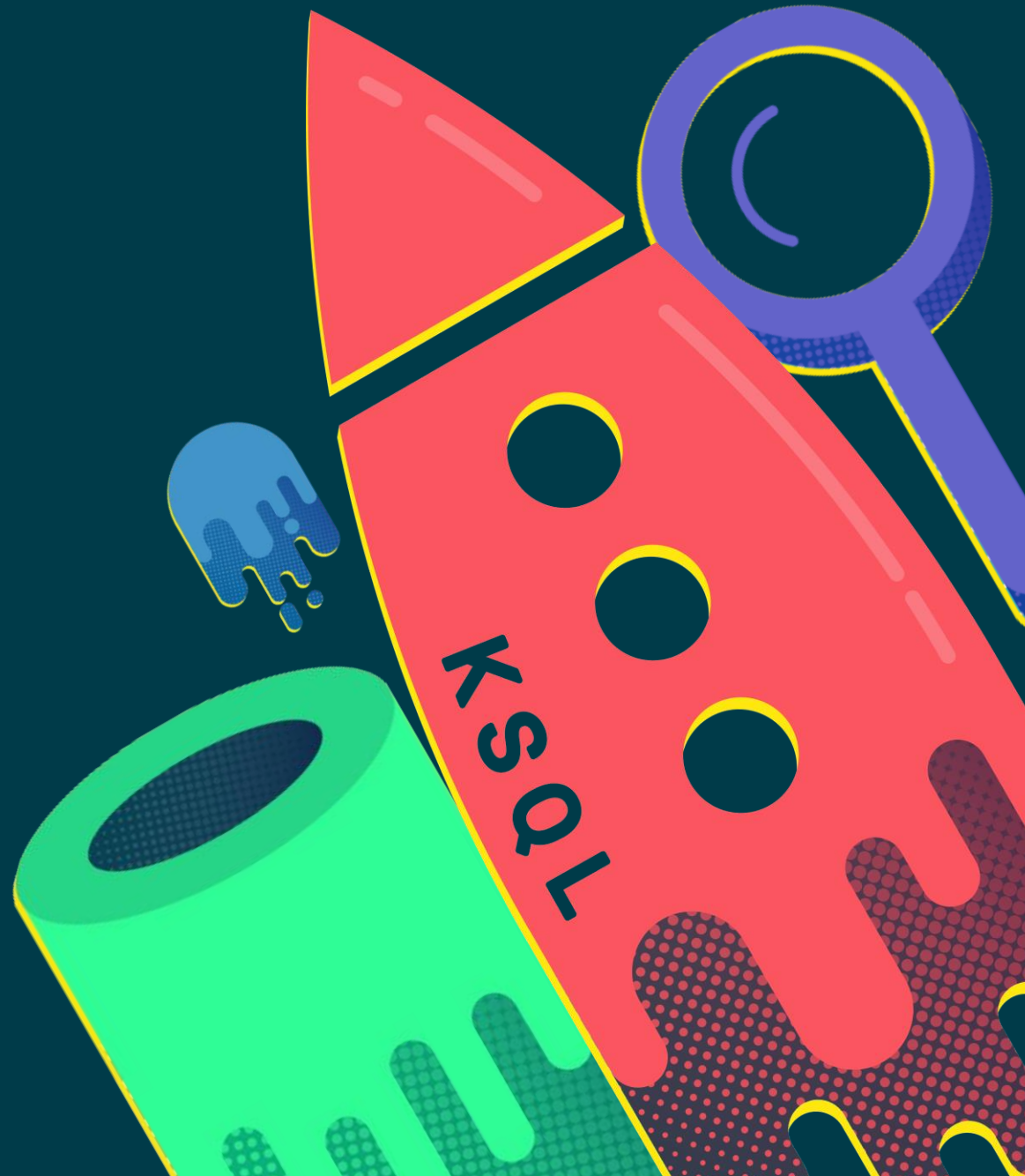
Stream Processing with Kafka & KSQL



Nick Dearden
April 2020

Today's Agenda

10:00 - 10:45 AM

Streams Processing/KSQL Overview
Nick Dearden, Director Advanced Technologies, Confluent

10:45 AM - 12:15 PM

Interactive Streams Lab
Nick Dearden, Director Advanced Technologies, Confluent
Brian Likosar, Solutions Engineer, Confluent

