Kafka as a Platform: the Ecosystem from the Ground Up



https://events.confluent.io/meetups

Robin Moffatt

@rmoff

$ whoami

- Robin Moffatt (@rmoff)
- Senior Developer Advocate at Confluent

  (Apache Kafka, not Wikis 😉)

- Working in data & analytics since 2001
-  Oracle ACE Director (Alumnus)



http://rmoff.dev/**talks** • http://rmoff.dev/**blog** • http://rmoff.dev/**youtube**



EVENTS

@rmoff | #ConfluentVUG | @confluentinc

EVENTS

EVENTS

- Notification
- State change

Human generated events

A Sale



A Stock movement

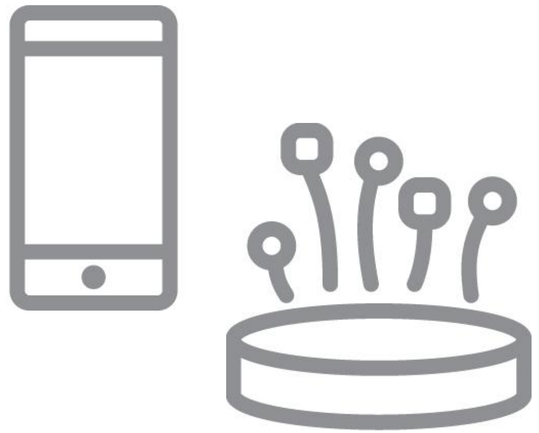


Machine generated events

Networking



IoT



Applications



EVENTS

are

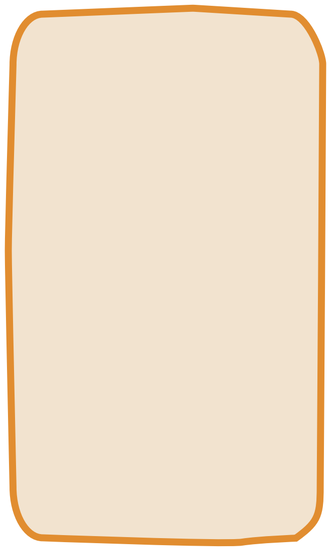
EVERYWHERE

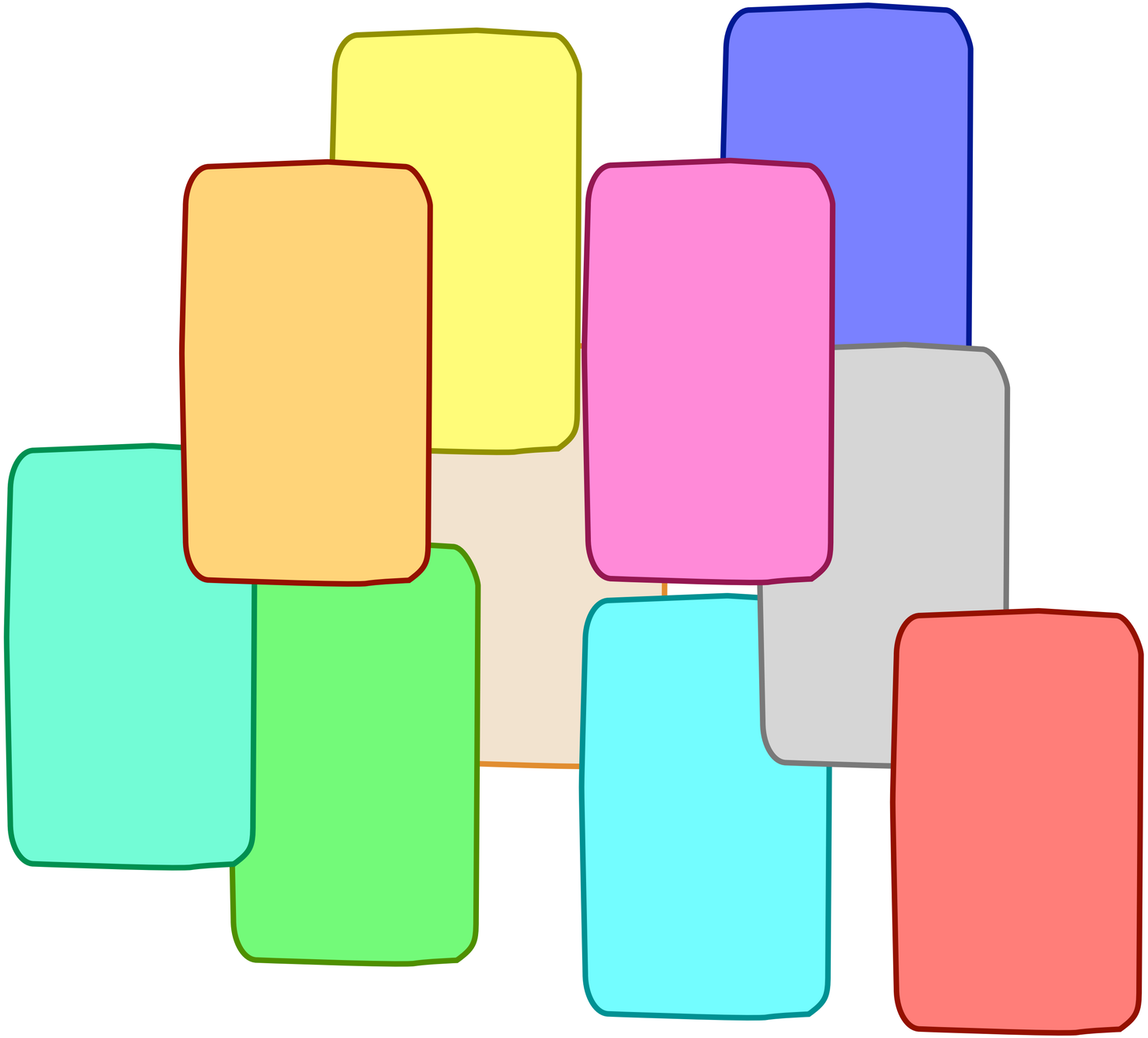
EVENTS

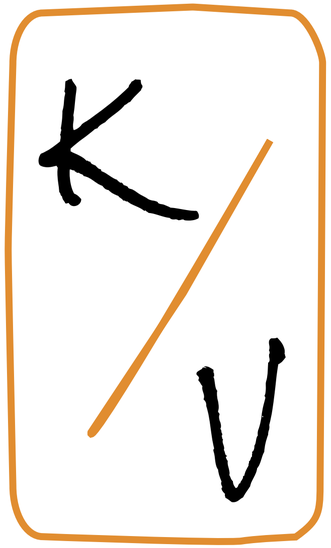
are

very

POWERFUL






K
V



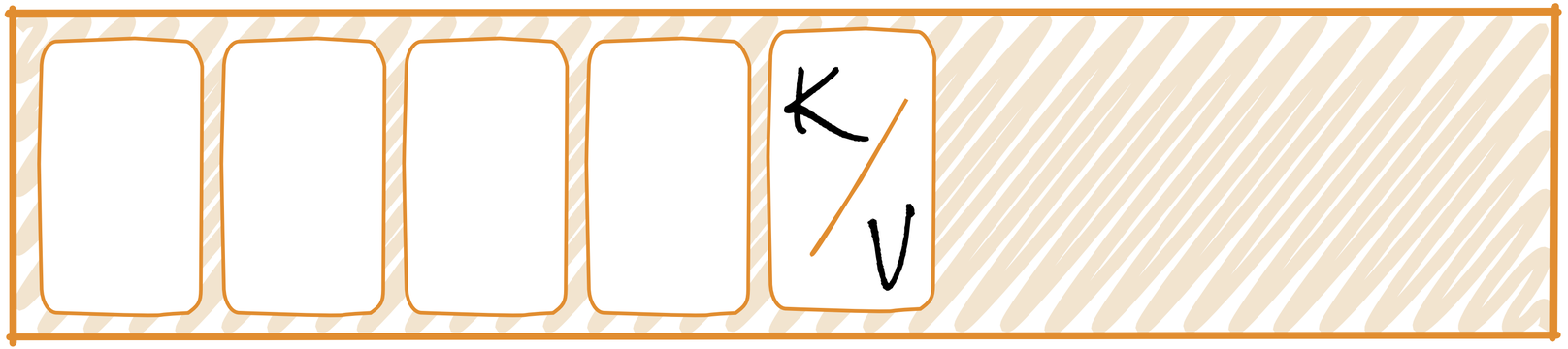
LOG

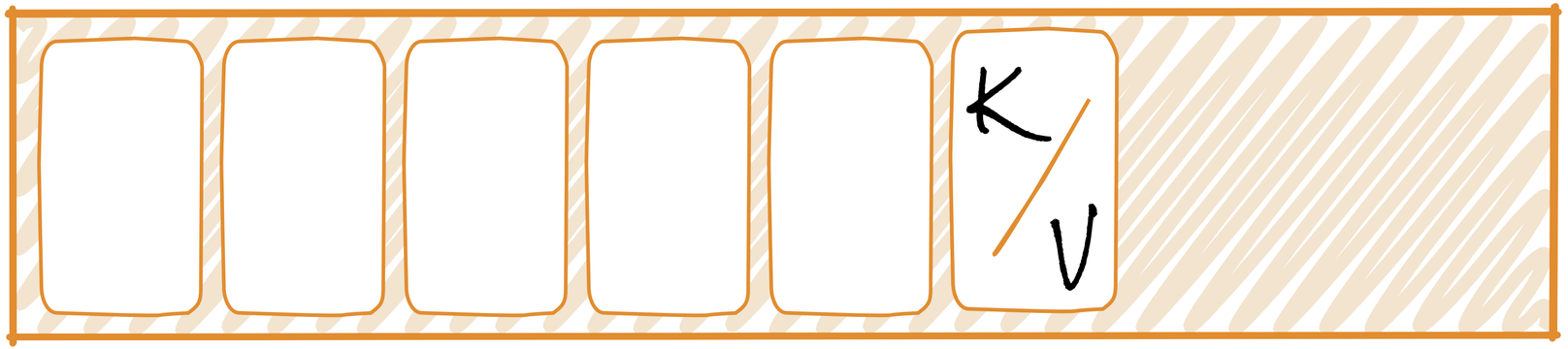

K
V

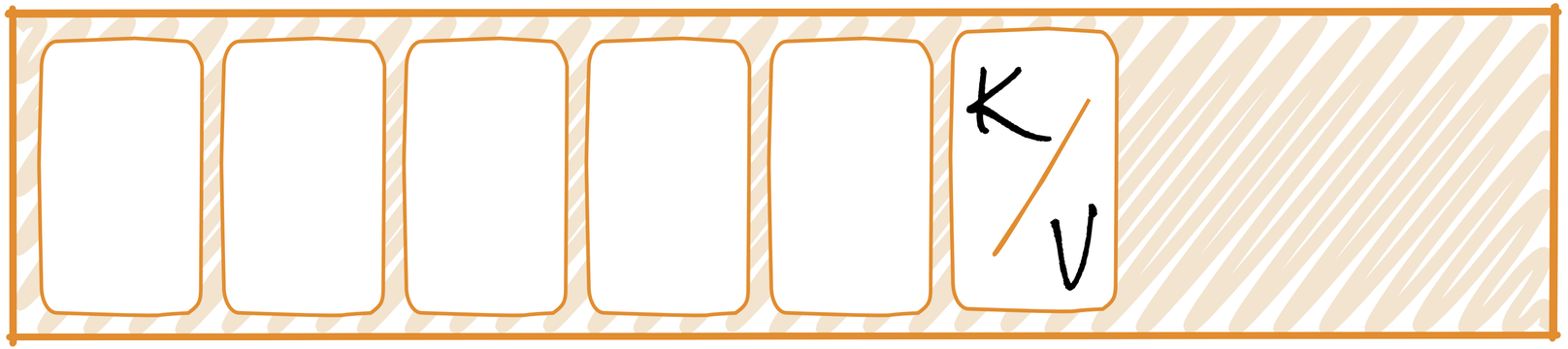

K
V


K
V

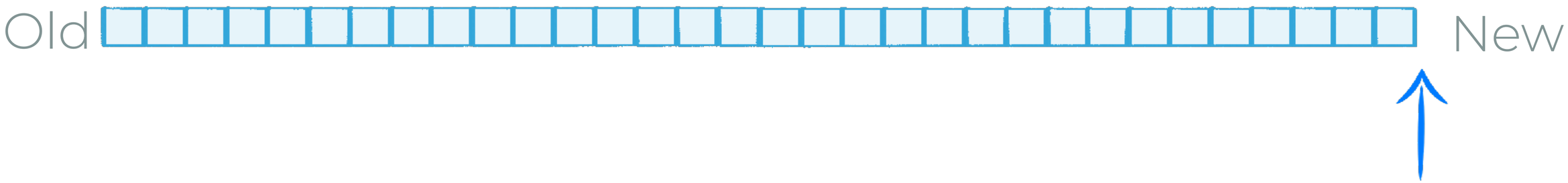

K
V


K
V


K
V


K
V

Immutable Event Log



Events are added at the end of the log



TOPICS

Topics

Clicks



