Un'introduzione a Kafka Streams e KSQL and why they matter!

ITOUG Tech Day Roma
1 Febbraio 2018

confluent

ITOUG
ITALIAN ORACLE USER GROUP

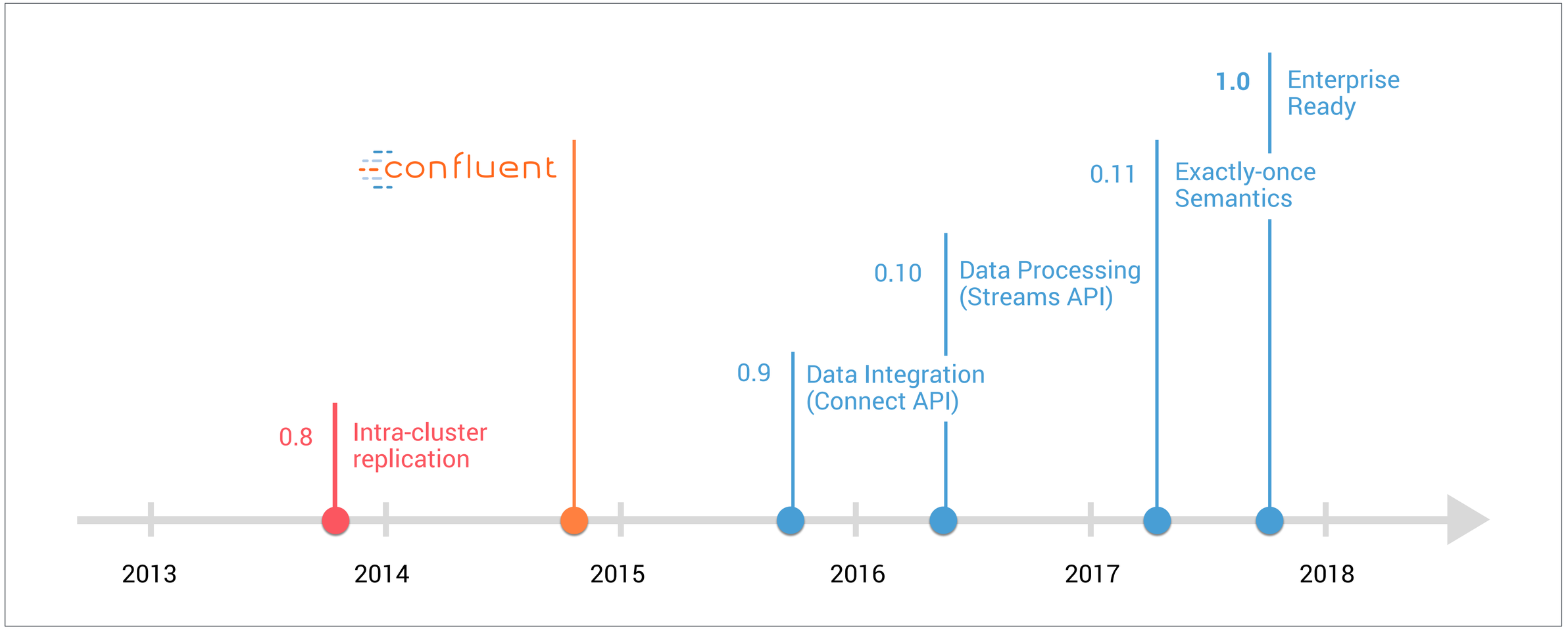
RETHINKING

Stream Processing

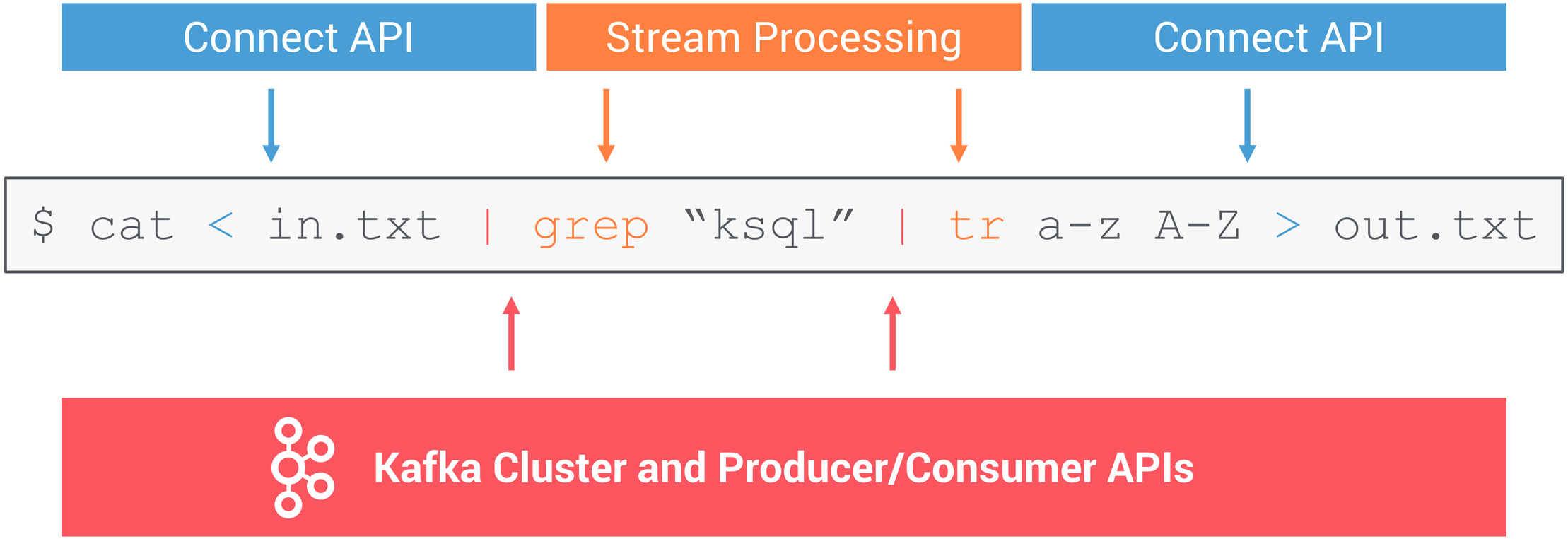
with Apache Kafka

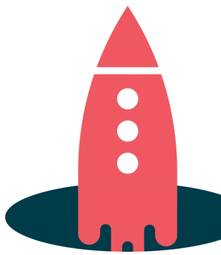
confluent

Kafka the Streaming Data Platform



Apache Kafka APIs: UNIX analogy



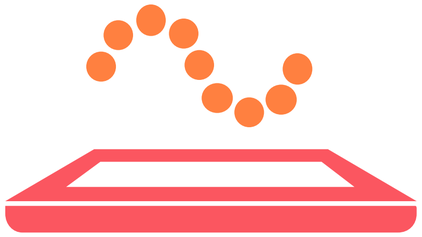


As developers, we want

to build **APPS** not **INFRASTRUCTURE**

We want our apps to be:

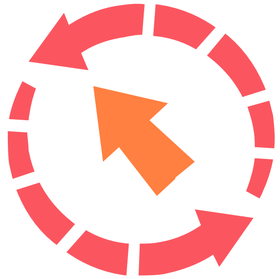
Scalable



Elastic

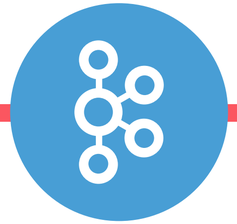
Fault-tolerant

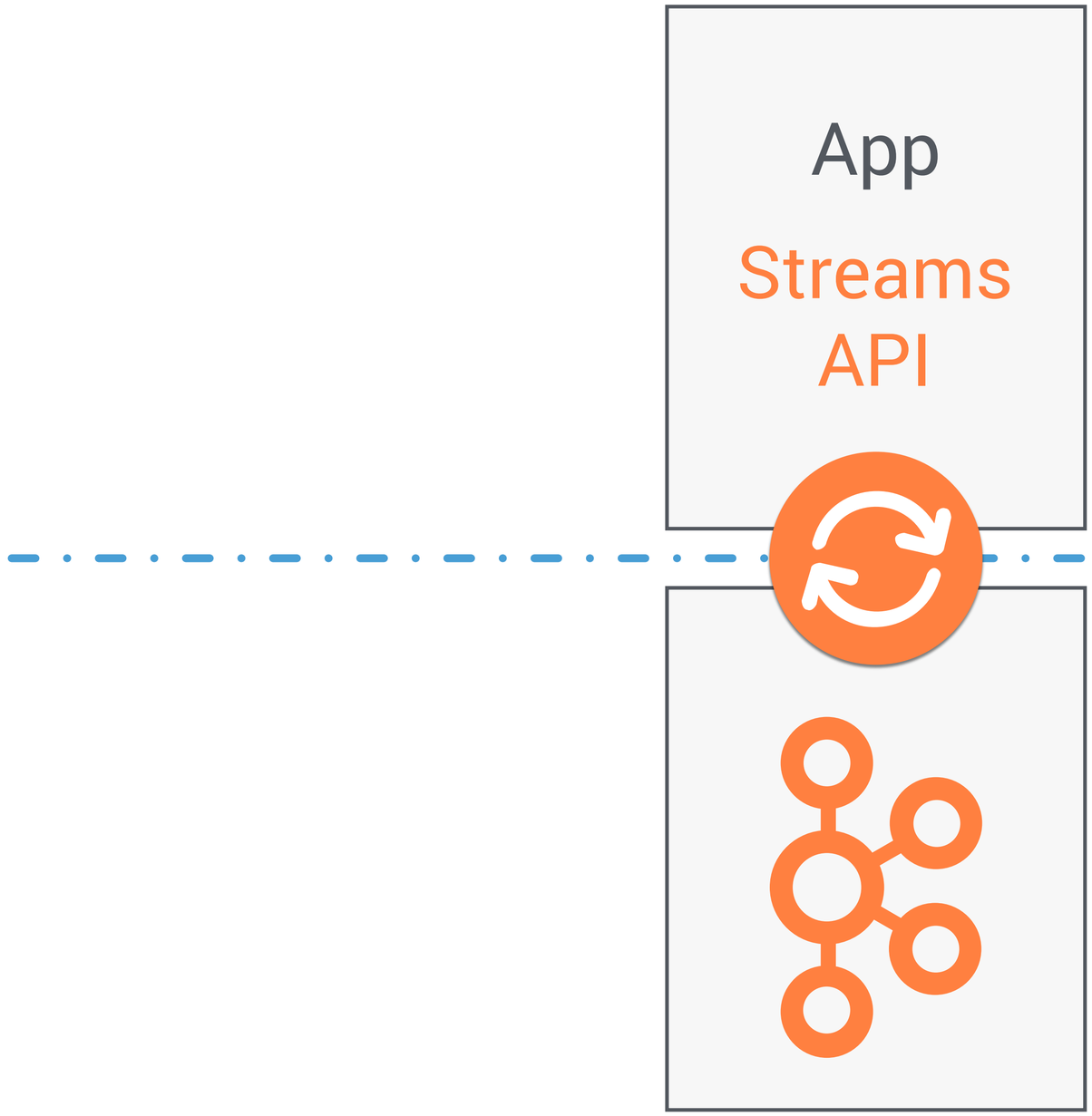
Stateful



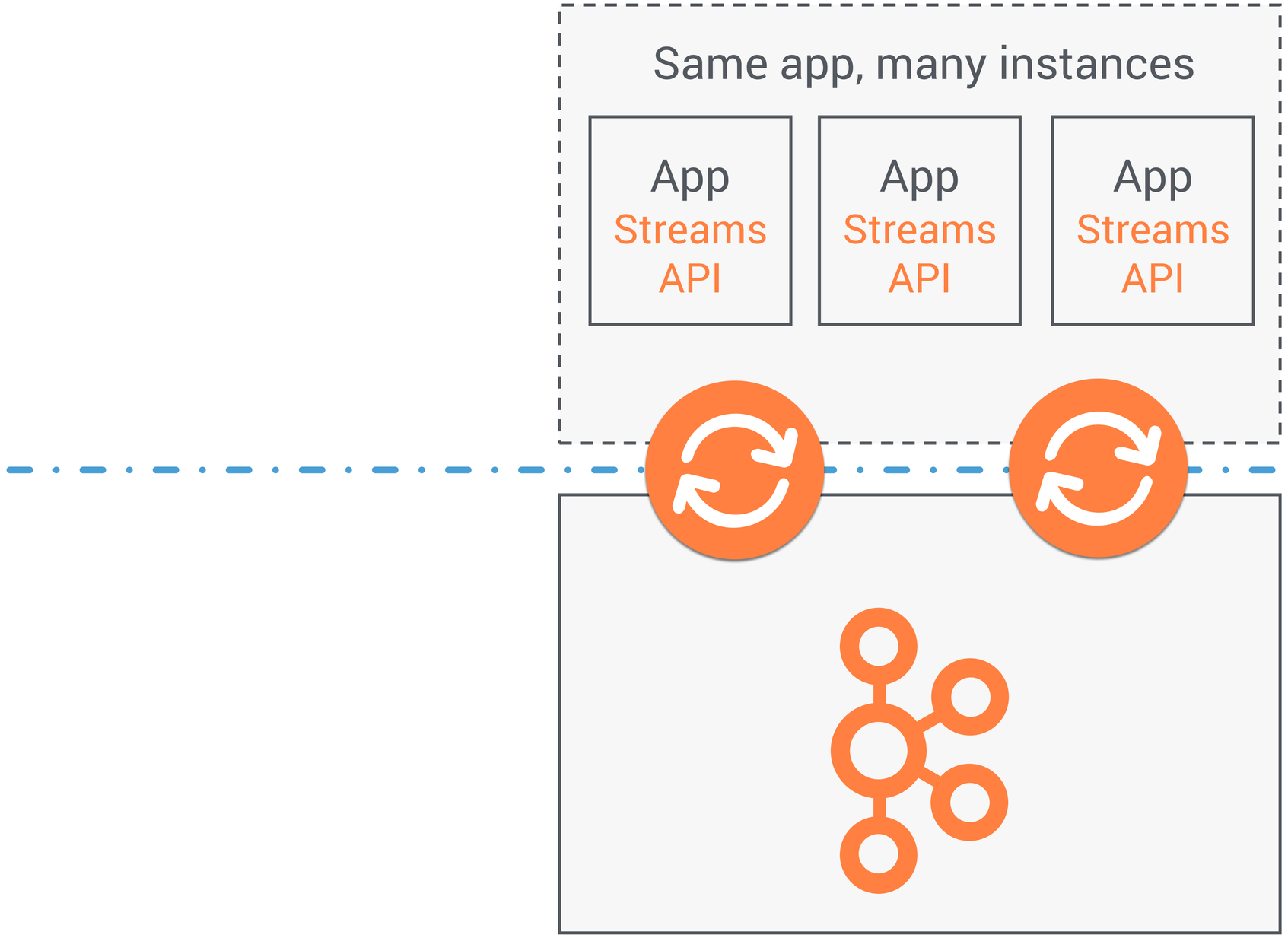
Distributed

the **KAFKA STREAMS API** is a
JAVA API to
BUILD REAL-TIME APPLICATIONS to
POWER THE BUSINESS



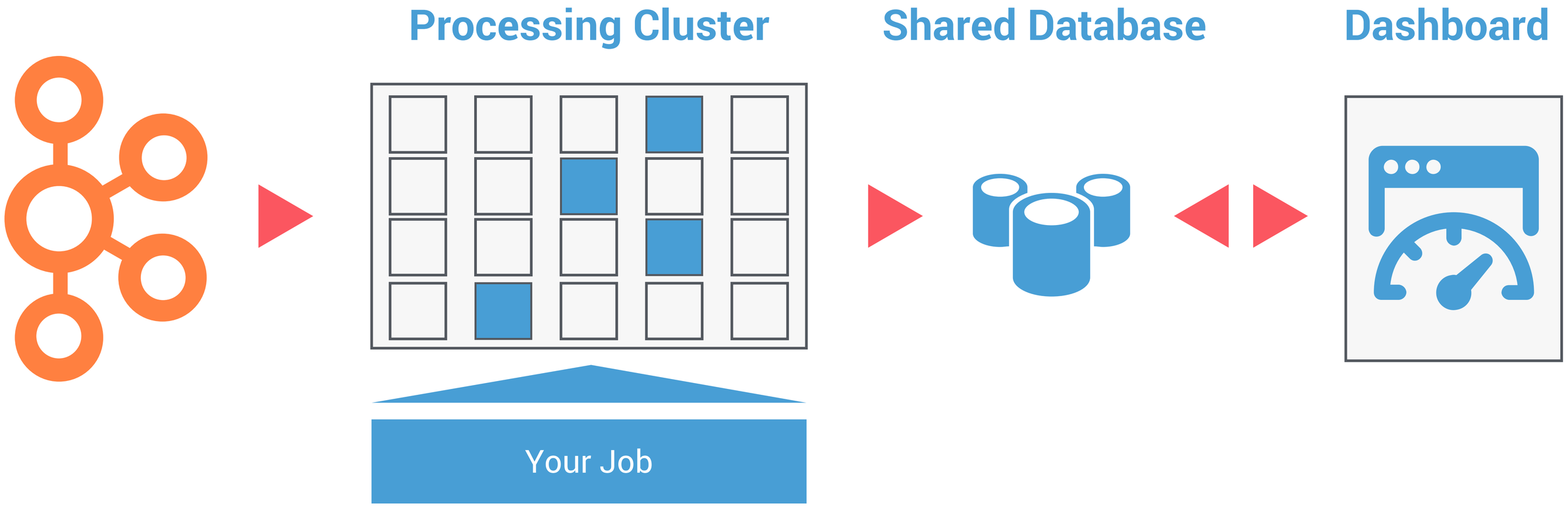