12:15 - 12:30 PM

Q&A and Next Steps
Open Discussion

Workshop Tips & Help:

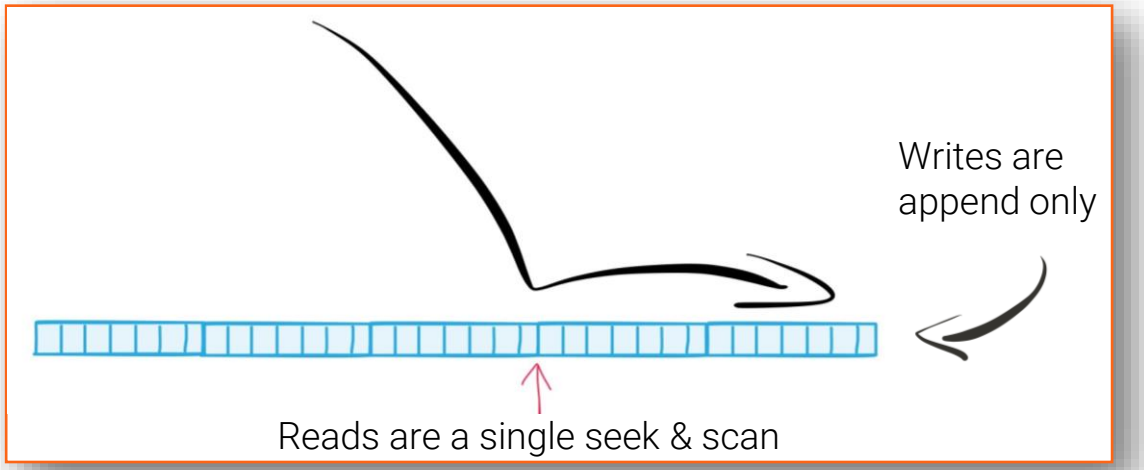
1. Your username to log in is:
   first initial last name
   [all one word, lowercase]

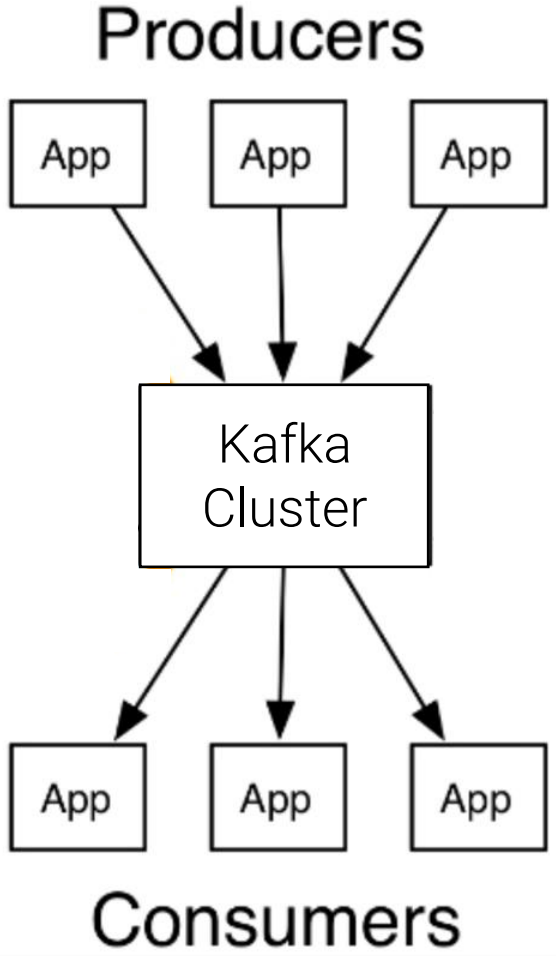
2. Check the **'Chat' window** during the session for instructions
   [icon located at the bottom of the Zoom toolbar]

3. For any technical issues, click the **'Raise Hand' button** or post in the **'Chat' window**
   [a Confluent team member will assist you]

