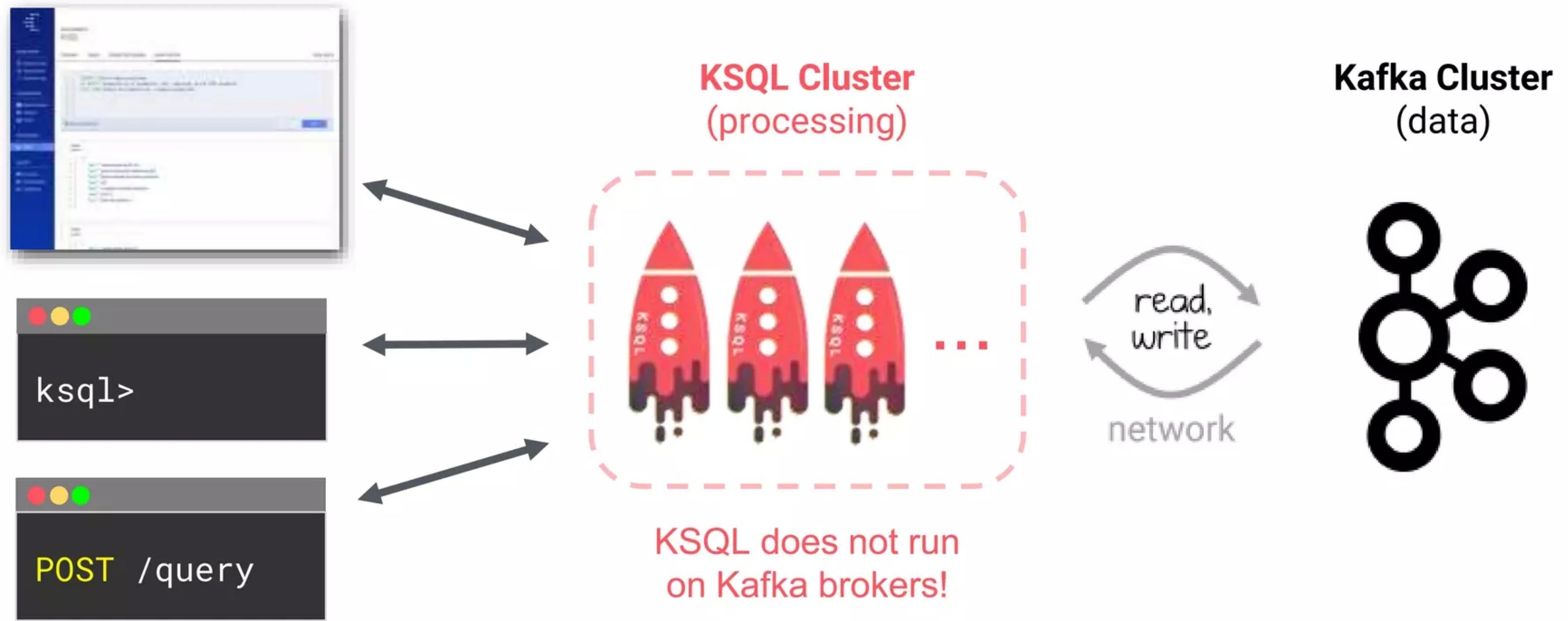


- SQL
 - Interactive service
 - ...
-
- Stateful stream processing
 - Window operations
 - DSL
 - `map()`, `filter()`,
`join()`, `aggregate()`, ...
- ...

- Elasticity
 - Fault Tolerance
- ...