Not running inside brokers!

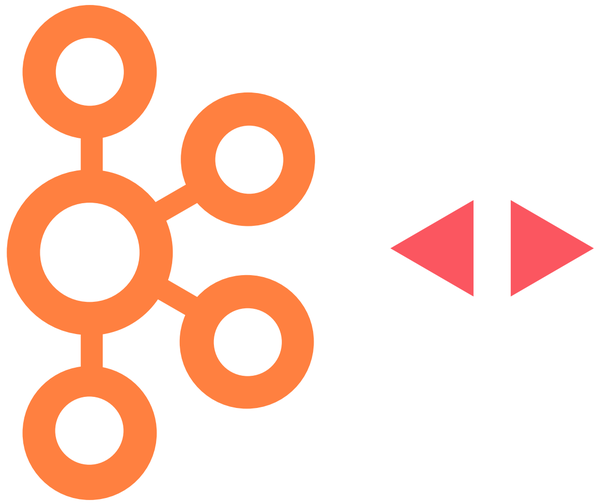


Brokers?
Nope!

Before



After



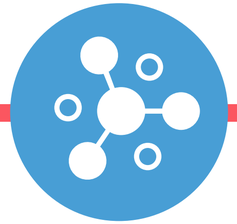
Dashboard

APP

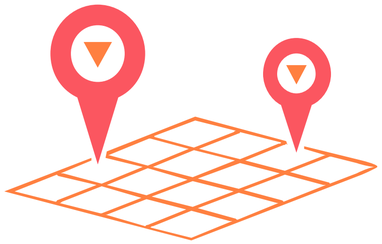
Streams API

this means you can

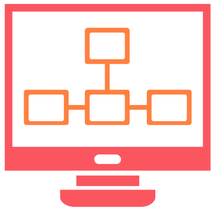
DEPLOY your app ANYWHERE using WHATEVER TECHNOLOGY YOU WANT