Kafka

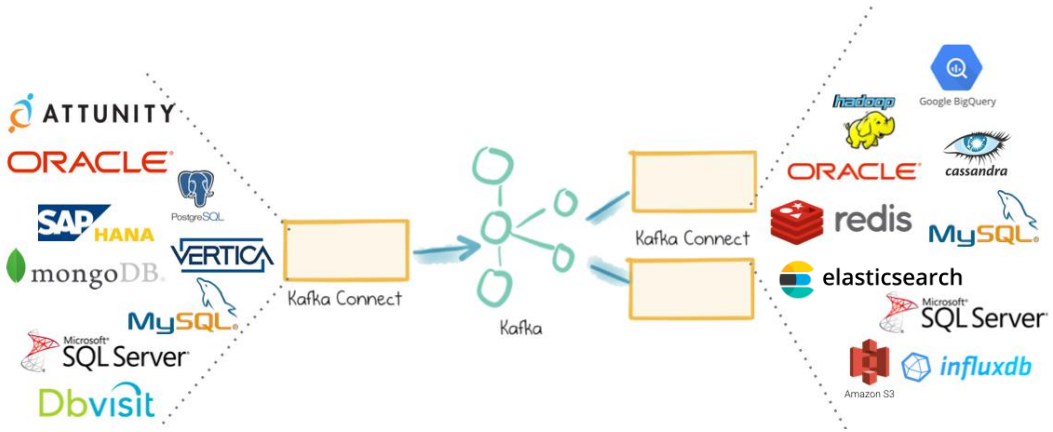
A **Distributed Commit Log**. Publish and subscribe to streams of records. Highly scalable, high throughput. Supports transactions. Persisted data.





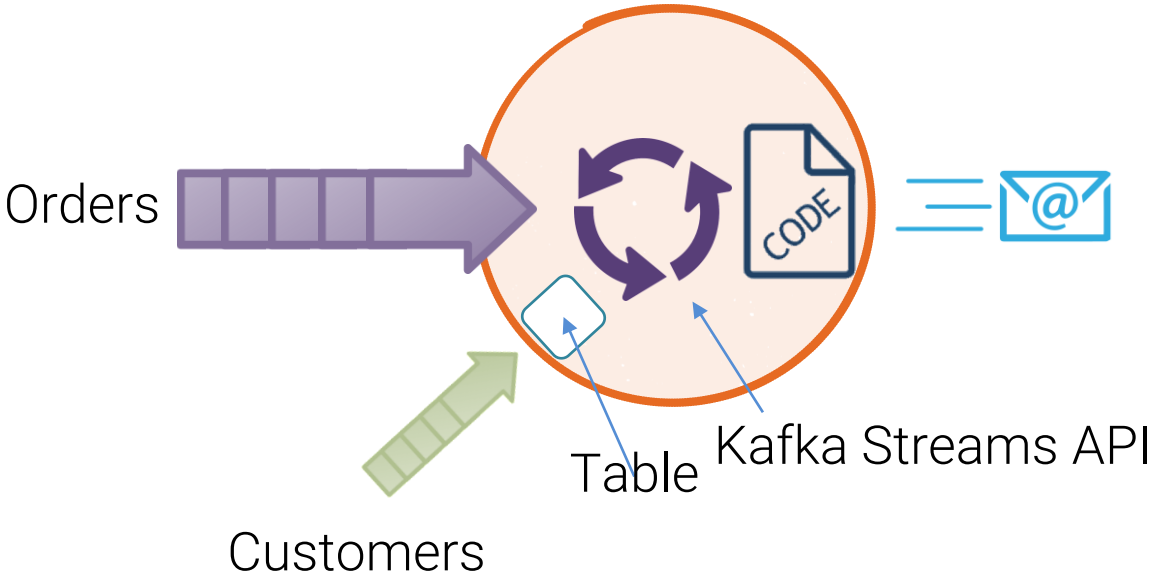
Kafka Connect API

Reliable and scalable integration of Kafka with other systems – no coding required.



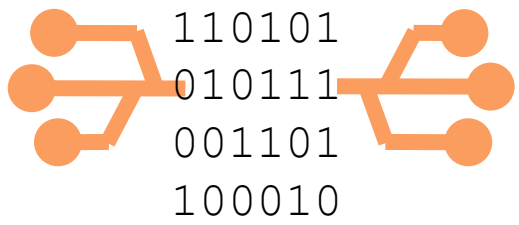
Kafka Streams API

Write standard Java applications & microservices to process your data in real-time



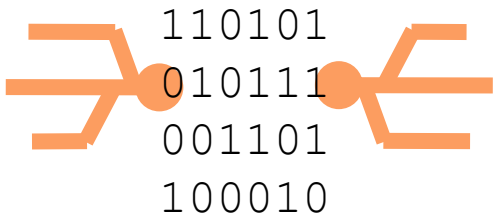
What does a streaming platform do?