Orders



Customers

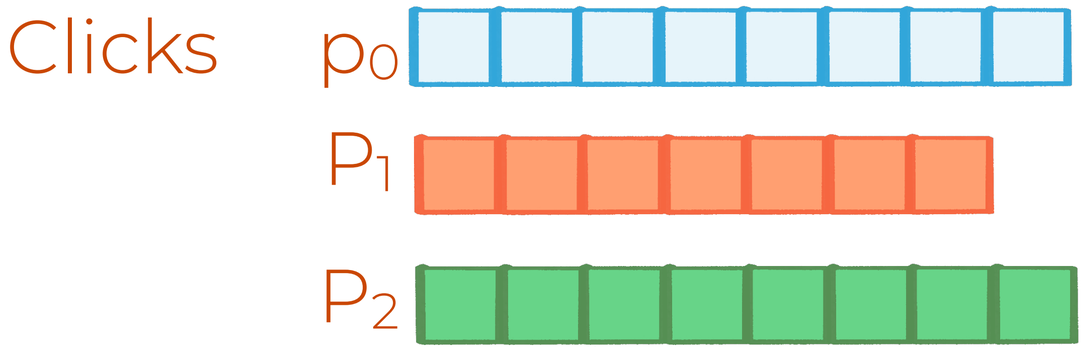


Topics are similar in concept to tables in a database



PARTITIONS

Partitions



Messages are guaranteed to be strictly ordered within a partition

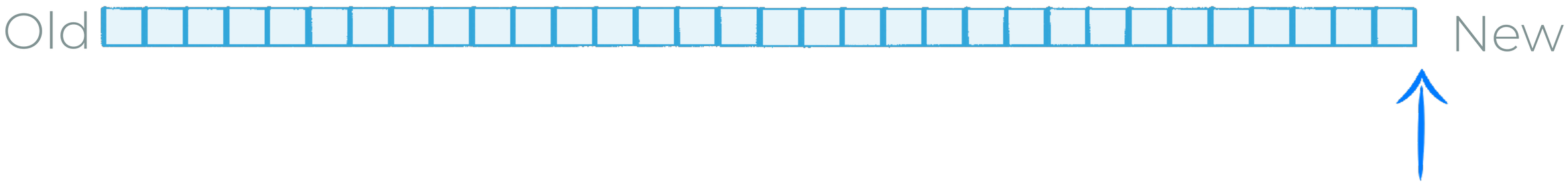


PUB / SUB

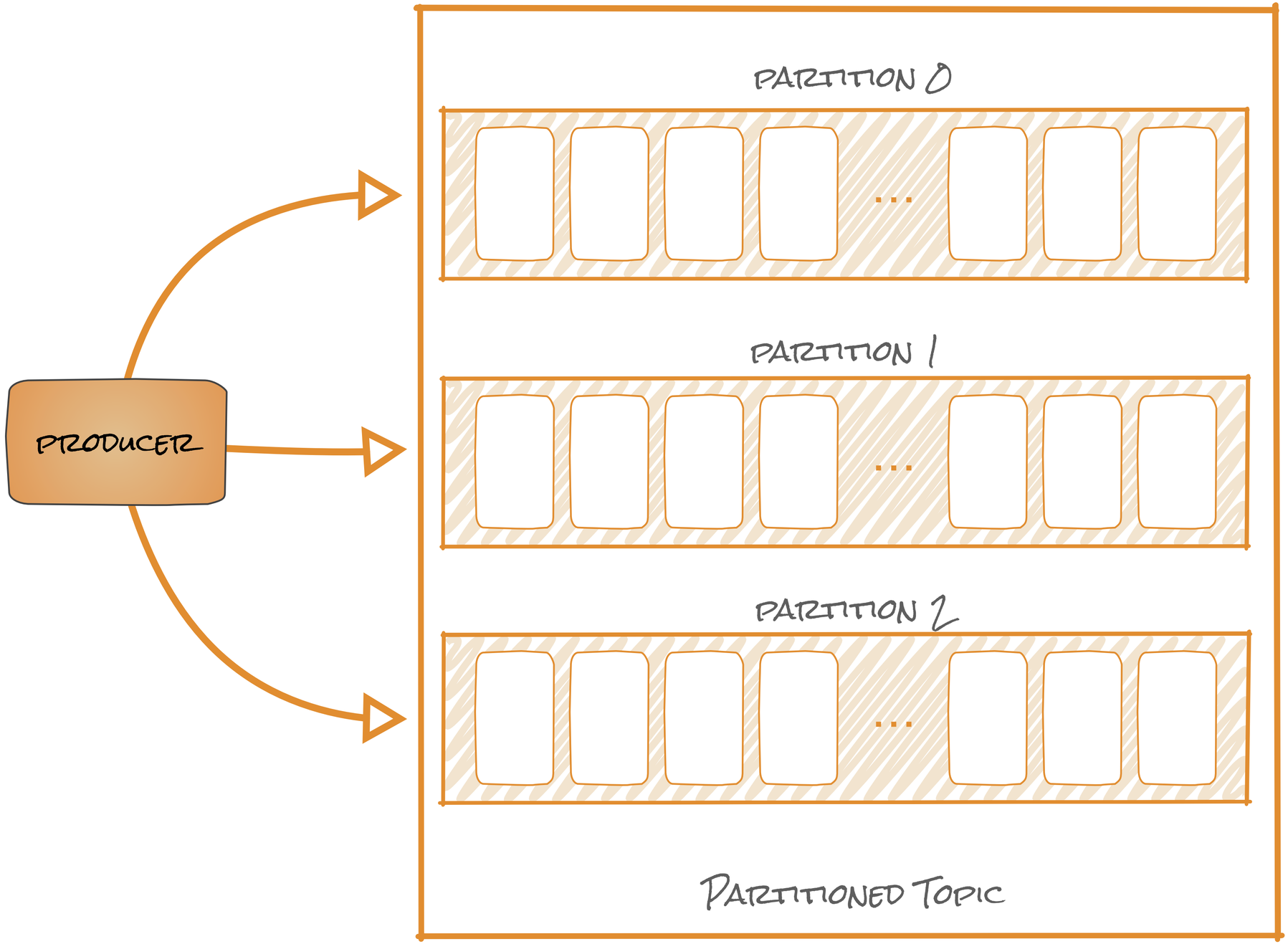


PUB / SUB

Producing data



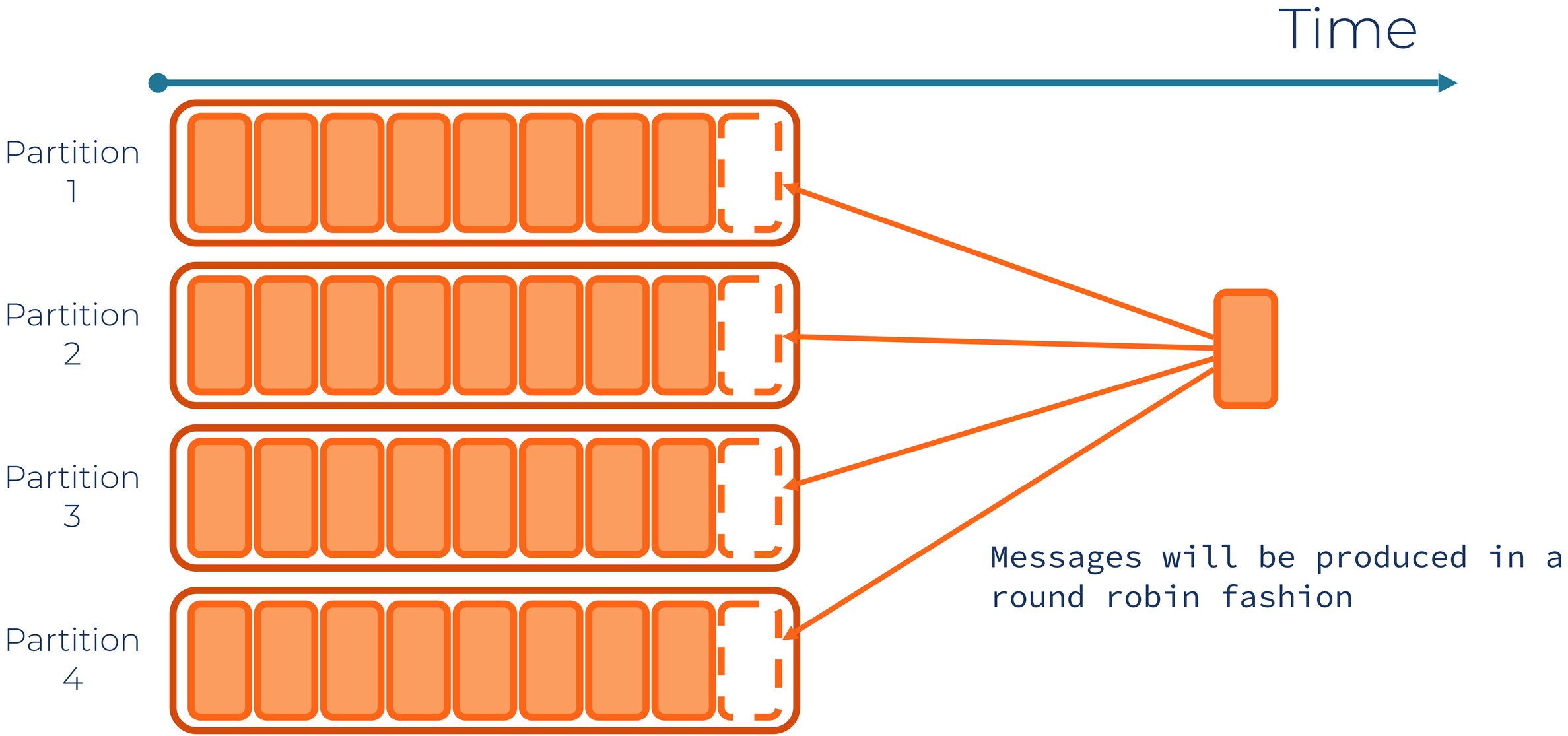
Messages are added at the end of the log


PRODUCER
PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC

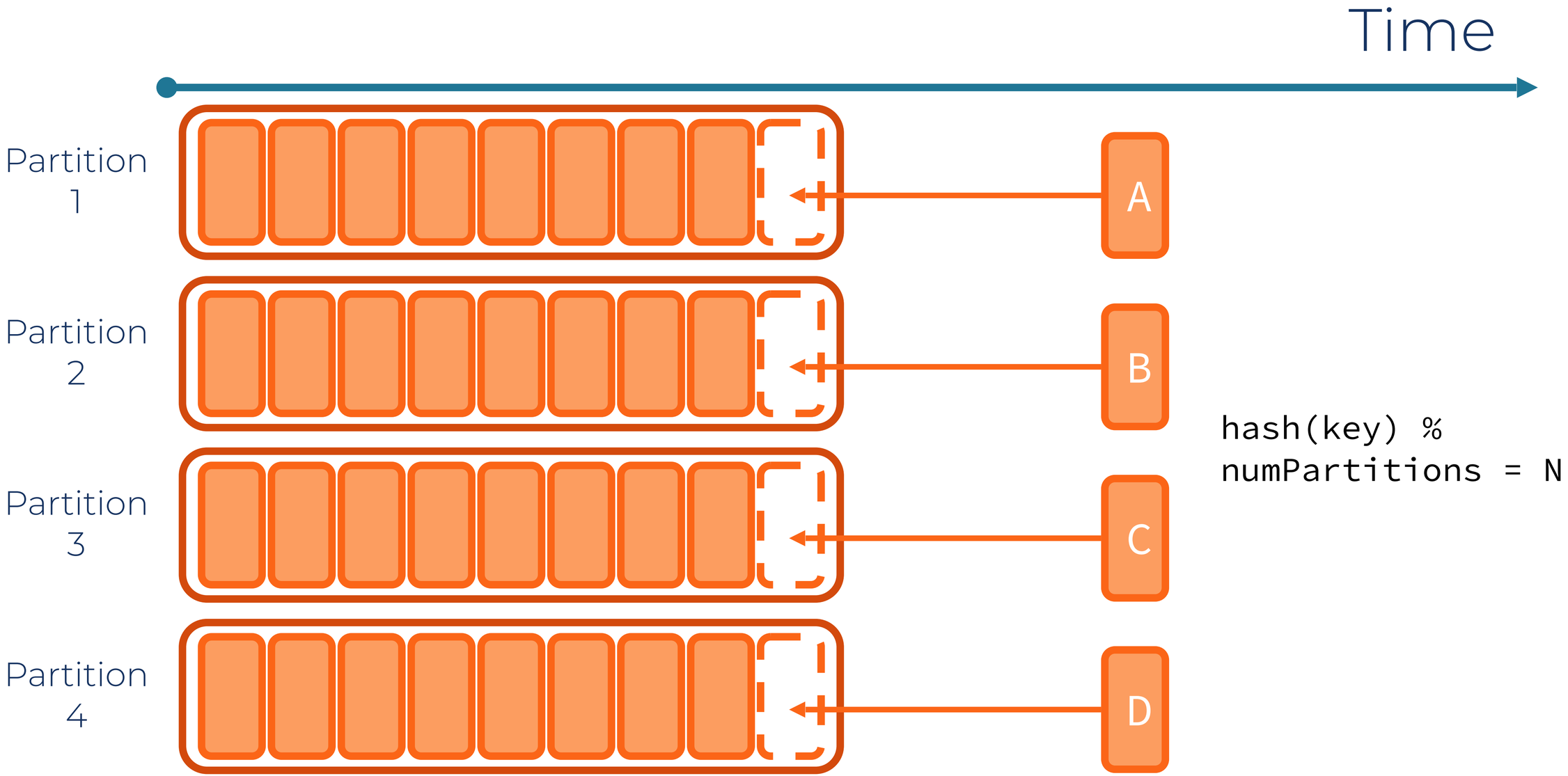
```
try (KafkaProducer<String, Payment> producer = new KafkaProducer<String, Payment>(props)) {

    for (long i = 0; i < 10; i++) {
        final String orderId = "id" + Long.toString(i);
        final Payment payment = new Payment(orderId, 1000.00d);
        final ProducerRecord<String, Payment> record =
           new ProducerRecord<String, Payment>("transactions",
                                               payment.getId().toString(),
                                               payment);
        producer.send(record);
    }
} catch (final InterruptedException e) {
    e.printStackTrace();
}
```