Things Kafka Streams Does



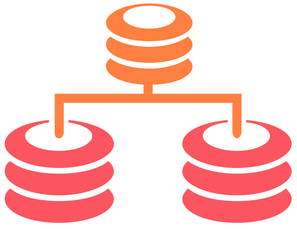
Runs everywhere



Clustering done for you

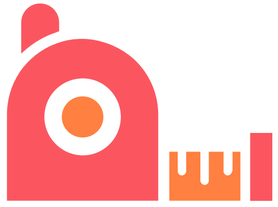
Exactly-once processing

Event-time processing



Integrated database

Joins, windowing, aggregation



S/M/L/XL/XXL/XXXL sizes

KStream

```
// Example: reading data from Kafka
KStream<byte[], String> textLines = builder.stream("textlines-topic", Consumed.with(
                                                    Serdes.ByteArray(), Serdes.String()));

// Example: transforming data
KStream<byte[], String> upperCasedLines= rawRatings.mapValues(String::toUpperCase));
```

KTable

```
// Example: aggregating data
KTable<String, Long> wordCounts = textLines
      .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
      .groupBy((key, word) -> word)
      .count();
```

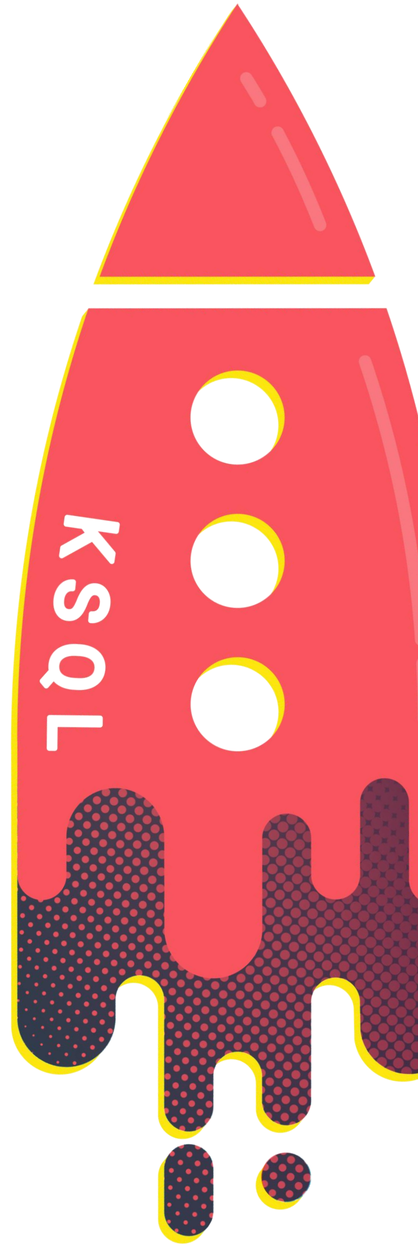
RETHINKING

KSQL

Streaming SQL Engine for Apache Kafka

confluent

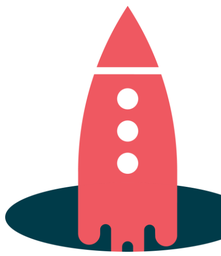
what are some KSQL use cases?



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)';
```

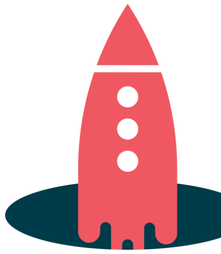


confluent

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.

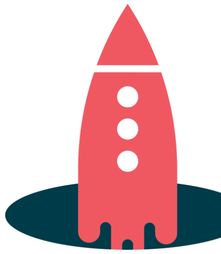
```
CREATE STREAM vip_actions AS
  SELECT userid, page, action FROM clickstream c
    LEFT JOIN users u ON c.userid = u.user_id
    WHERE u.level = 'Platinum';
```



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;
```

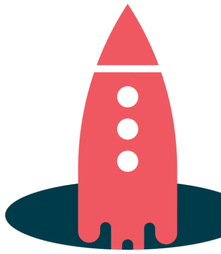


confluent

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;
```

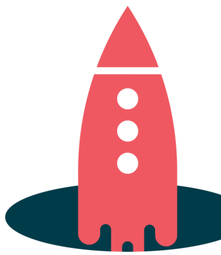


confluent

KSQL for Data Transformation

Make simple derivations of existing topics from the command line