Publish and subscribe to streams of data



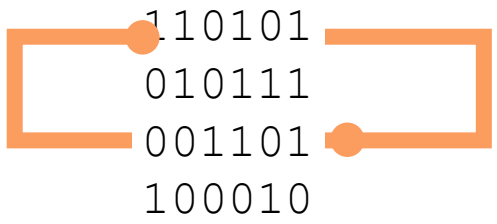
similar to a message queue or enterprise messaging system.

Store streams of data



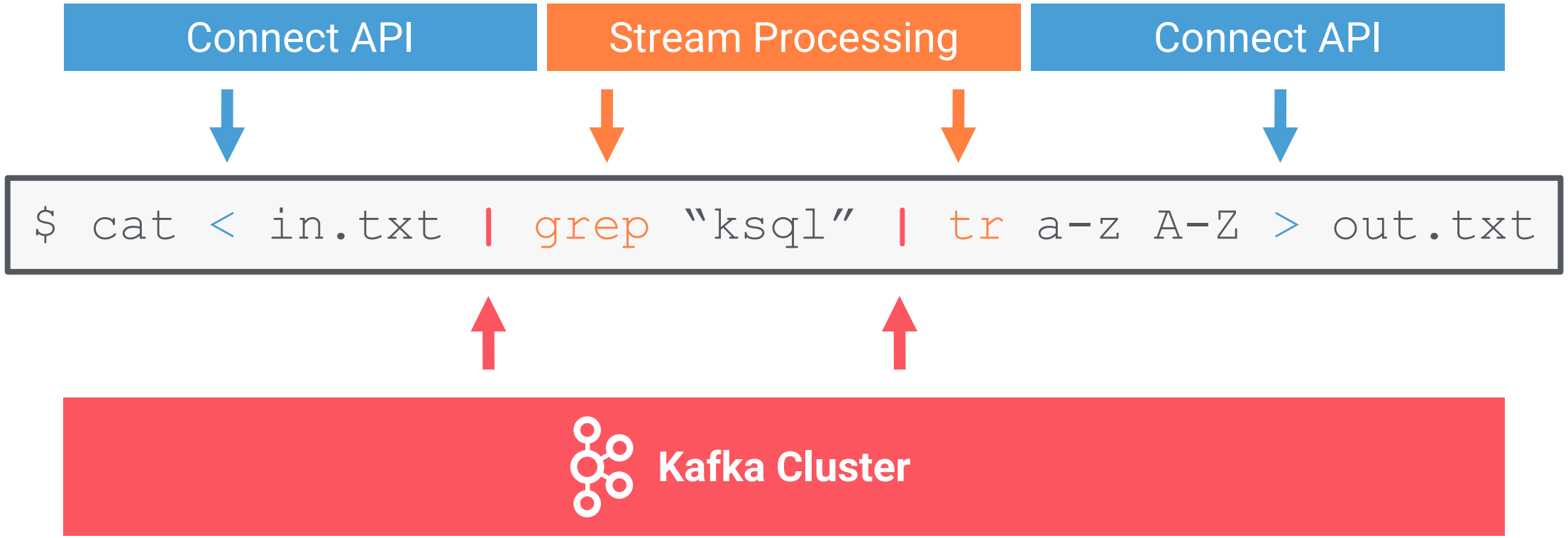
in a durable, fault-tolerant way.