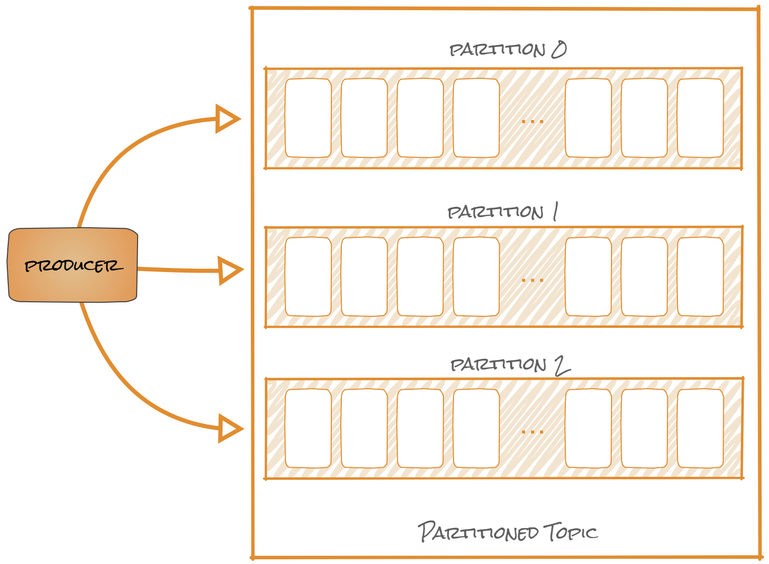
Producing to Kafka - No Key



Producing to Kafka - With Key



Producers



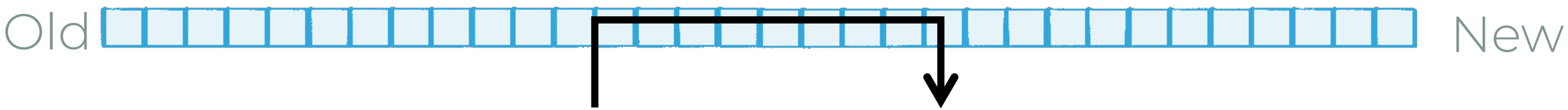
- A client application
- Puts messages into topics
- Handles partitioning, network protocol
- Java, Go, .NET, C/C++, Python
- Also every other language