```
CREATE STREAM views_by_userid
  WITH (PARTITIONS=6,
        VALUE_FORMAT='JSON',
        TIMESTAMP='view_time') AS
  SELECT * FROM clickstream PARTITION BY user_id;
```



confluent

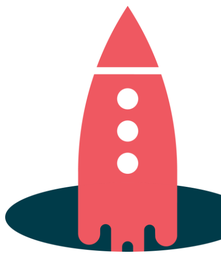
Where is KSQL not such a great fit?

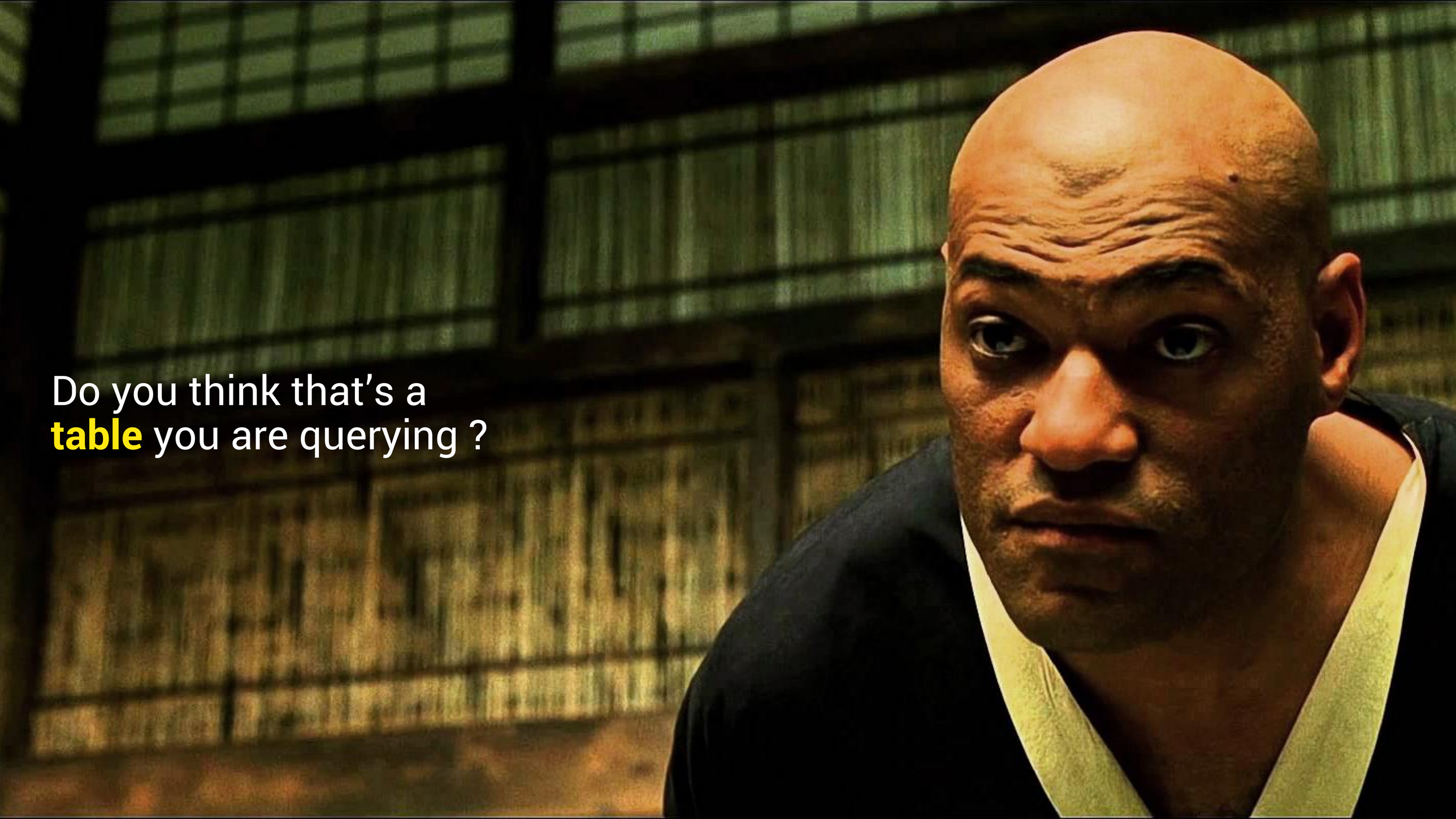
Ad-hoc queries

- 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!)




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

Stream/Table Duality

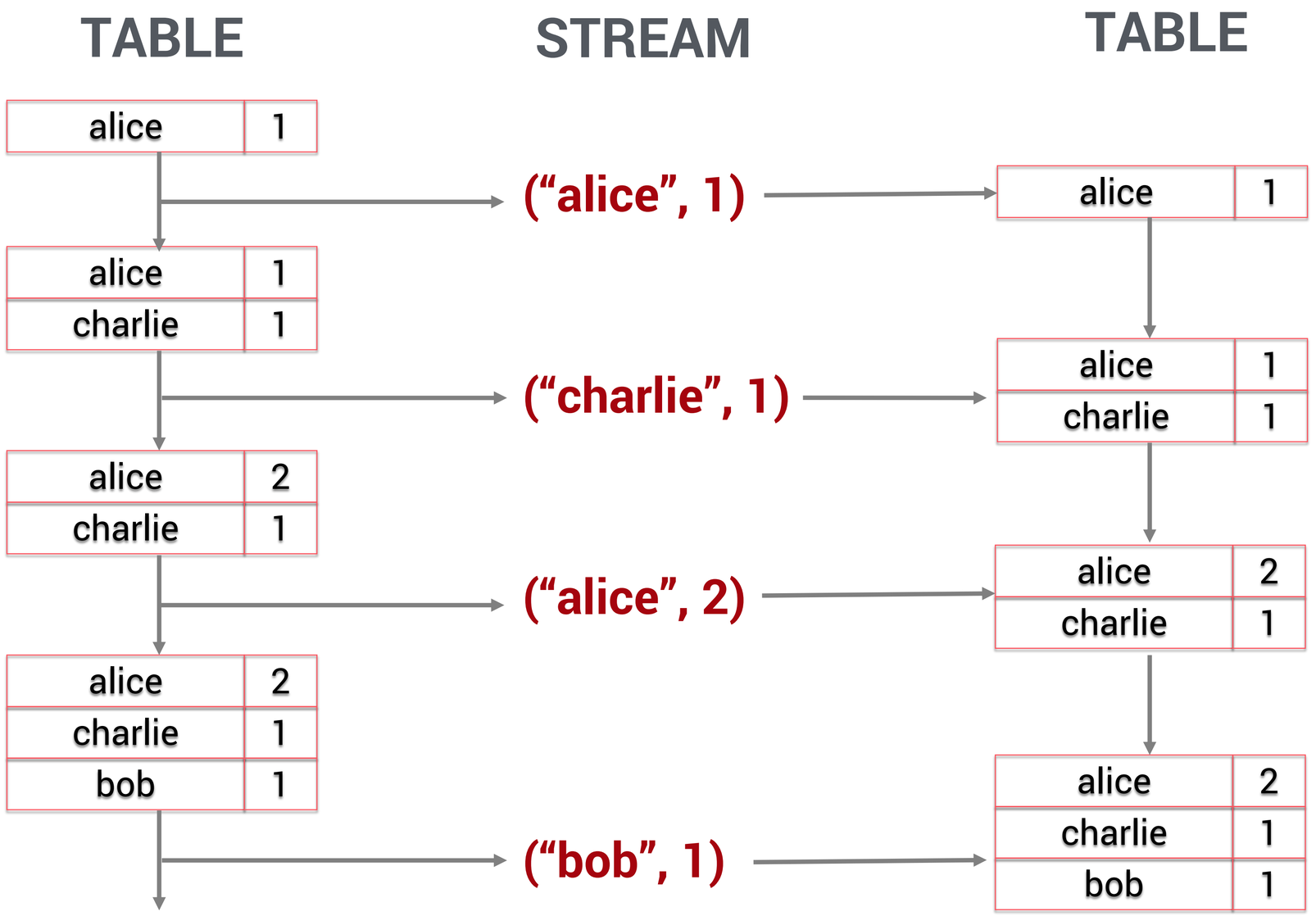
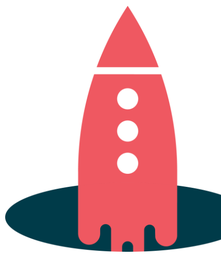


TABLE
STREAM
TABLE
alice 1
alice 1
charlie 1
alice 2
charlie 1
alice 2
charlie 1
bob 1
("alice", 1)
("charlie", 1)
("alice", 2)
("bob", 1)
alice 1
alice 1
charlie 1
alice 2
charlie 1
alice 2
charlie 1
bob 1

Creating a Stream