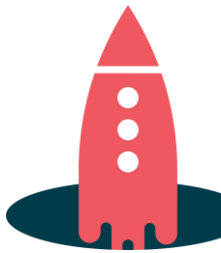
Process streams of data



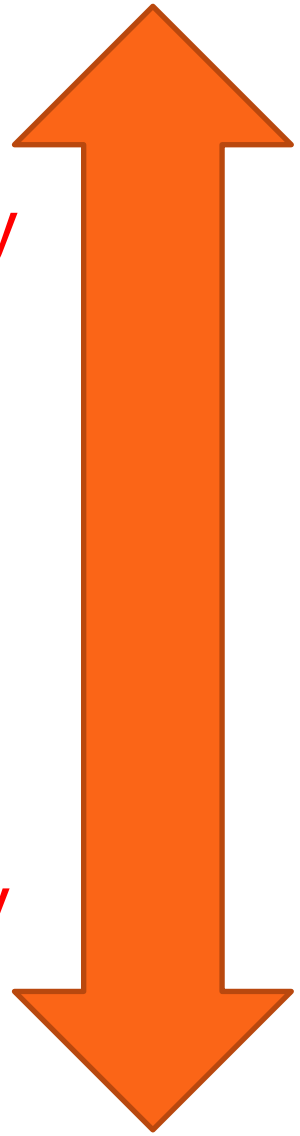
in real time, as they occur.

Stream Processing by Analogy





Client Trade-offs



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

Stream processing with Kafka

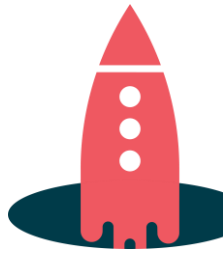
```
object FraudFilteringApplication extends App {
  val builder: StreamsBuilder = new StreamsBuilder()

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

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

Example: Using **Kafka's Streams API** for writing elastic, scalable, fault-tolerant Java and Scala applications

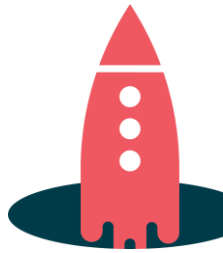
confluent



Stream processing with Kafka

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

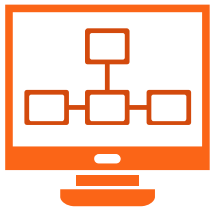
Same example, now with **KSQL**.
Not a single line of Java or Scala code needed.



confluent

Kafka Streams & KSQL - The Easiest Way to Process Data Streams

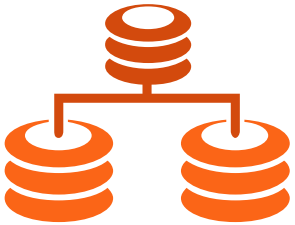
Runs everywhere



Clustering done for you

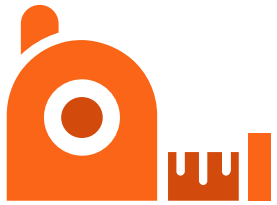
Exactly-once processing

Event-time processing



Integrated database

Joins, windowing, aggregation



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

What is it for ?

Streaming ETL / Enrichment

```
CREATE STREAM clicks_with_city AS
SELECT c.*, u.city
FROM clickstream c
LEFT JOIN users u ON c.user_id = u.id;
```

What is it 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;
```