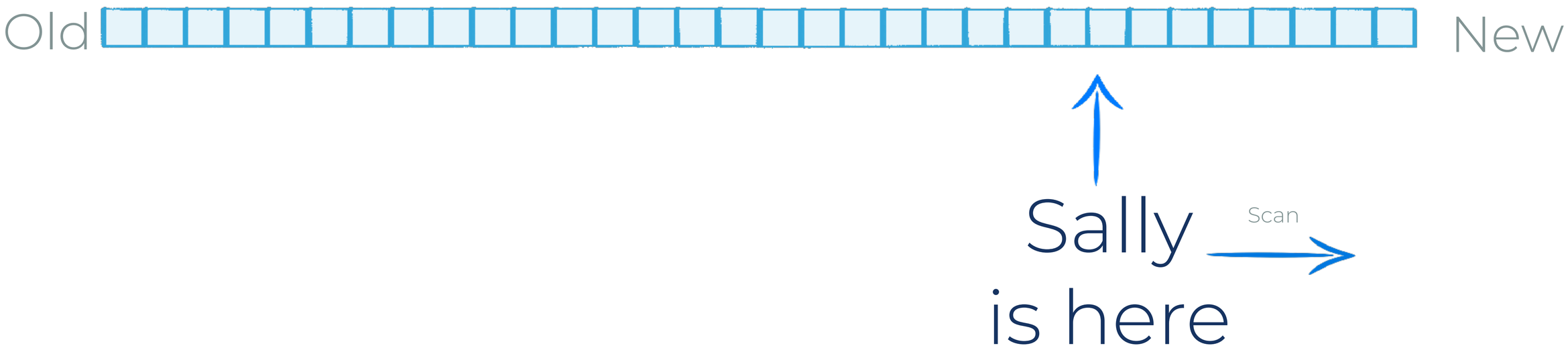
PUB / SUB

Consuming data - access is only sequential

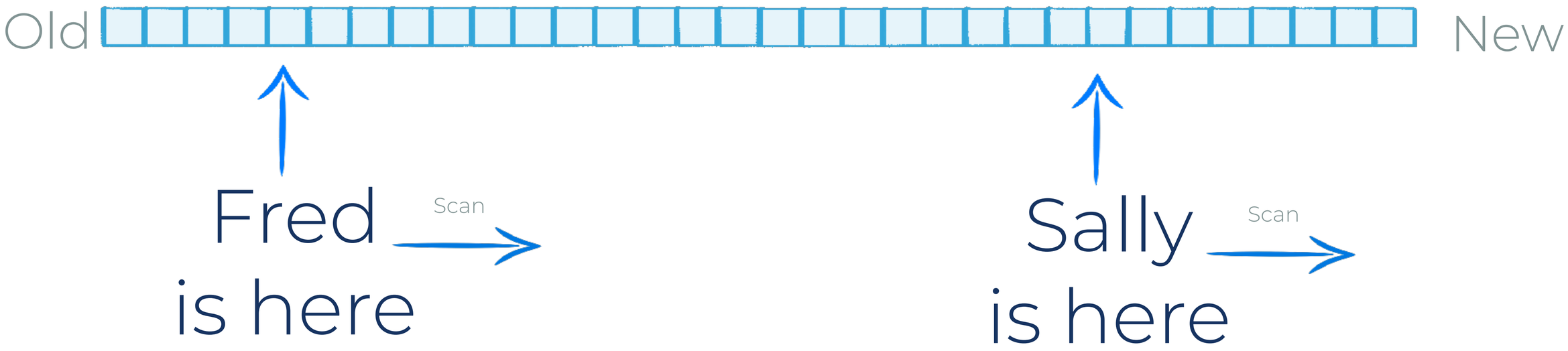
Read to offset & scan



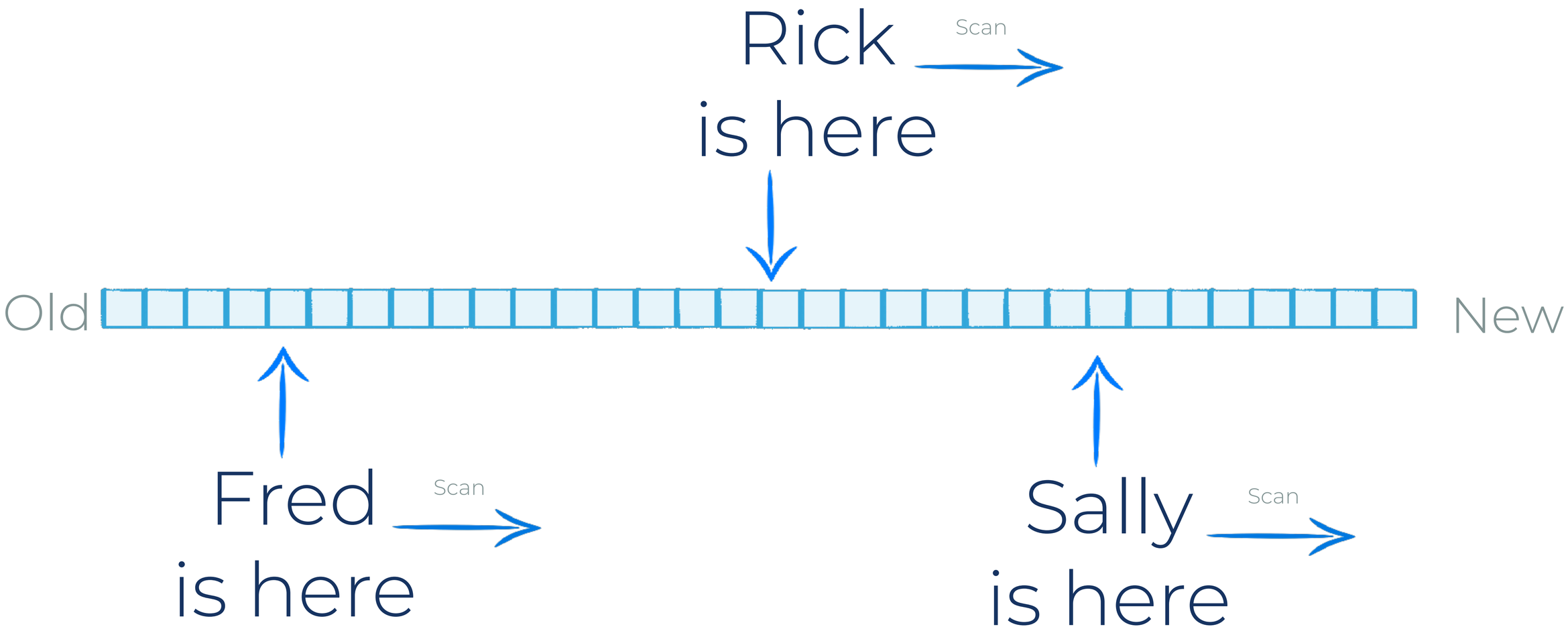
Consumers have a position of their own



Consumers have a position of their own

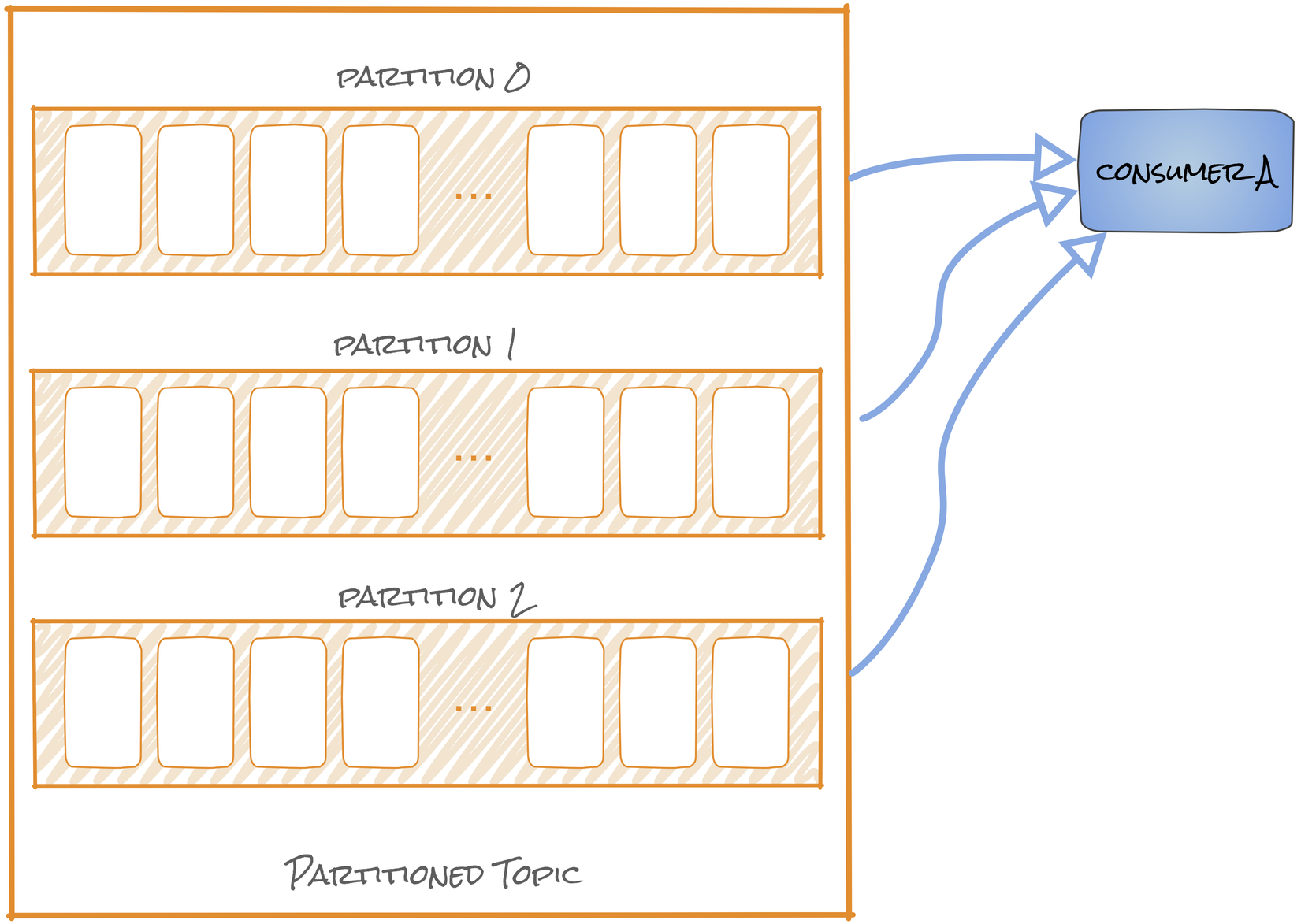


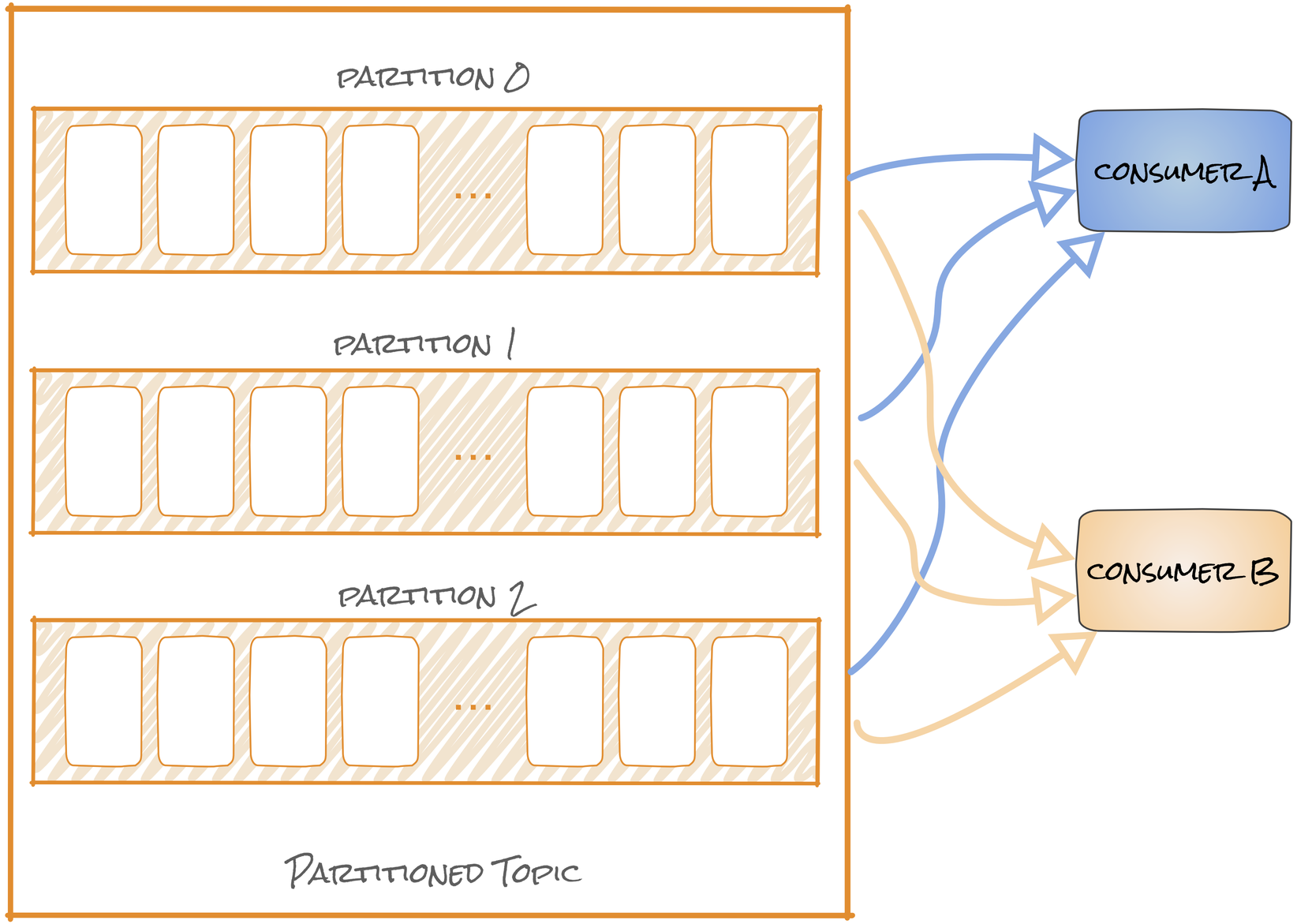
Consumers have a position of their own



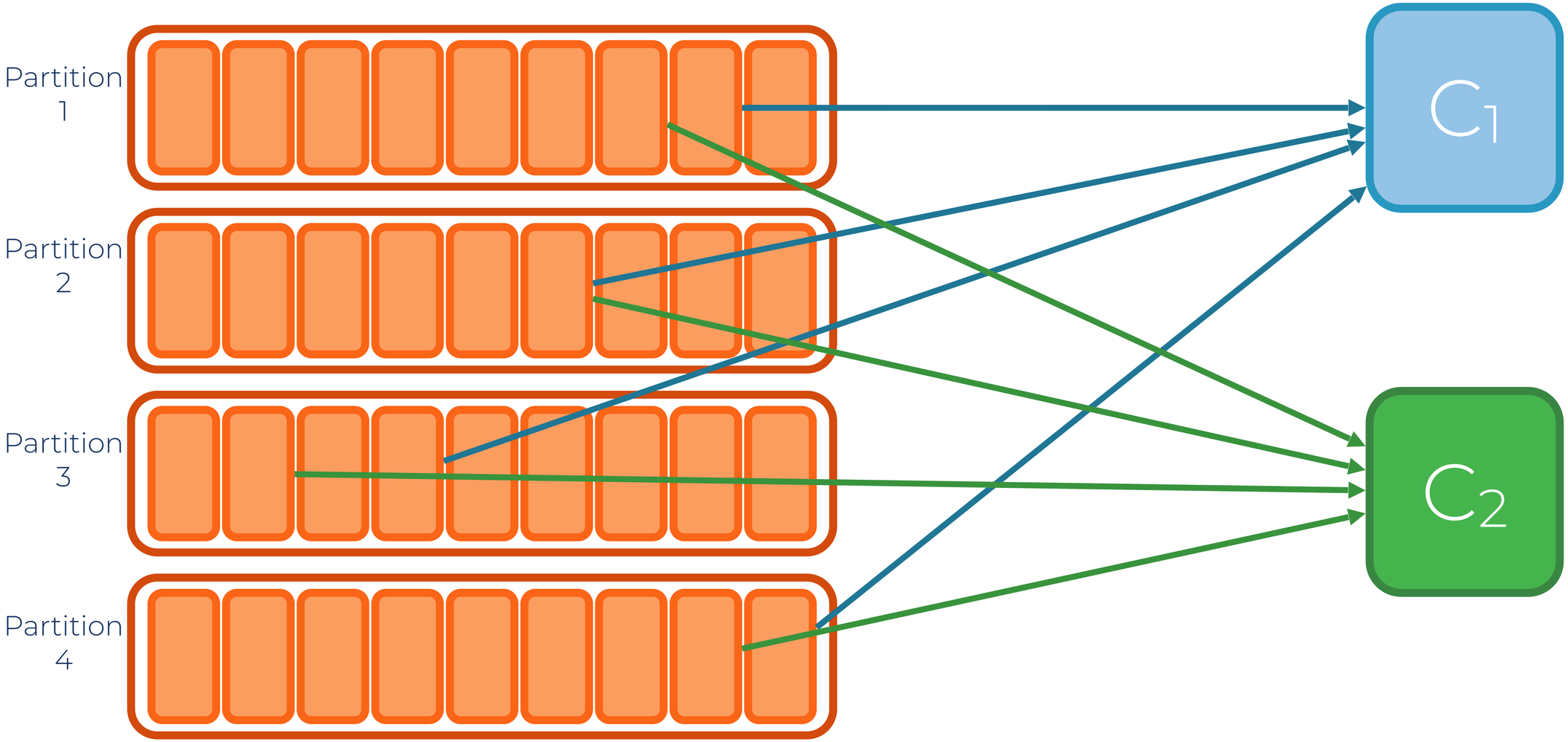
```
try (final KafkaConsumer<String, Payment> consumer = new KafkaConsumer<>(props)) {
    consumer.subscribe(Collections.singletonList(TOPIC));

    while (true) {
        ConsumerRecords<String, Payment> records = consumer.poll(100);
        for (ConsumerRecord<String, Payment> record : records) {
            String key = record.key();
            Payment value = record.value();
            System.out.printf("key = %s, value = %s%n", key, value);
        }
    }
}
```