```
CREATE STREAM clickstream (
  time    BIGINT,
  url     VARCHAR,
  status  INTEGER,
  bytes   INTEGER,
  userid VARCHAR,
  agent   VARCHAR)
WITH (
  value_format = 'JSON',
  kafka_topic='my_clickstream_topic'
);
```

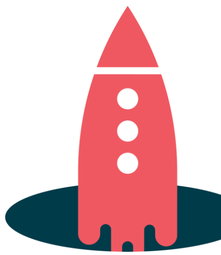
confluent



Creating a Table

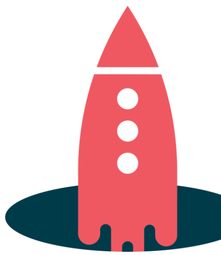
```
CREATE TABLE users (
  user_id         INTEGER,
  registered_at LONG,
  username        VARCHAR,
  name            VARCHAR,
  city            VARCHAR,
  level           VARCHAR)
WITH (
  key = 'user_id',
  kafka_topic='clickstream_users',
  value_format='JSON');
```

confluent

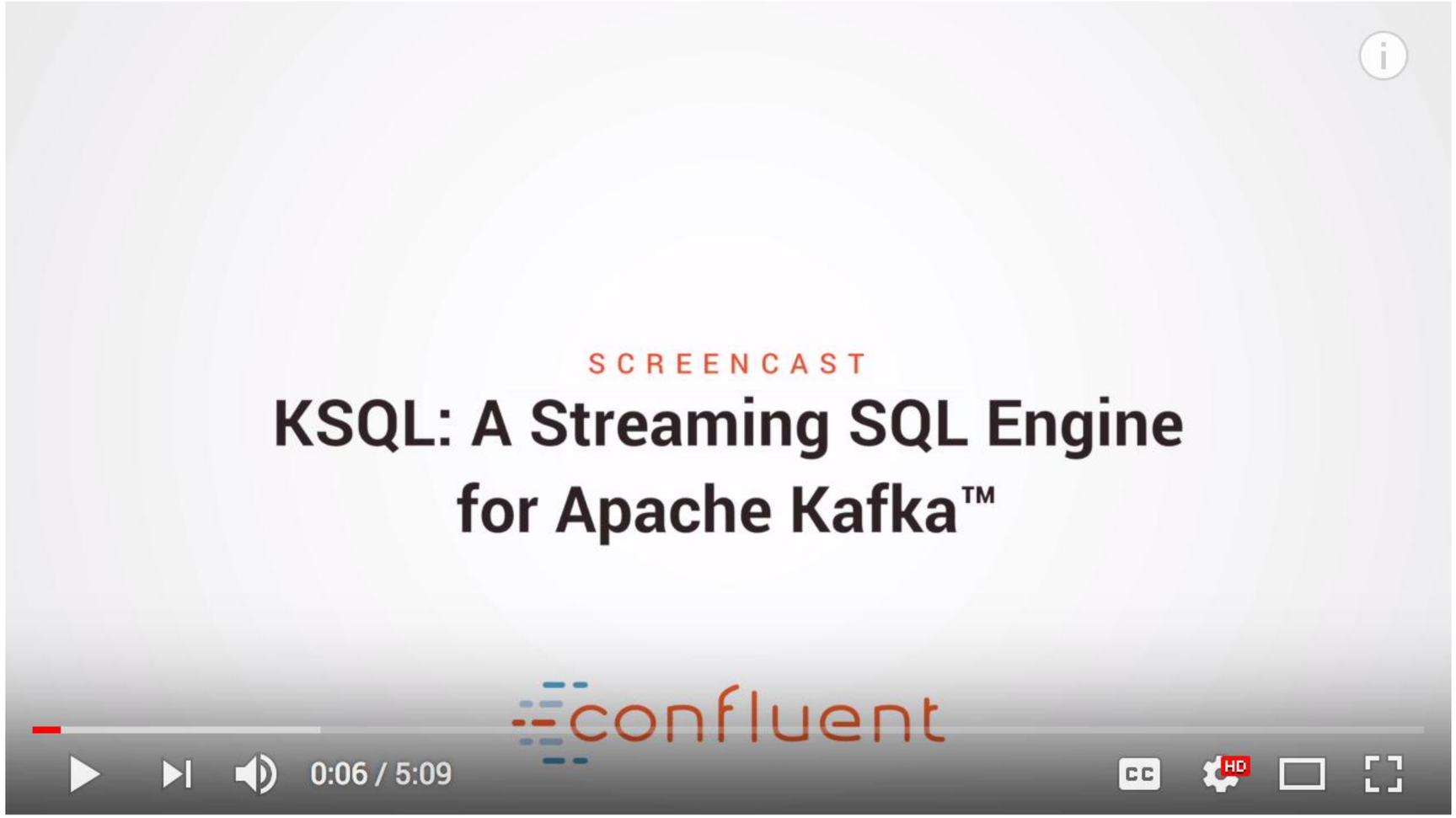


Joins for Enrichment

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



KSQL in less than 5 minutes

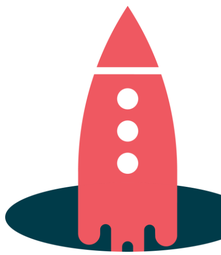


https://www.youtube.com/watch?v=A45uRzJiv7I

Trade-Offs

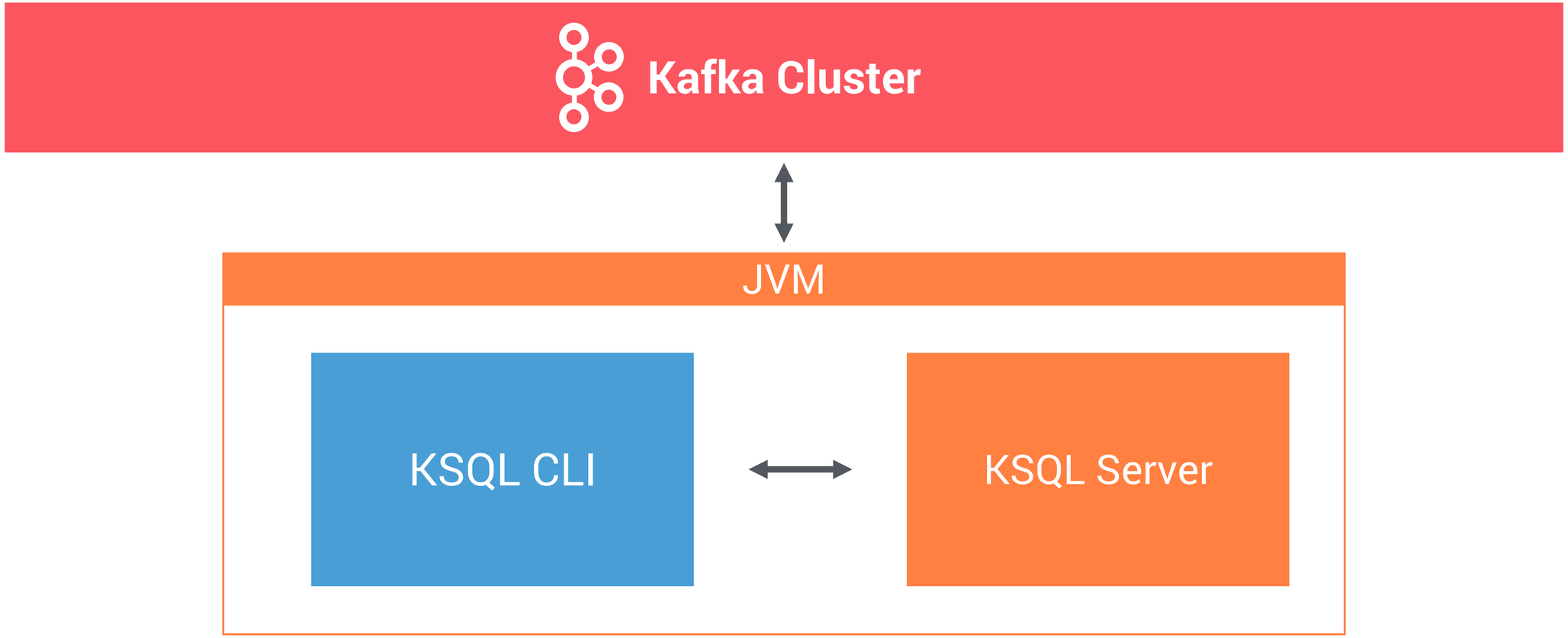
Flexibility ⟷ Simplicity

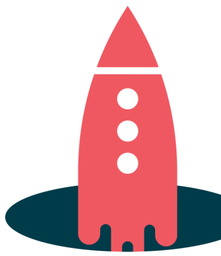
Consumer, Producer	Kafka Streams	KSQL
• subscribe() • poll() • send() • flush()	• filter() • join() • aggregate()	• Select…from… • Join…where… • Group by..