What is it 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;
```

What is it for ?

- Redaction
- Event De-duplication
- Event Re-ordering
- Sessionization
- Validation
- Microservices….


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

Where is KSQL not such a great fit?

Post-fact Ad-hoc queries

- Limited span of time usually retained in Kafka
- No indexes for random lookups

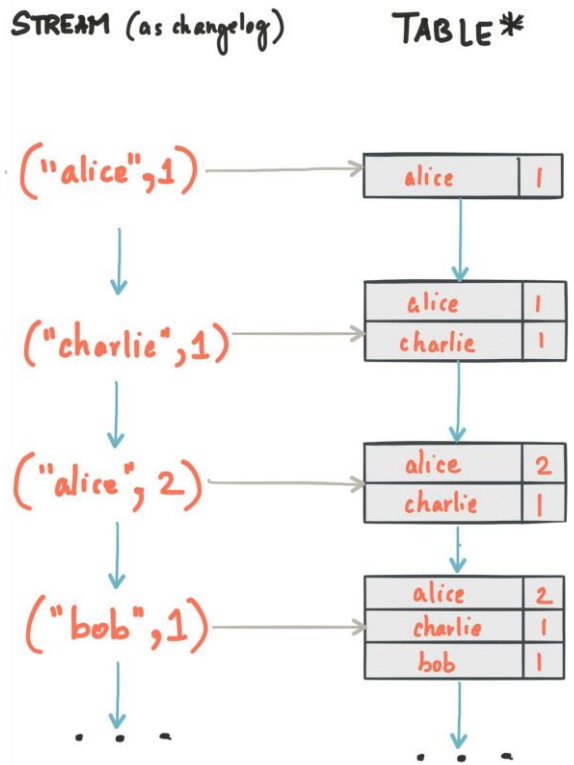
BI reports (Tableau etc.)

- No secondary indexes