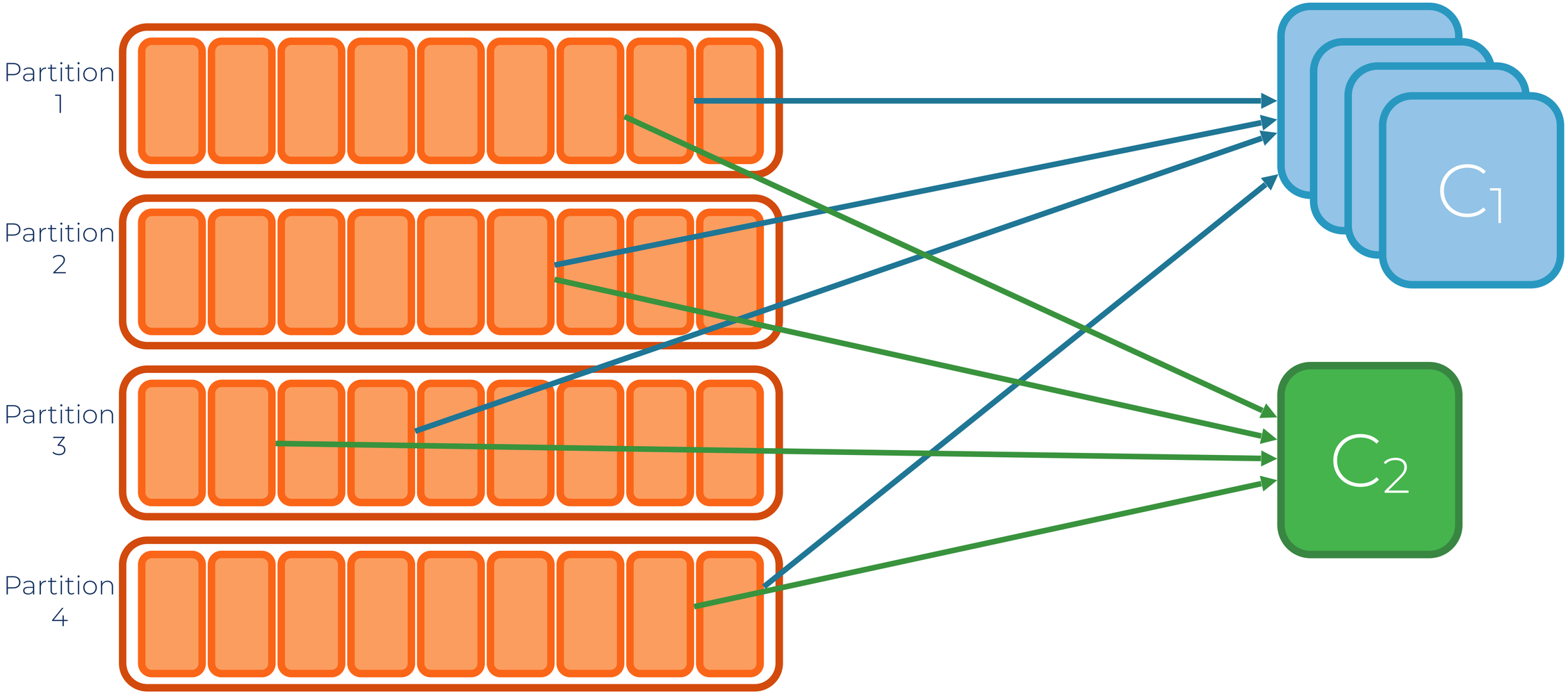

PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A


PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER B

Consuming From Kafka - Multiple Consumers

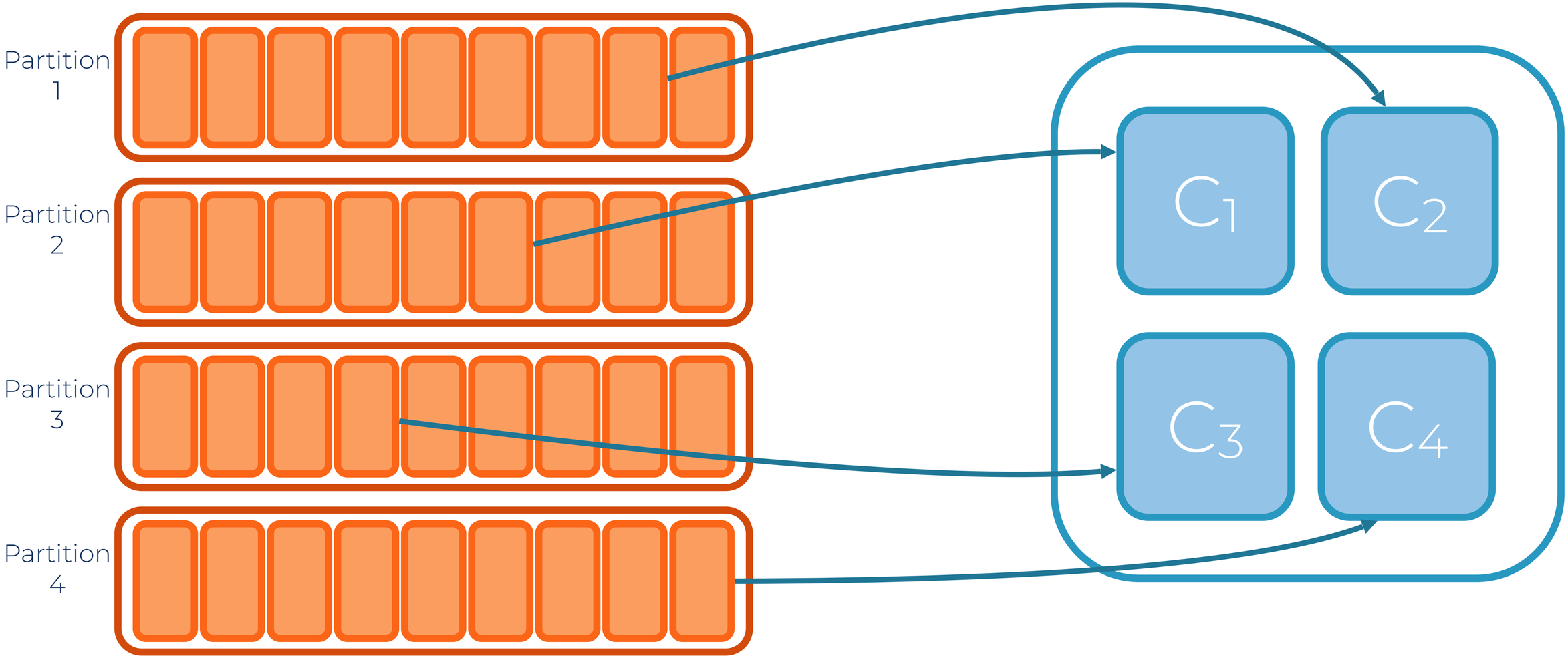


Consuming From Kafka - Grouped Consumers

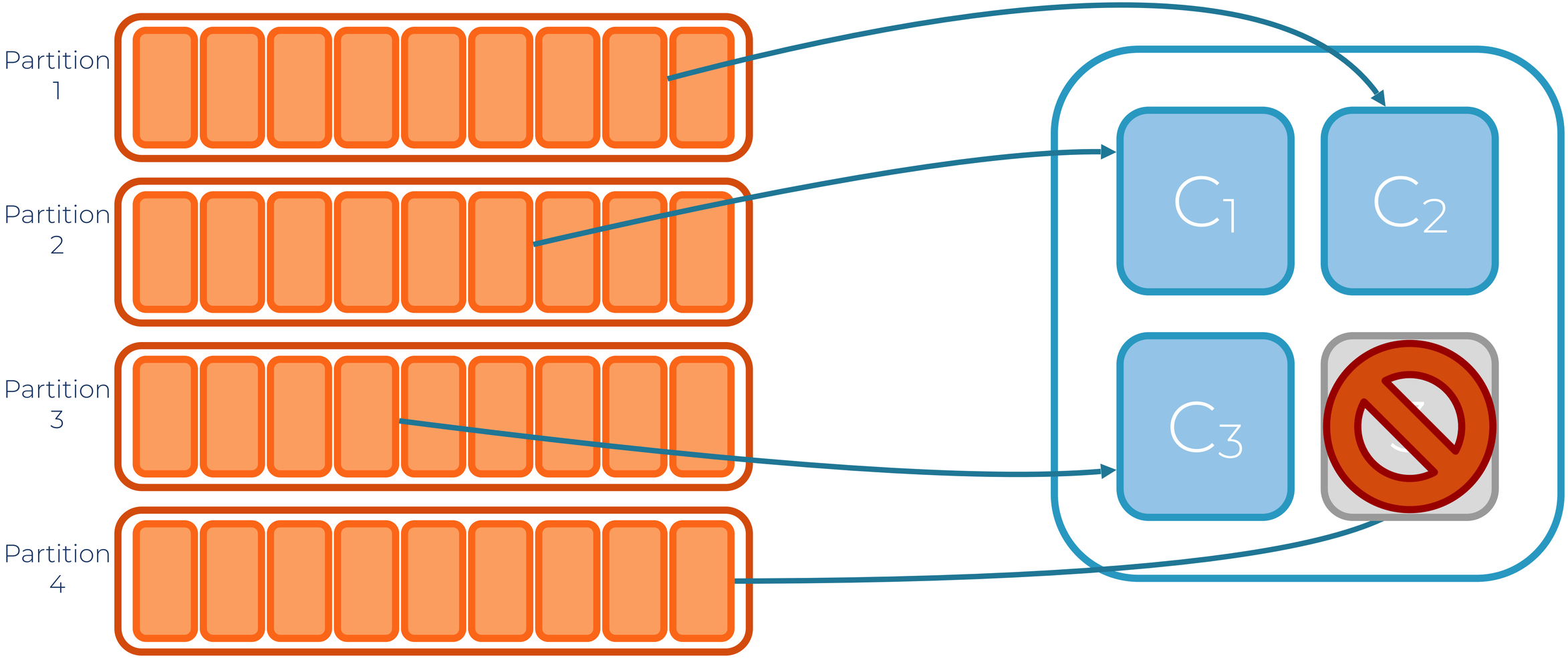



CONSUMER GROUP COORDINATOR
CONSUMERS
CONSUMER GROUP

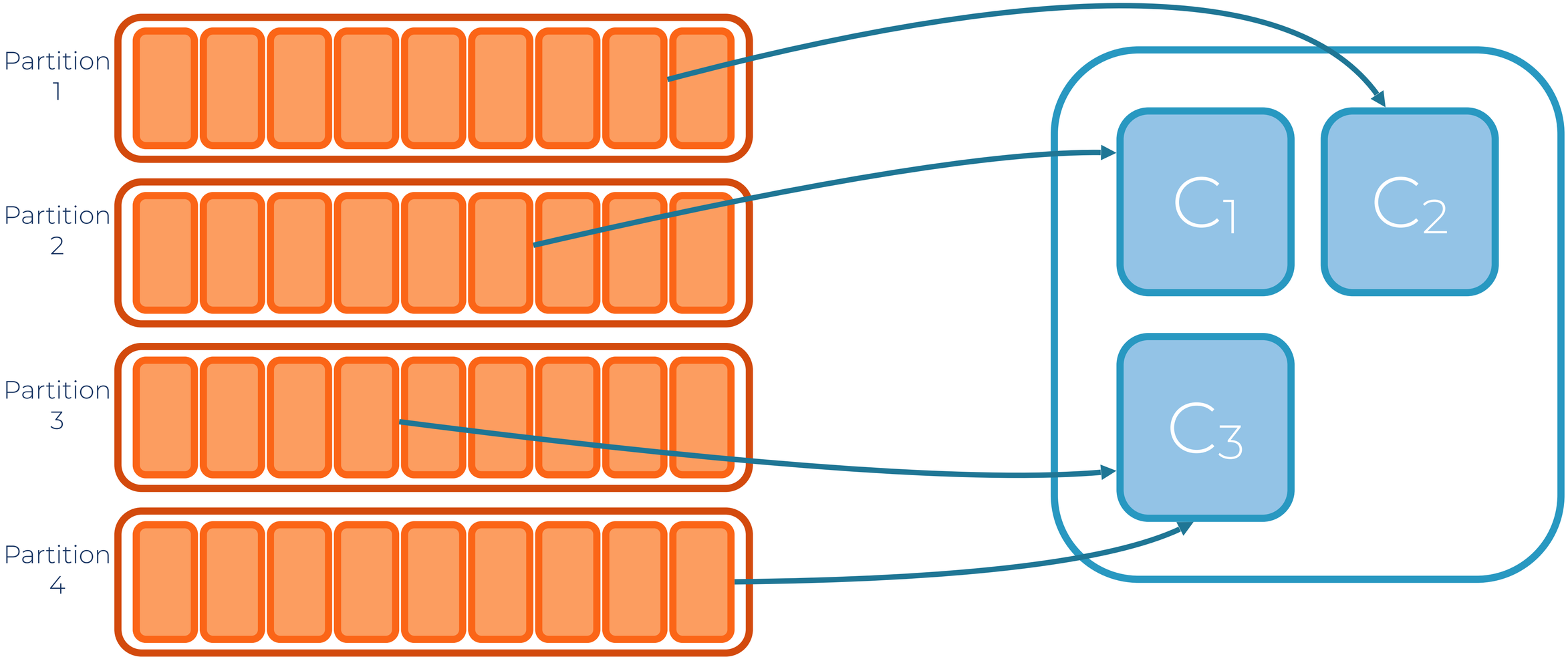
Consuming From Kafka - Grouped Consumers



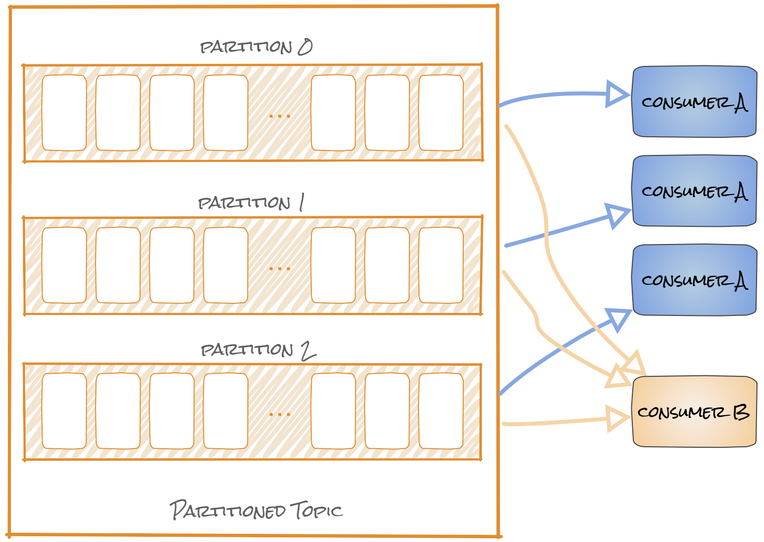
Consuming From Kafka - Grouped Consumers



Consuming From Kafka - Grouped Consumers



Consumers



- A client application
- Reads messages from topics
- Horizontally, elastically scalable (if stateless)
- Java, Go, .NET, C/C++, Python, everything else