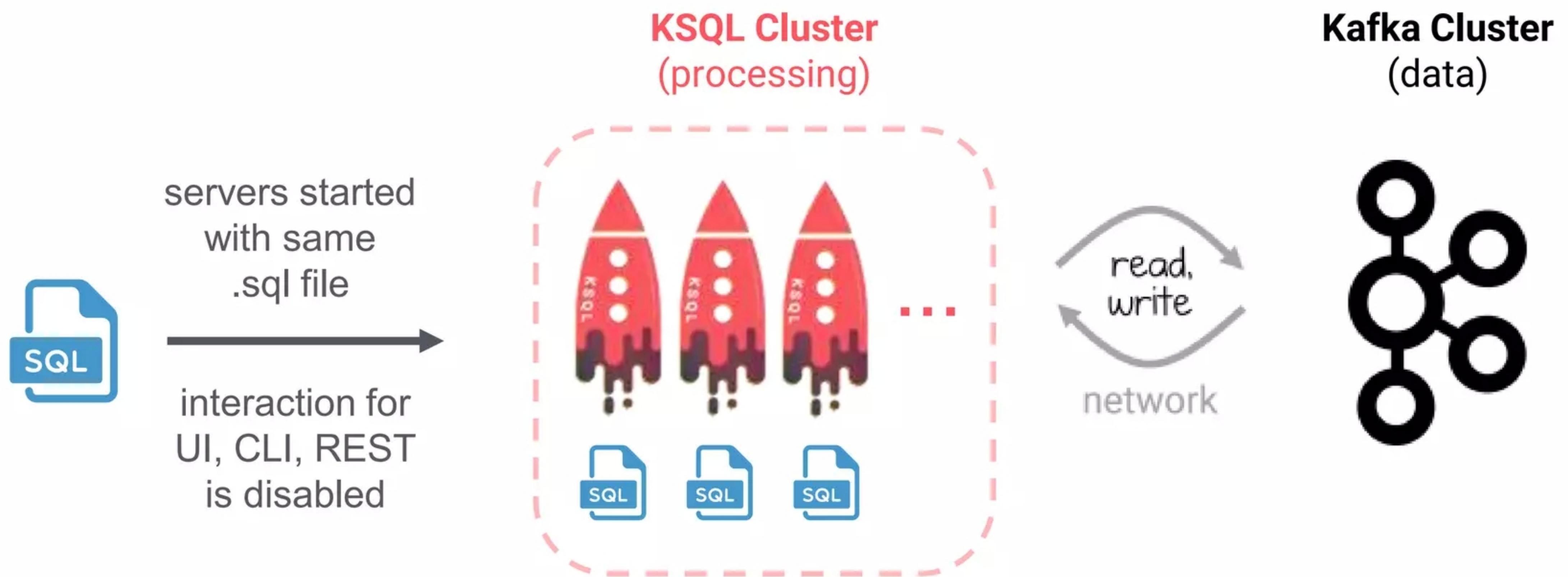
How to run KSQL

#1 Interactive KSQL, for development & testing