- No JDBC (most BI tools are not good with continuous results!)

Stream/Table Duality

Streams & Tables

- STREAM and TABLE as first-class citizens
- Interpretations of topic content
- STREAM - data in motion
- TABLE - collected state of a stream
  - One record per key (per window)
  - Current values (compacted topic)
  - Changelog
- STREAM – TABLE Joins



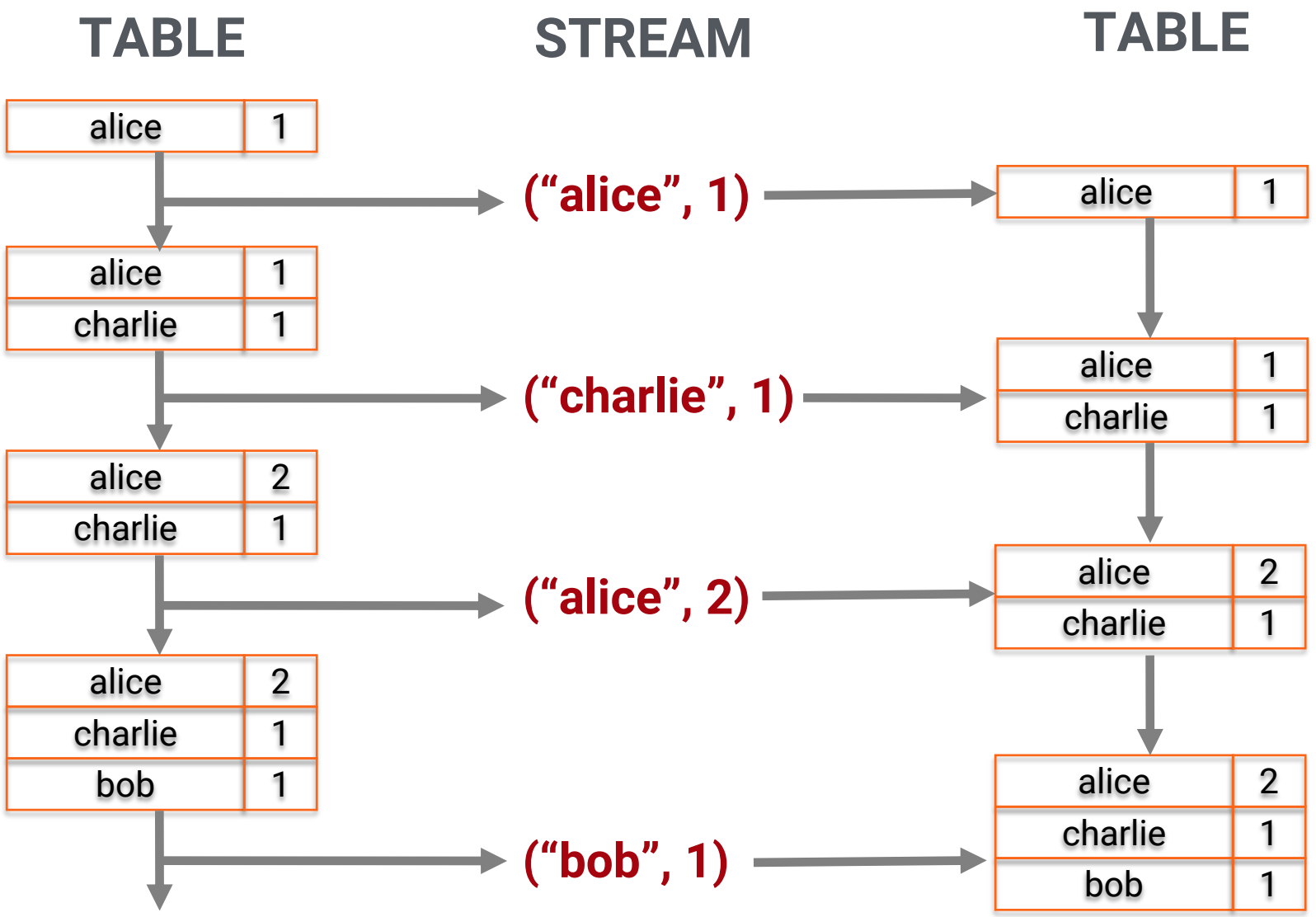
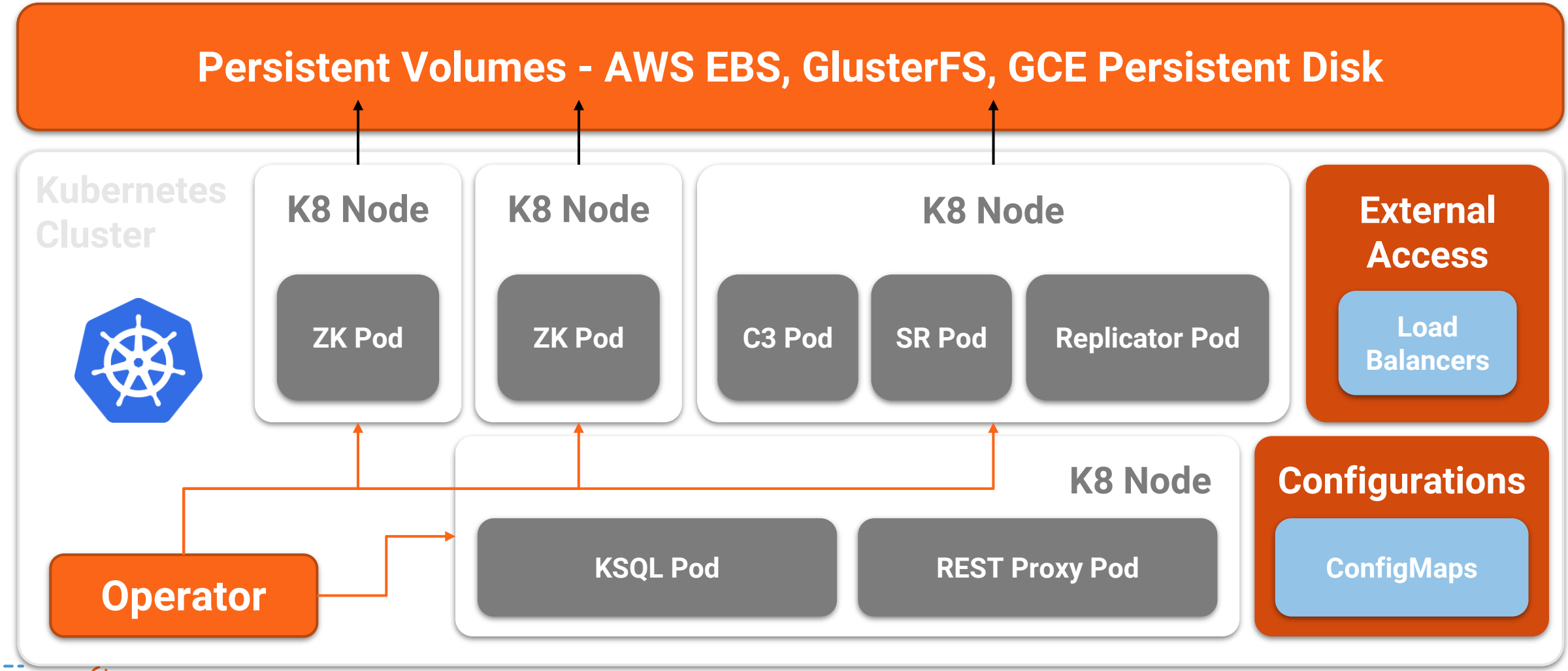


TABLE
STREAM
TABLE
alice 1
alice 1
charlie 1
alice 2
charlie 1
alice 2
charlie 1
bob 1