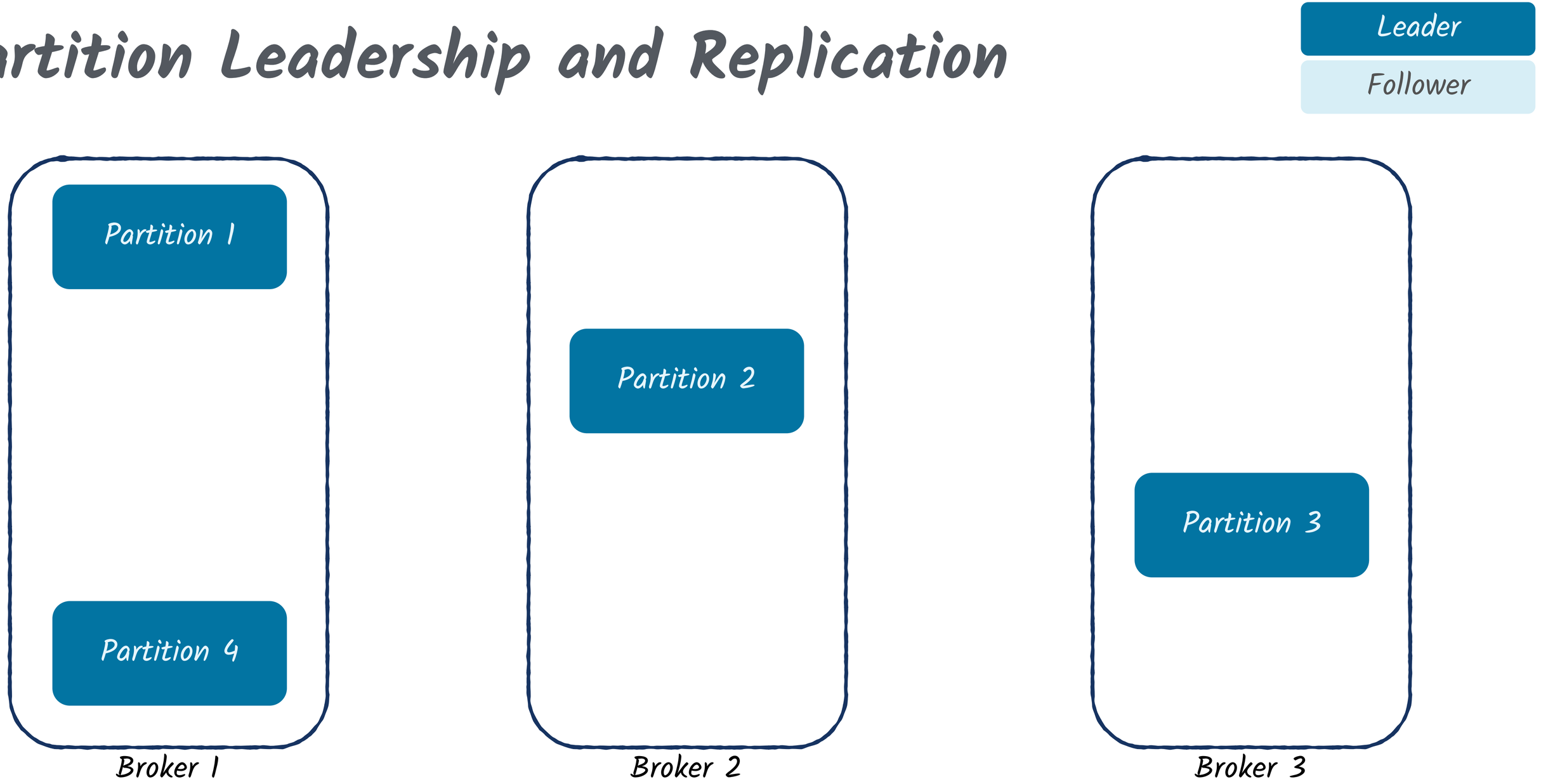


BROKERS

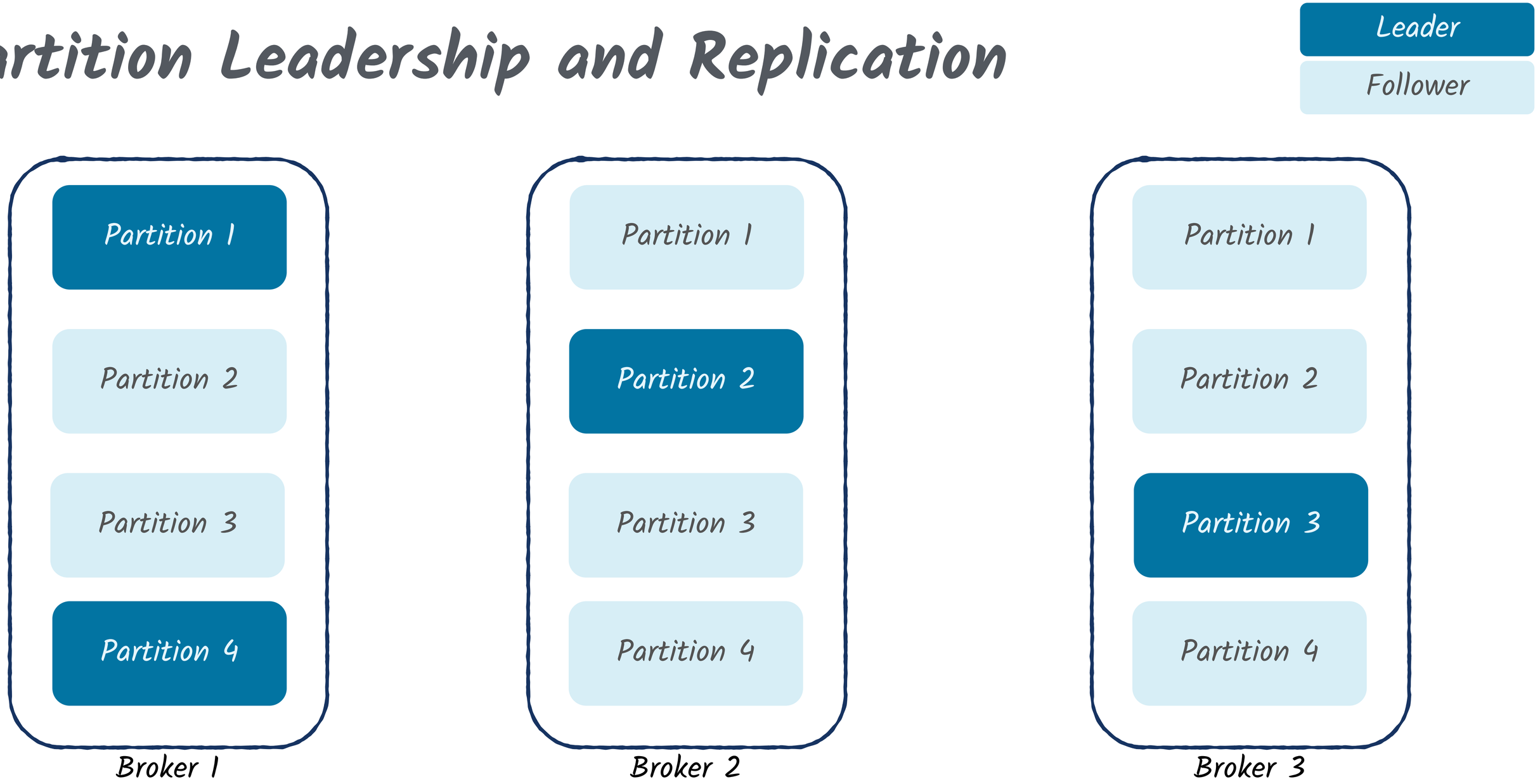
and REPLICATION

@rmoff | #ConfluentVUG | @confluentinc

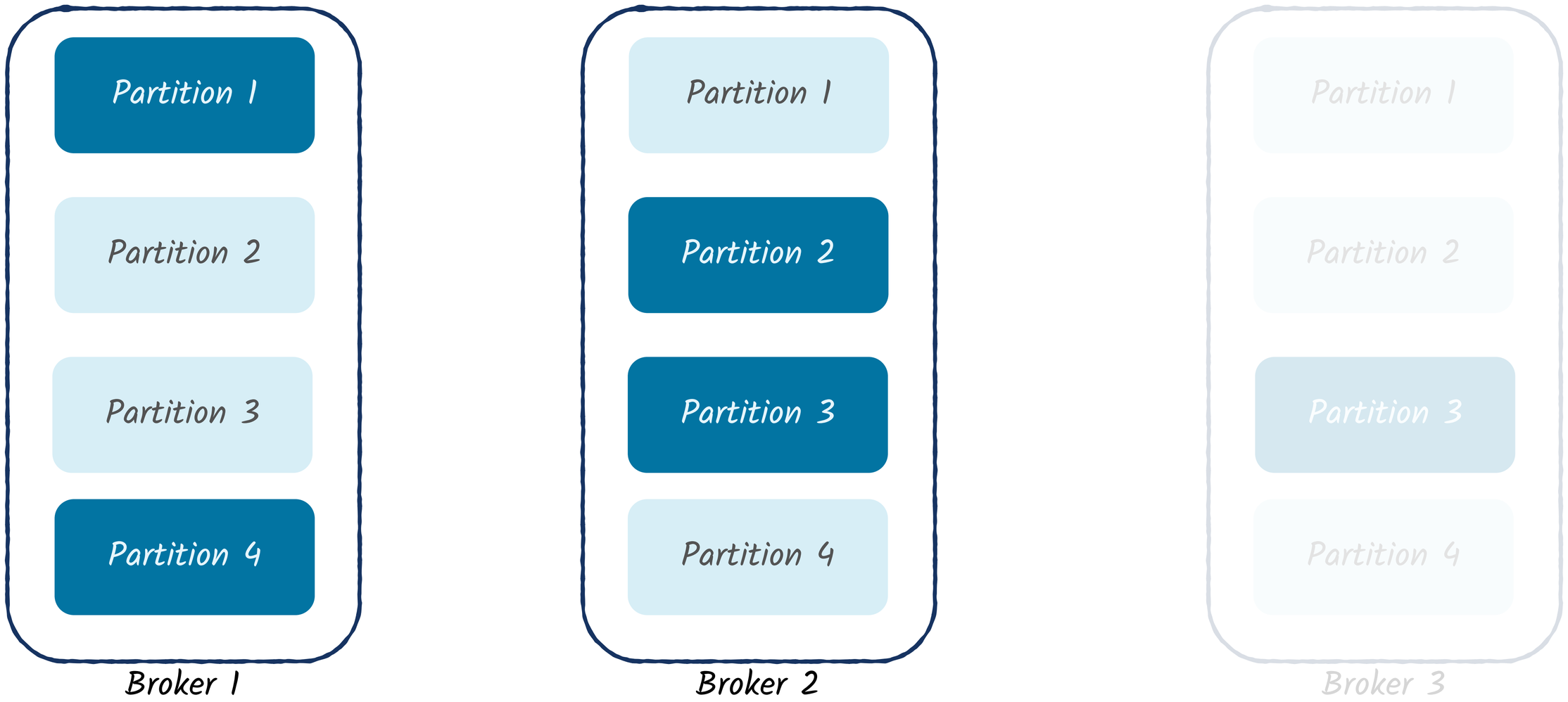
Partition Leadership and Replication



Partition Leadership and Replication



Partition Leadership and Replication





CONFLUENT

DEMO

@rmoff | #ConfluentVUG | @confluentinc



So far, this is

Pretty good

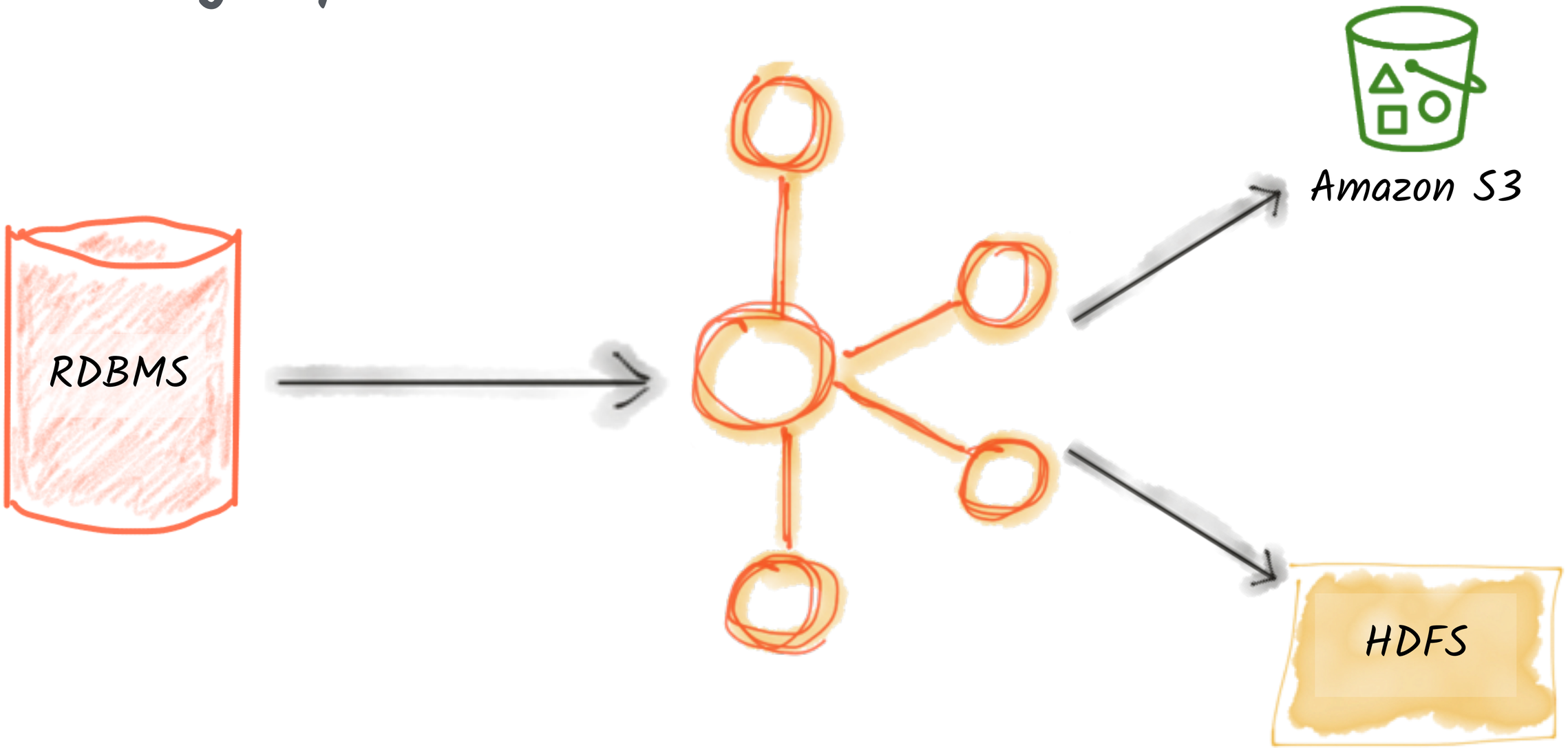


So far, this is

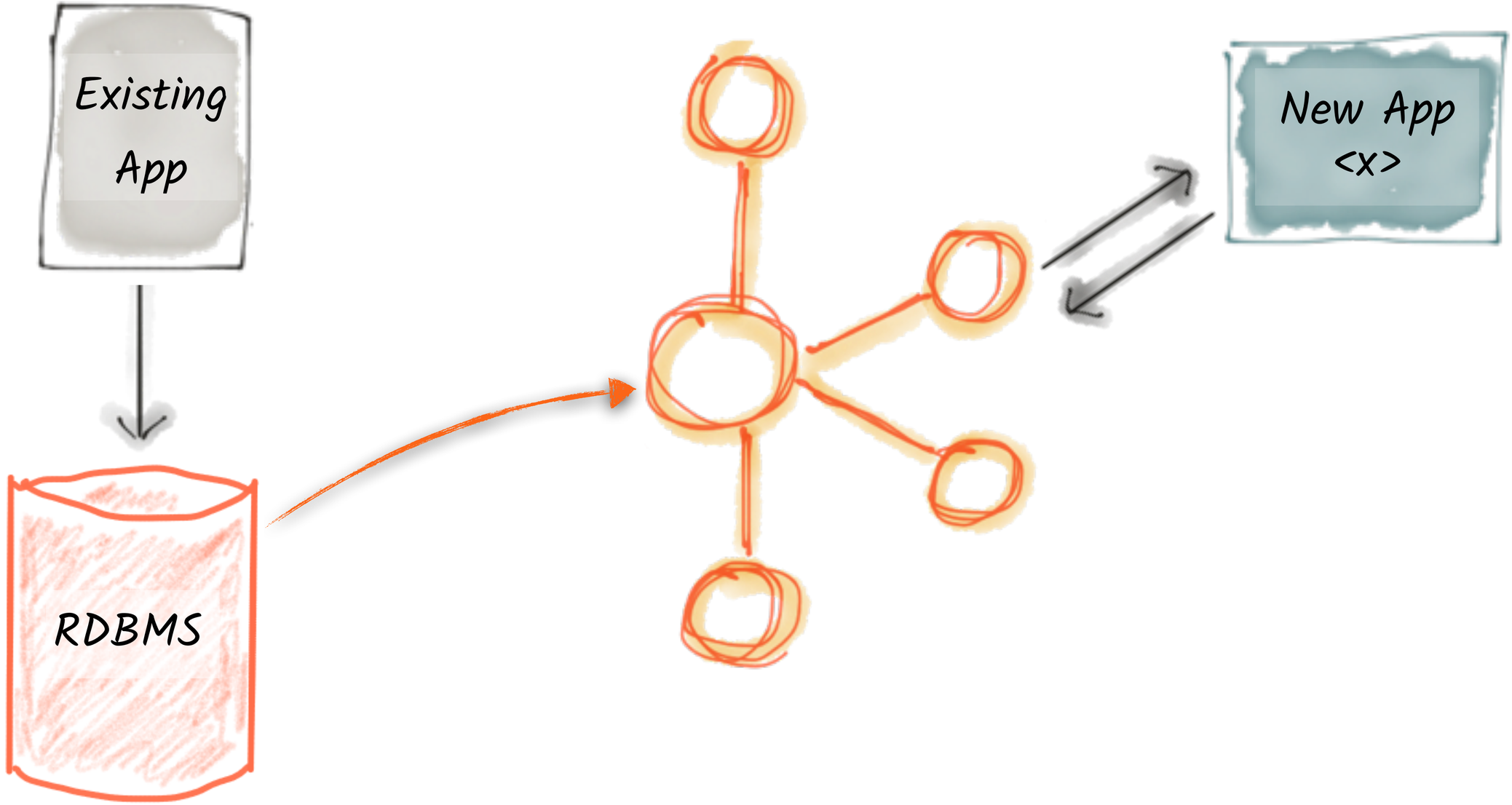
Pretty good

but I've not finished yet...