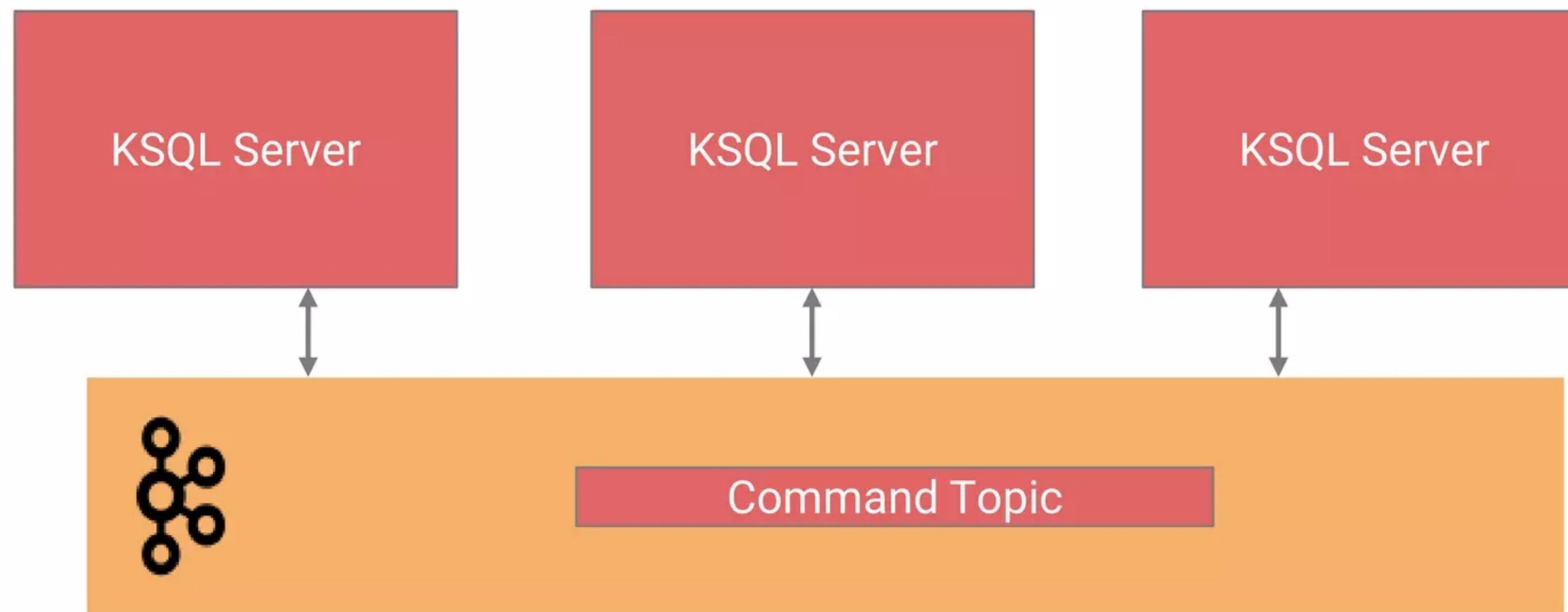
How to run KSQL

#2 Headless KSQL, for production

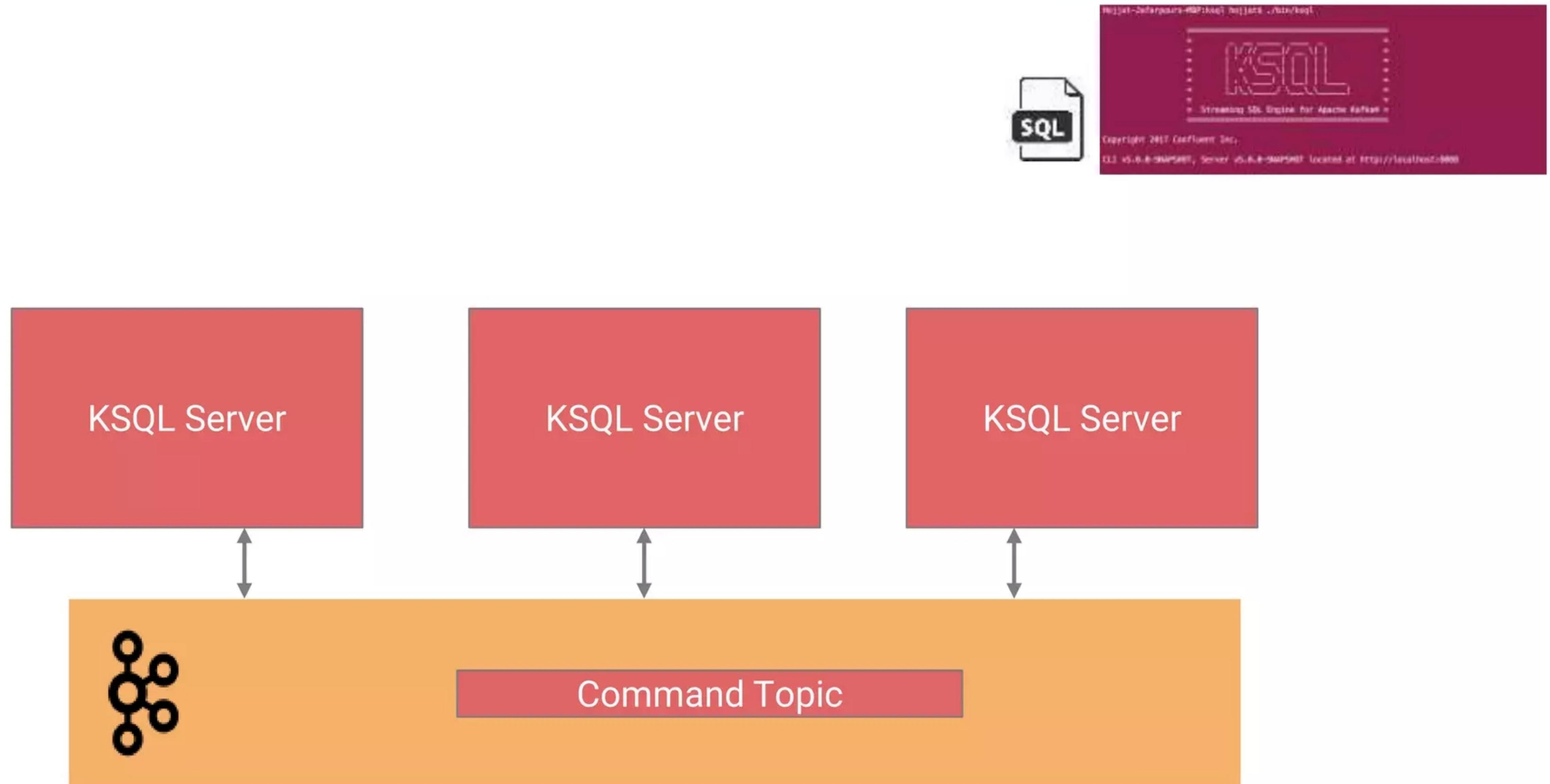