("alice", 1)
("charlie", 1)
("alice", 2)
("bob", 1)
alice 1
alice 1
charlie 1
alice 2
charlie 1
alice 2
charlie 1
bob 1


HIVE
INBOARD VALVES
RIG FOR DAVE

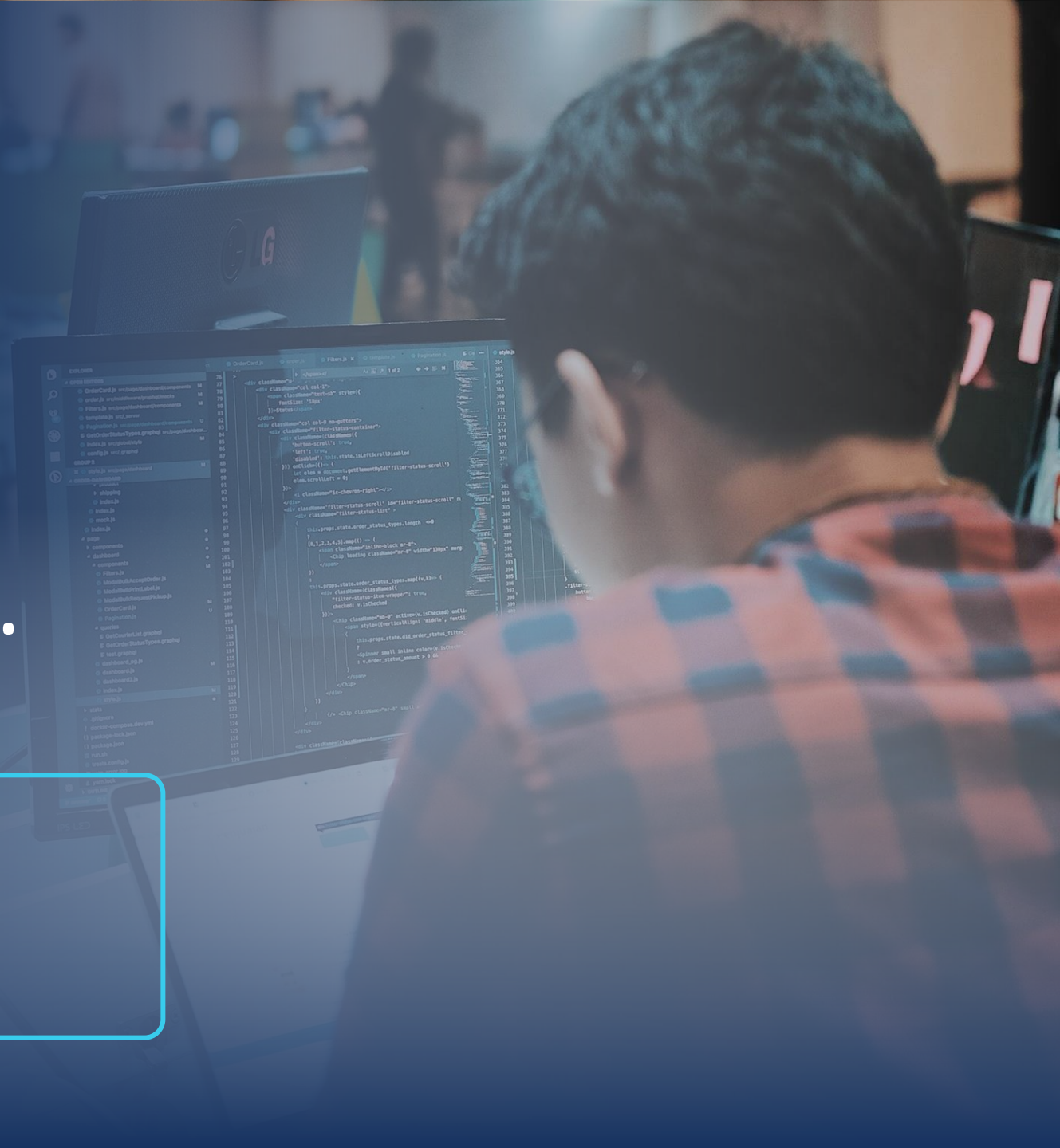
Confluent Operator Architecture and Deployment



http://ksql-chicago.gcp.selabs.net

Learn Kafka.

Start building with Apache Kafka at Confluent Developer.



Confluent Developer
developer.confluent.io

Stay in touch!

Confluent Blog

cnfl.io/blog

Streaming Audio

cnfl.io/podcast

Try Confluent

cnfl.io/download