Streaming Pipelines



Evolve processing from old systems to new

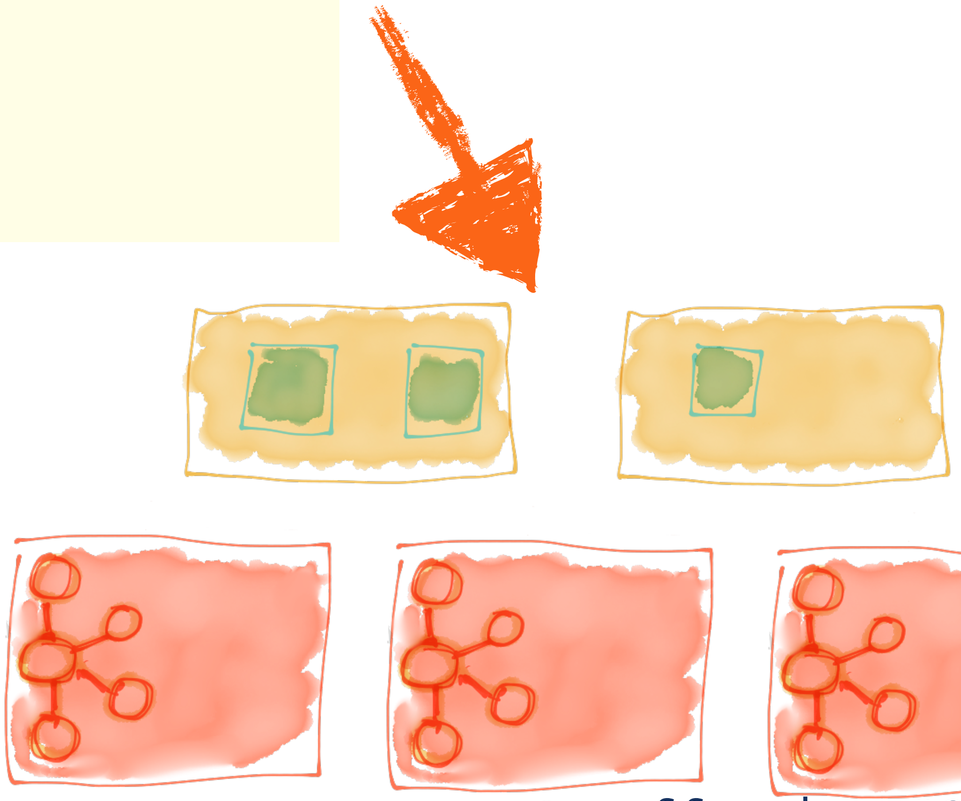




Streaming Integration with Kafka Connect



Sources



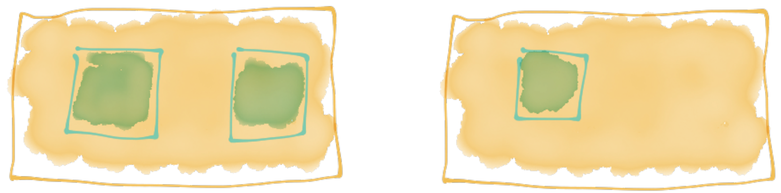
Kafka Connect

Kafka Brokers

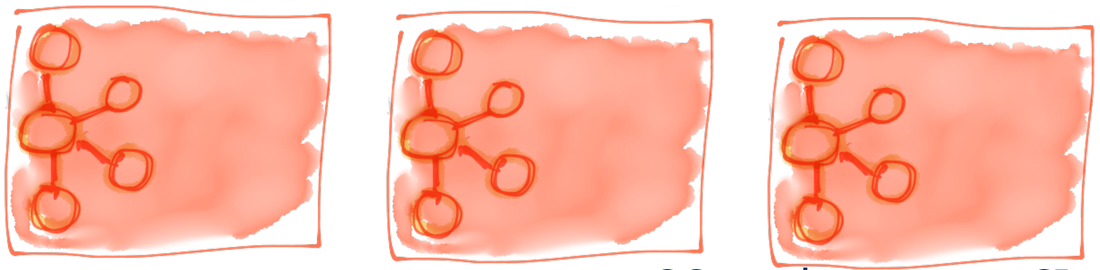
Streaming Integration with Kafka Connect

Sinks





Kafka Connect

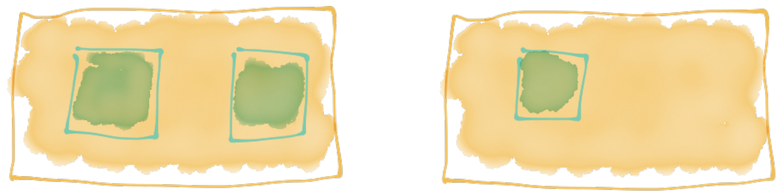


Kafka Brokers

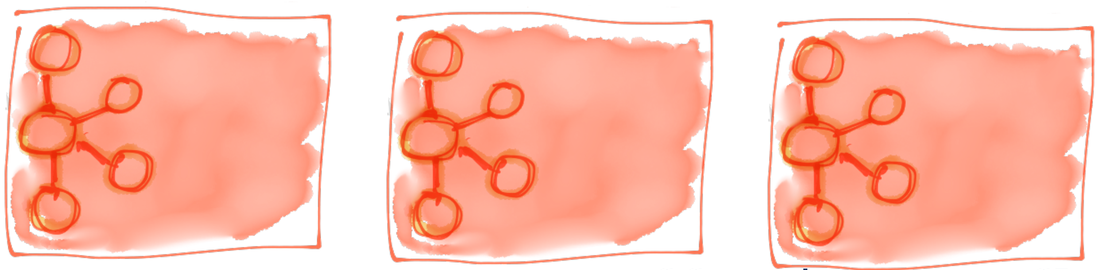
Streaming Integration with Kafka Connect







Kafka Connect

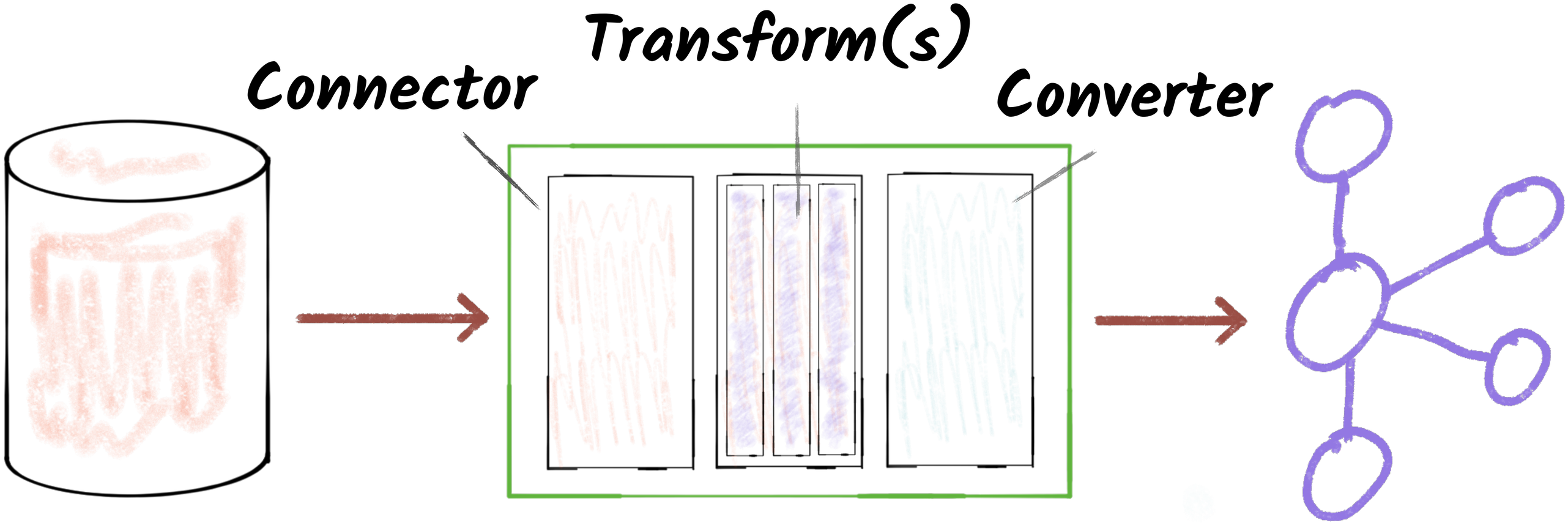


Kafka Brokers

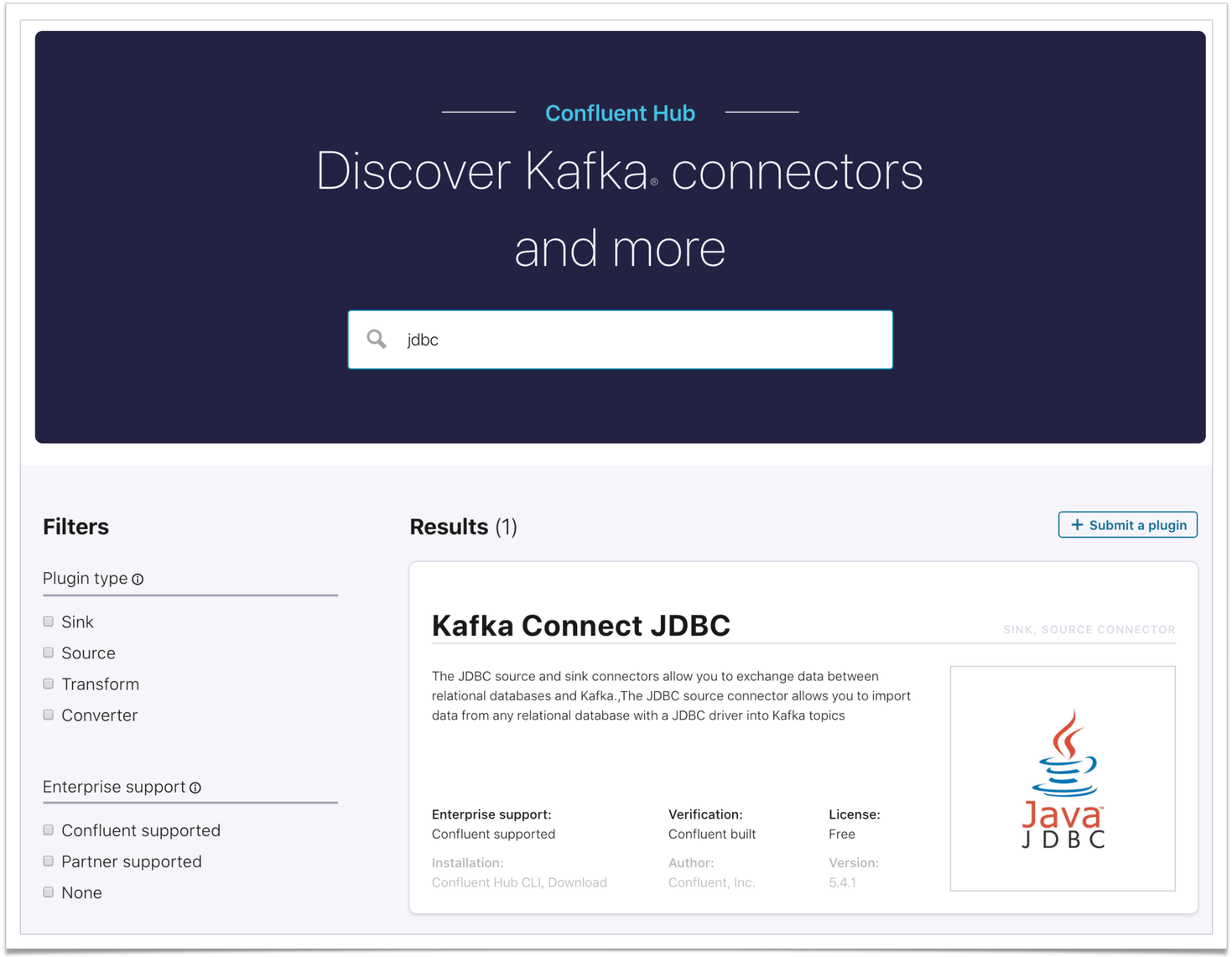
Look Ma, No Code!

```
{
  "connector.class":
    "io.confluent.connect.jdbc.JdbcSourceConnector",
  "connection.url":
    "jdbc:mysql://asgard:3306/demo",
  "table.whitelist":
    "sales,orders,customers"
}
```