KSQL Deployment

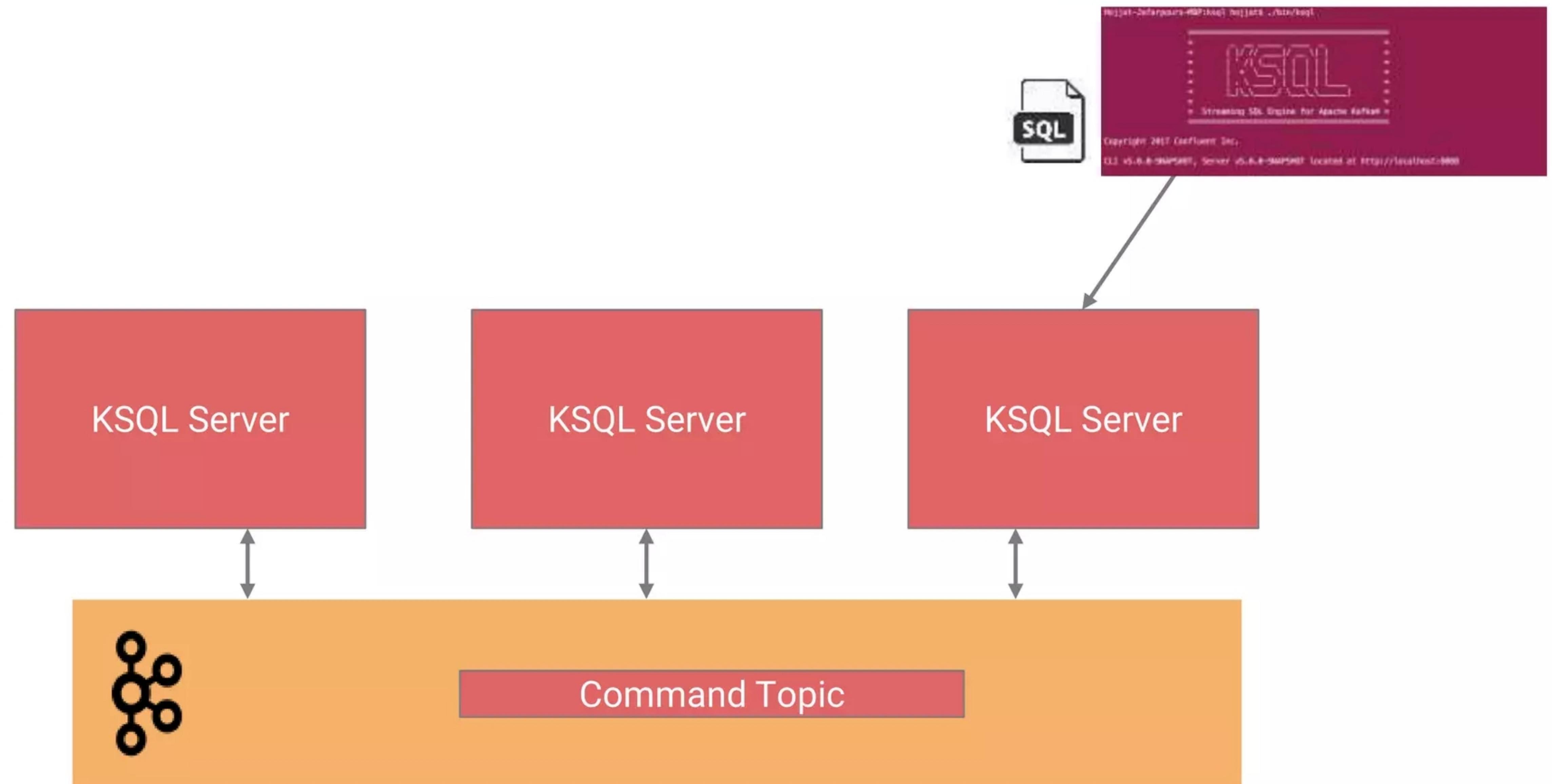
- Interactive Service
 - REST API
 - Command Topic