How to run KSQL

#1 Stand-alone aka ‘local mode’





How to run KSQL

#1 Stand-alone aka ‘local mode’

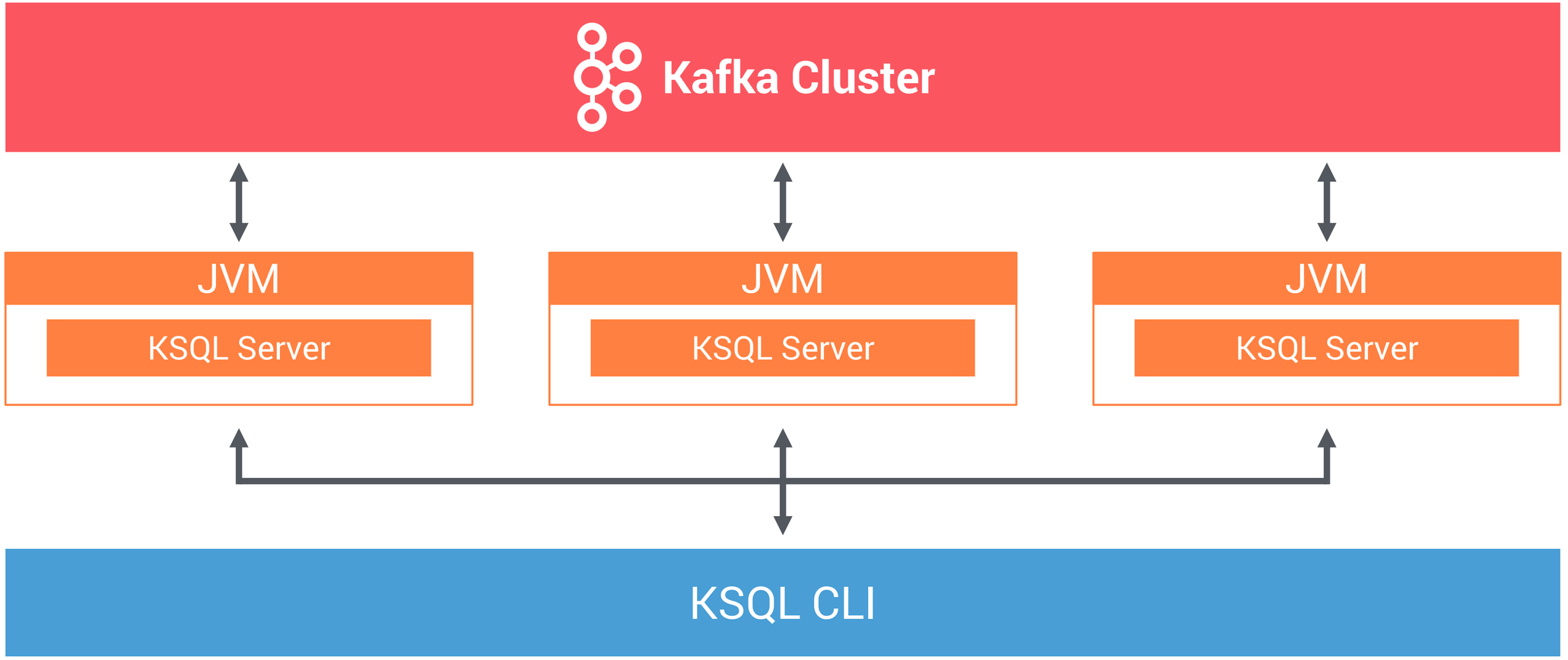
- **Starts a CLI and a server in the same JVM**
- **Ideal for developing on your laptop**

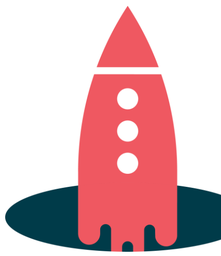
  bin/ksql-cli local
- **Or with customized settings**

  bin/ksql-cli local --properties-file ksql.properties

How to run KSQL

#2 Client-server





How to run KSQL

#2 Client-server

- **Start any number of server nodes**

  bin/ksql-server-start

- **Start one or more CLIs and point them to a server**

  bin/ksql-cli remote https://myksqlserver:8090

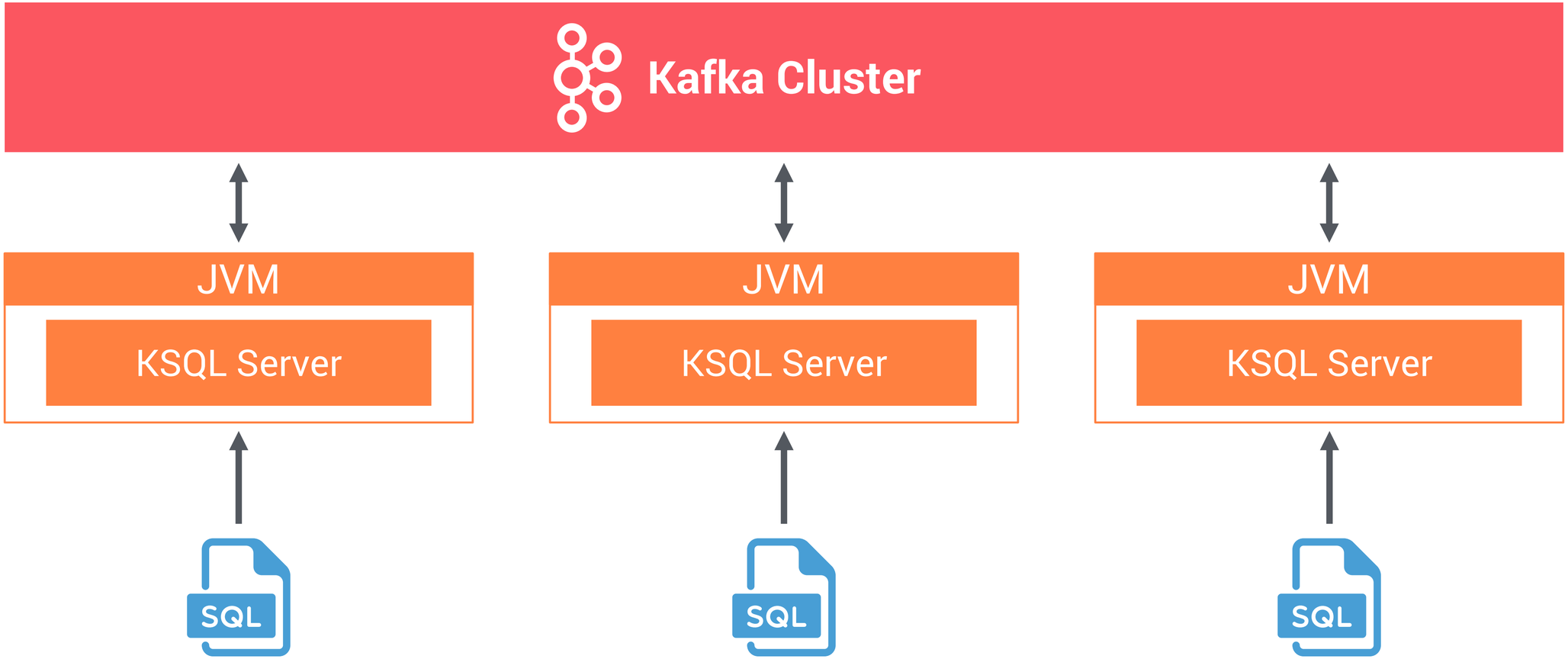
- **All servers share the processing load**

  Technically, instances of the same Kafka Streams Applications

  Scale up/down without restart

How to run KSQL

#3 as a standalone Application



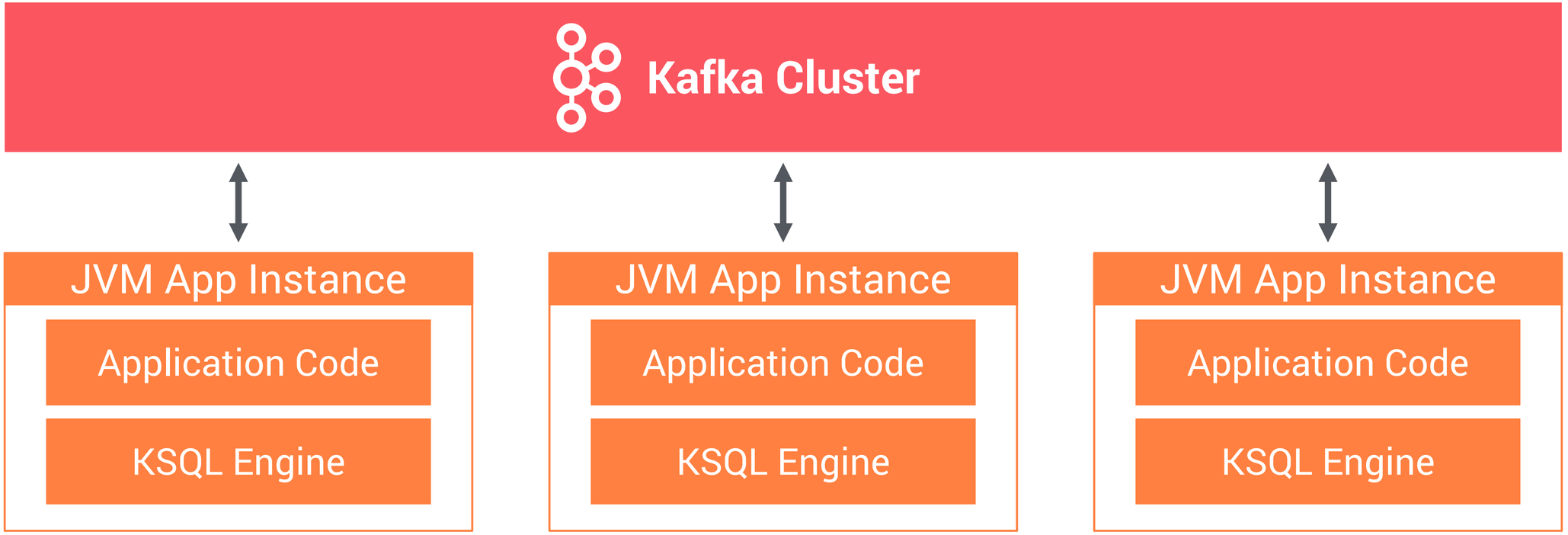
How to run KSQL

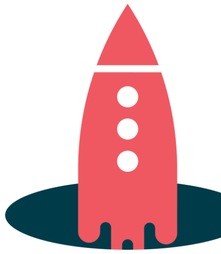
#3 as a standalone Application

- **Start any number of server nodes**
  - **Pass a file of KSQL statement to execute**
  - bin/ksql-node query-file=foo/bar.sql
- **Ideal for streaming ETL application deployment**
  - **Version-control your queries and transformations as code**
- **All running engines share the processing load**
  - **Technically, instances of the same Kafka Streams Applications**
  - **Scale up/down without restart**

How to run KSQL

#4 EMBEDDED IN AN APPLICATION





How to run KSQL

#4 EMBEDDED IN AN APPLICATION

- Embed directly in your Java application
- Generate and execute KSQL queries through the Java API
  - Version-control your queries and transformations as code
- All running application instances share the processing load
  - Technically, instances of the same Kafka Streams Applications
  - Scale up/down without restart

Remember: Developer Preview!

BE EXCITED, BUT BE ADVISED

Caveats

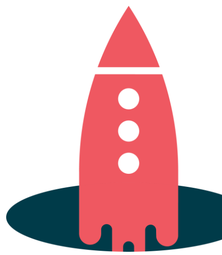
- No Stream-stream joins yet
- Limited function library
- No Avro support yet
- Breaking API/Syntax changes still possible

Resources and Next Steps

https://github.com/confluentinc/ksql

http://confluent.io/ksql

https://slackpass.io/confluentcommunity #ksql



confluent

Thank you!

confluent