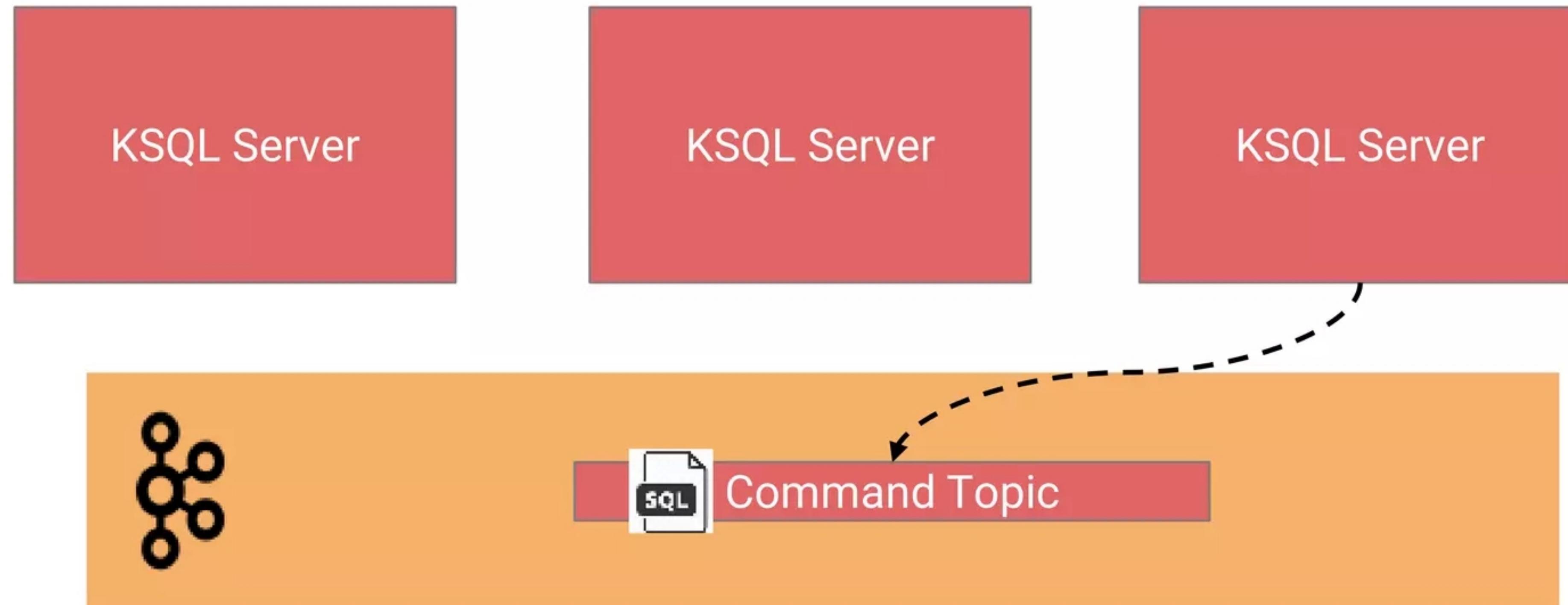
KSQL Deployment



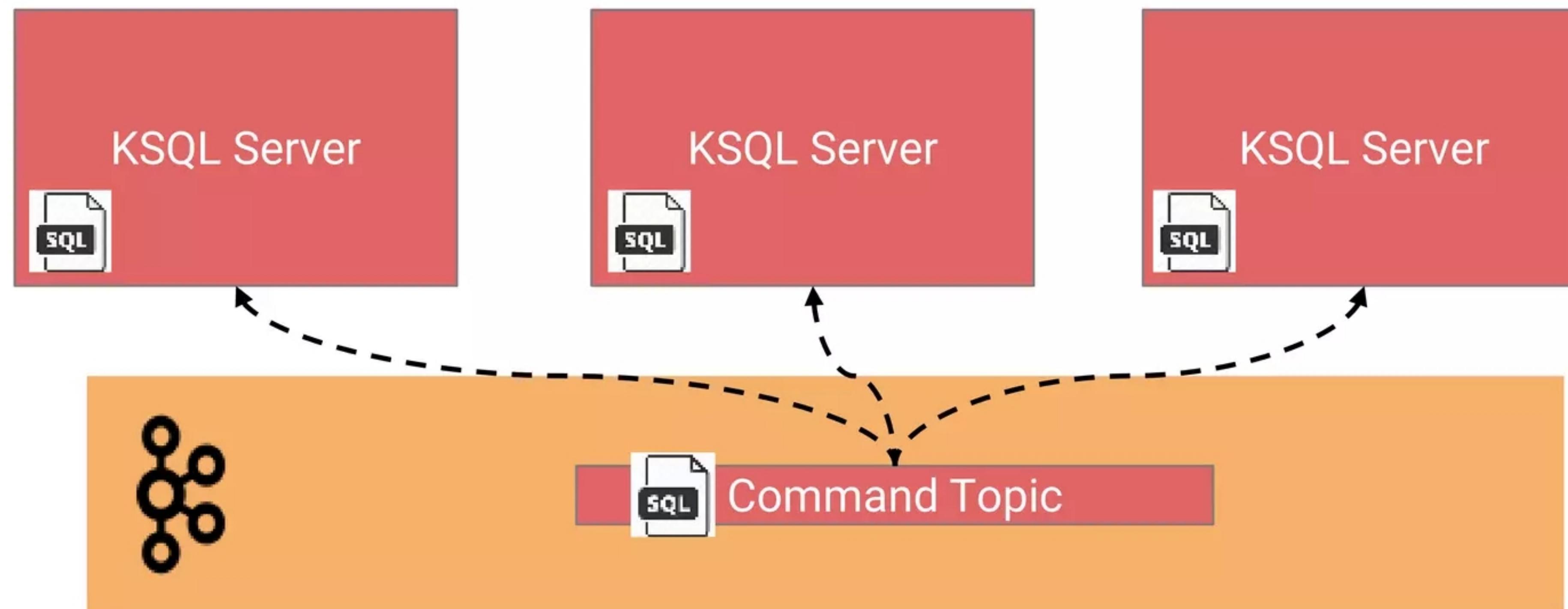
KSQL Deployment



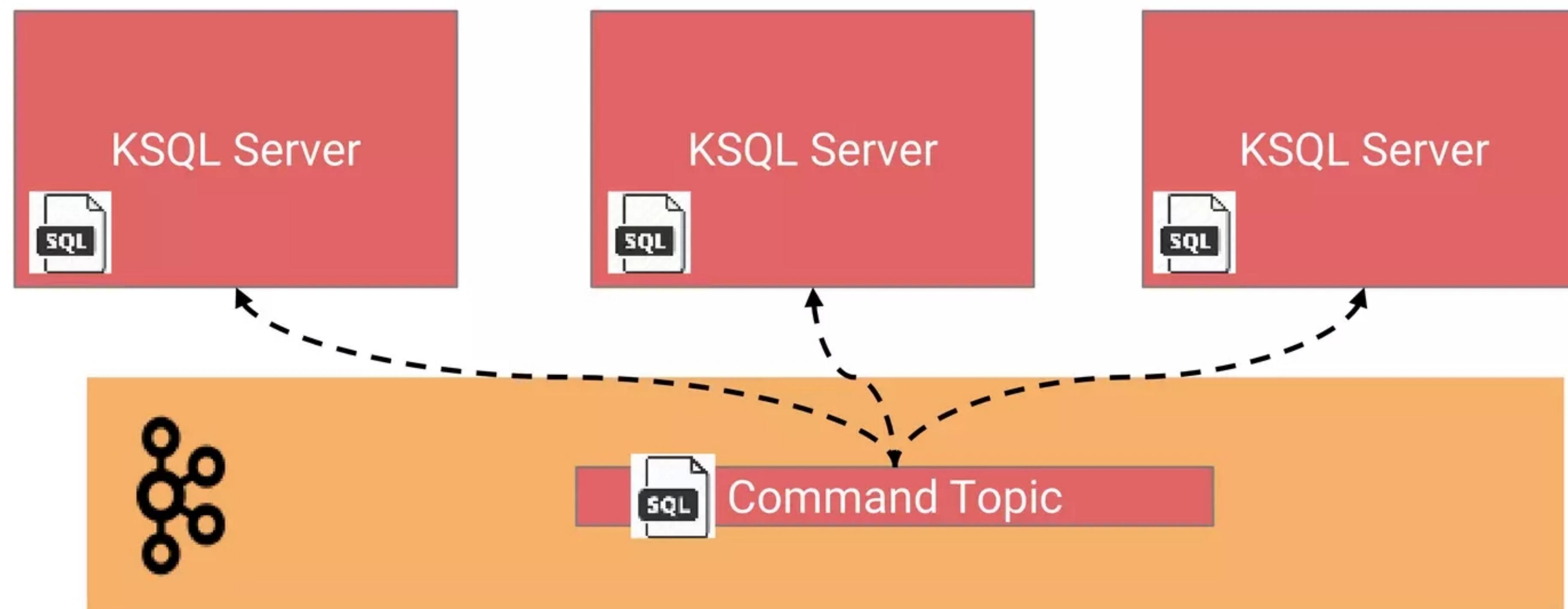
KSQL Deployment



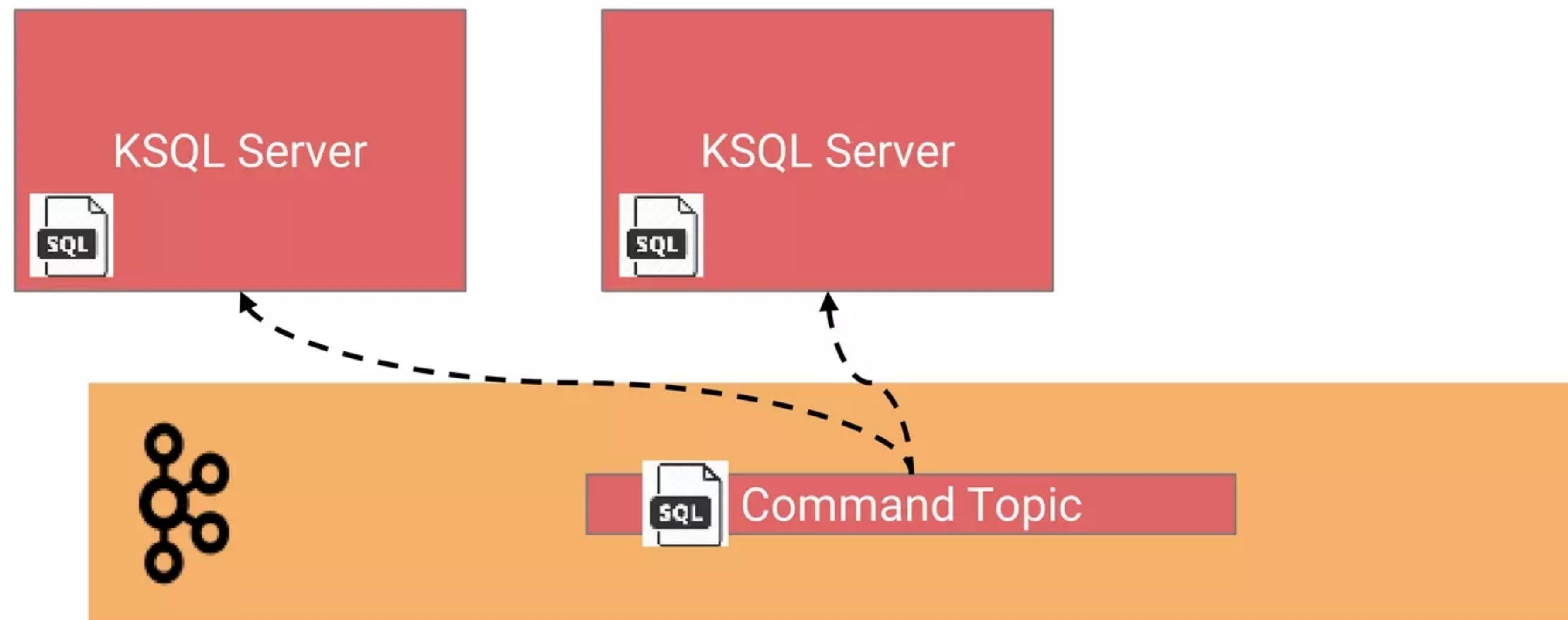
KSQL Deployment



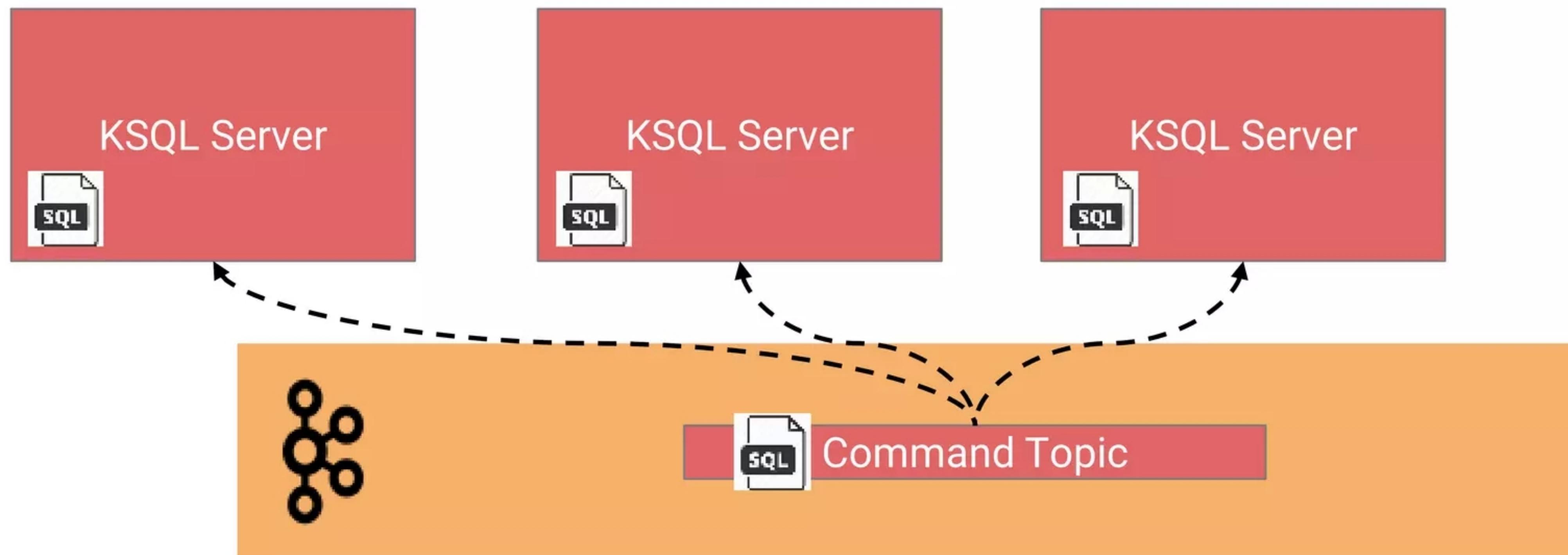
Fault Tolerance:



Fault Tolerance:

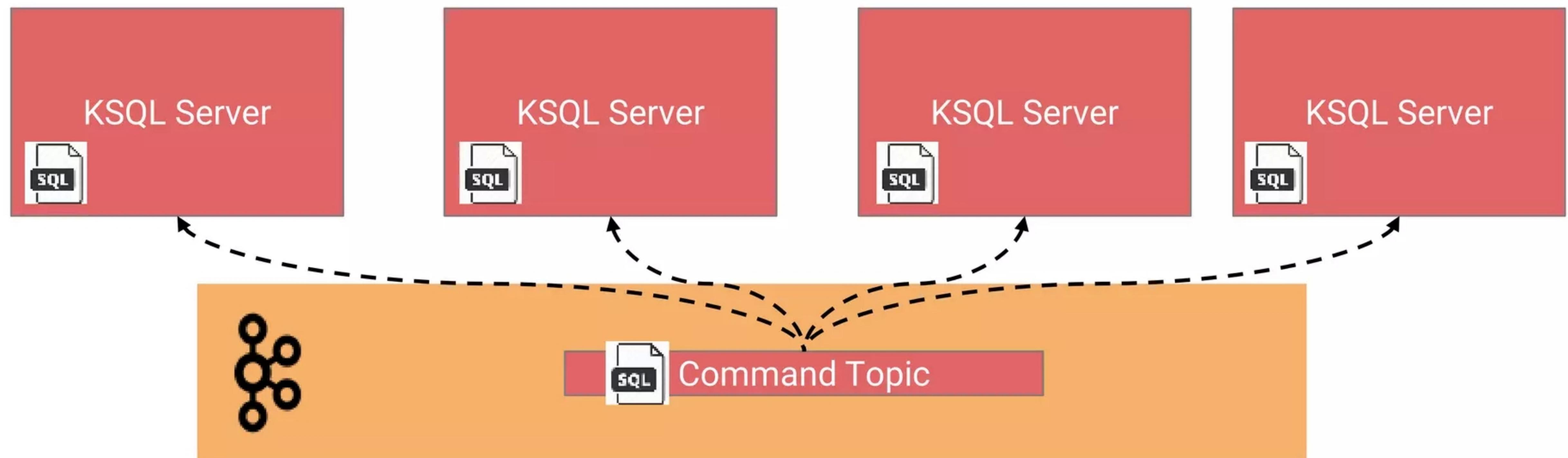


Scale Out:



KSQL Deployment

Scale Out:



How to run KSQL



KSQL Server
(JVM process)

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

Docker images
[confluentinc/cp-ksql-server](https://hub.docker.com/r/confluentinc/cp-ksql-server)
[confluentinc/cp-ksql-cli](https://hub.docker.com/r/confluentinc/cp-ksql-cli)



Physical



docker



kubernetes

 openstack.  vmware®

 Microsoft
Azure



Google Cloud Platform



amazon
web services

...and many more...

Resources and Next Steps

<http://confluent.io/ksql>



<http://cnfl.io/slack>



[confluentinc/ksql](#)