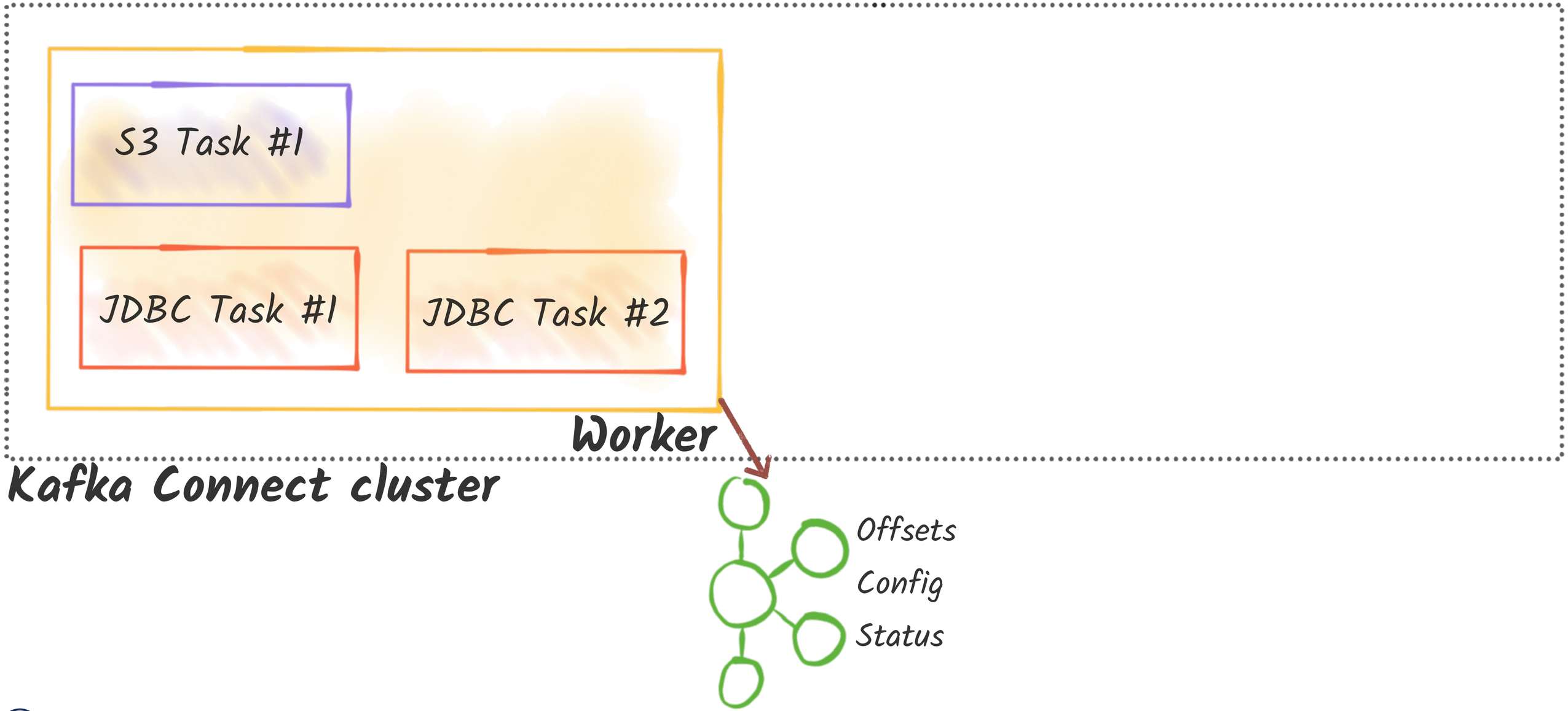
Extensible



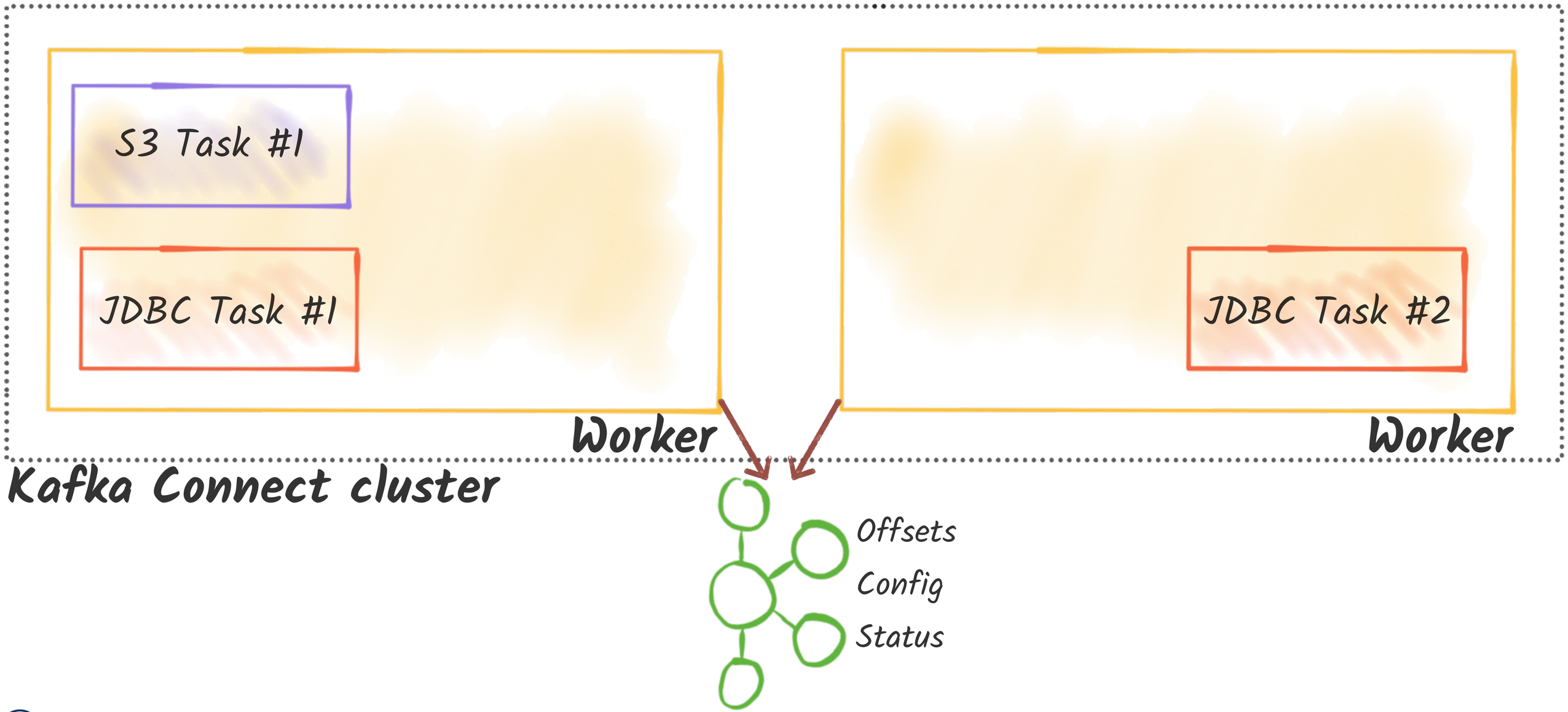
hub.confluent.io



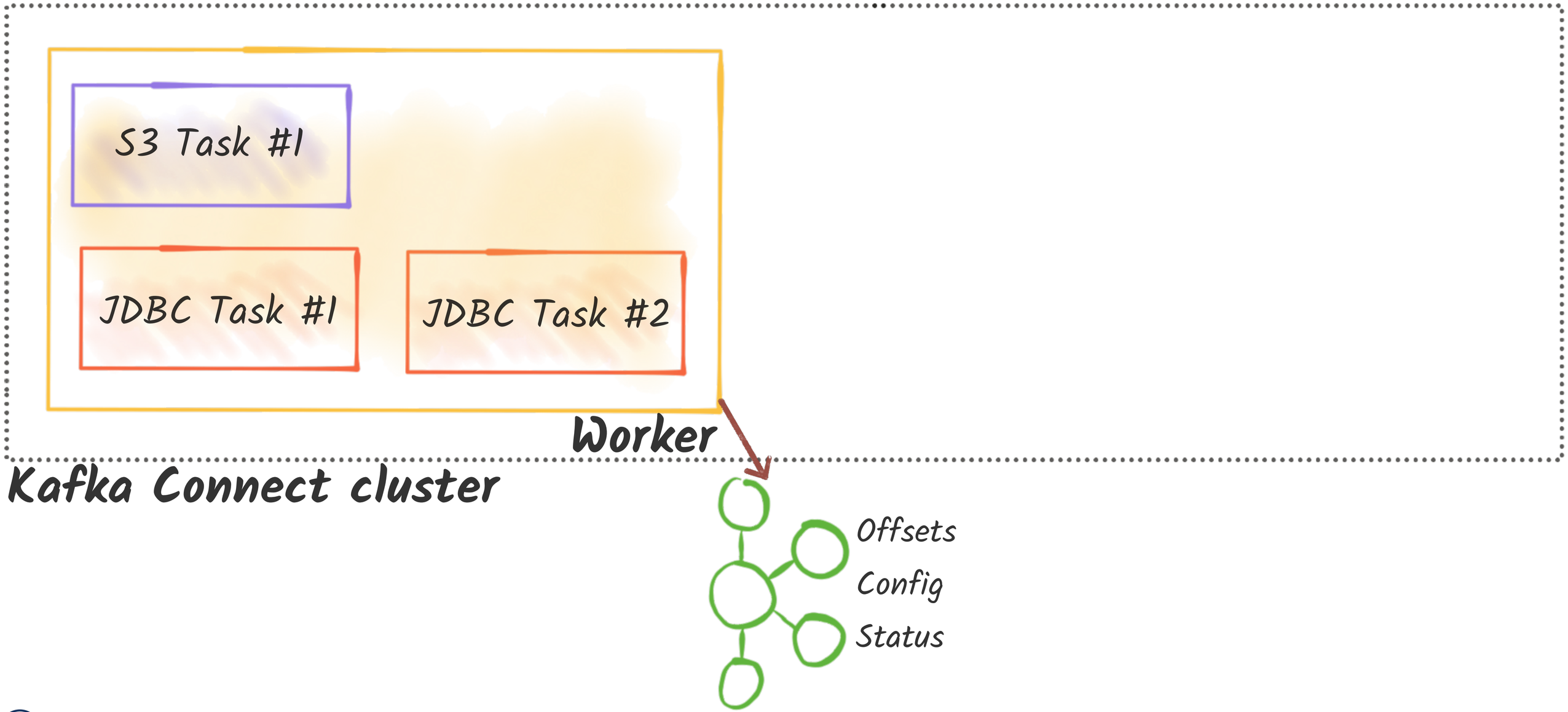
Single Kafka Connect node



Kafka Connect - scalable and fault-tolerant



Automatic fault tolerance

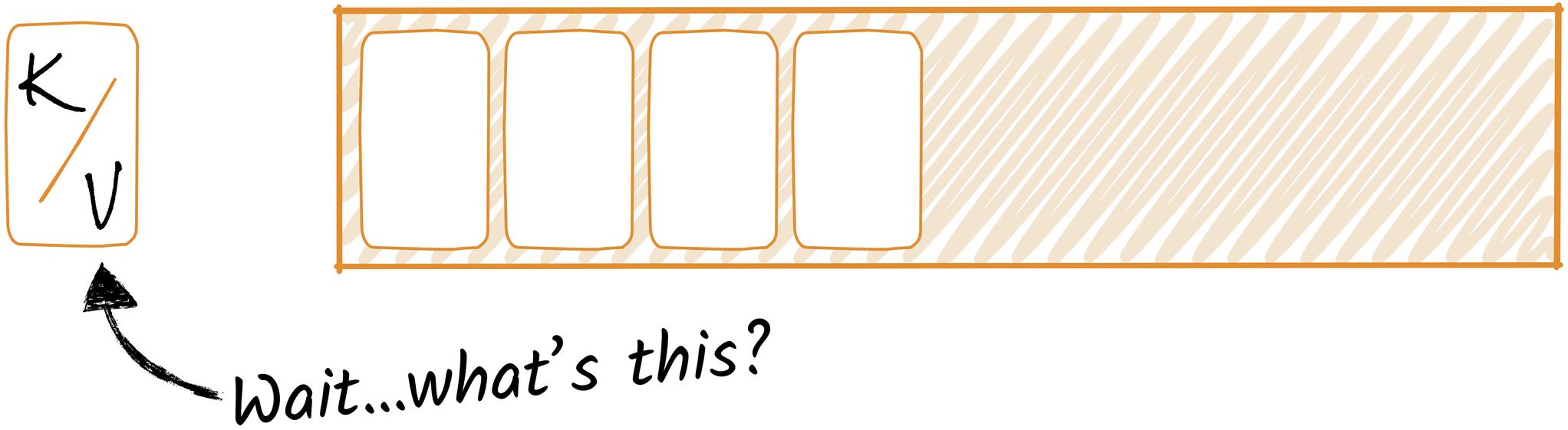



K
V


K
V


K
V

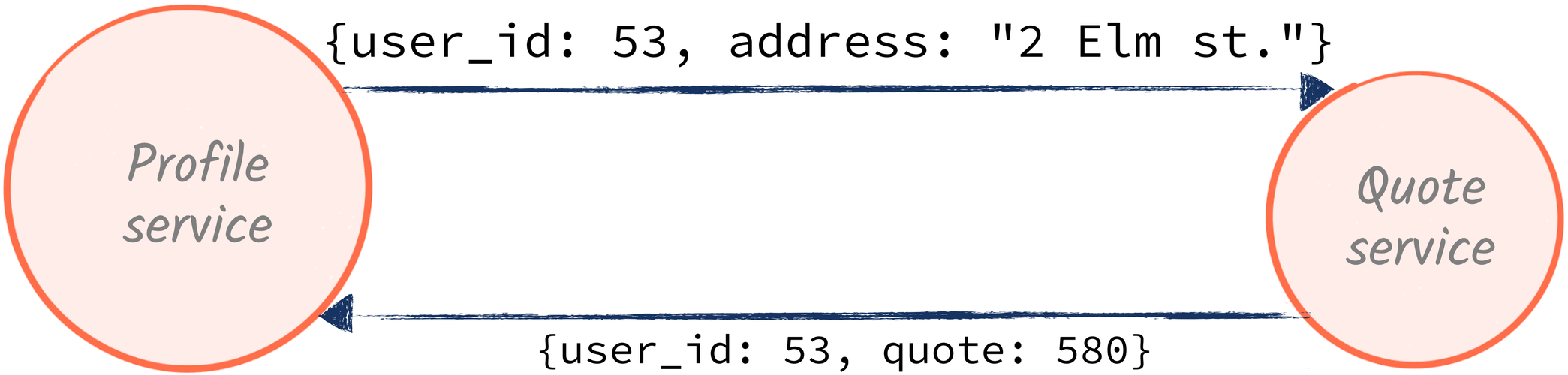

K
V


K
V
Wait...what's this?

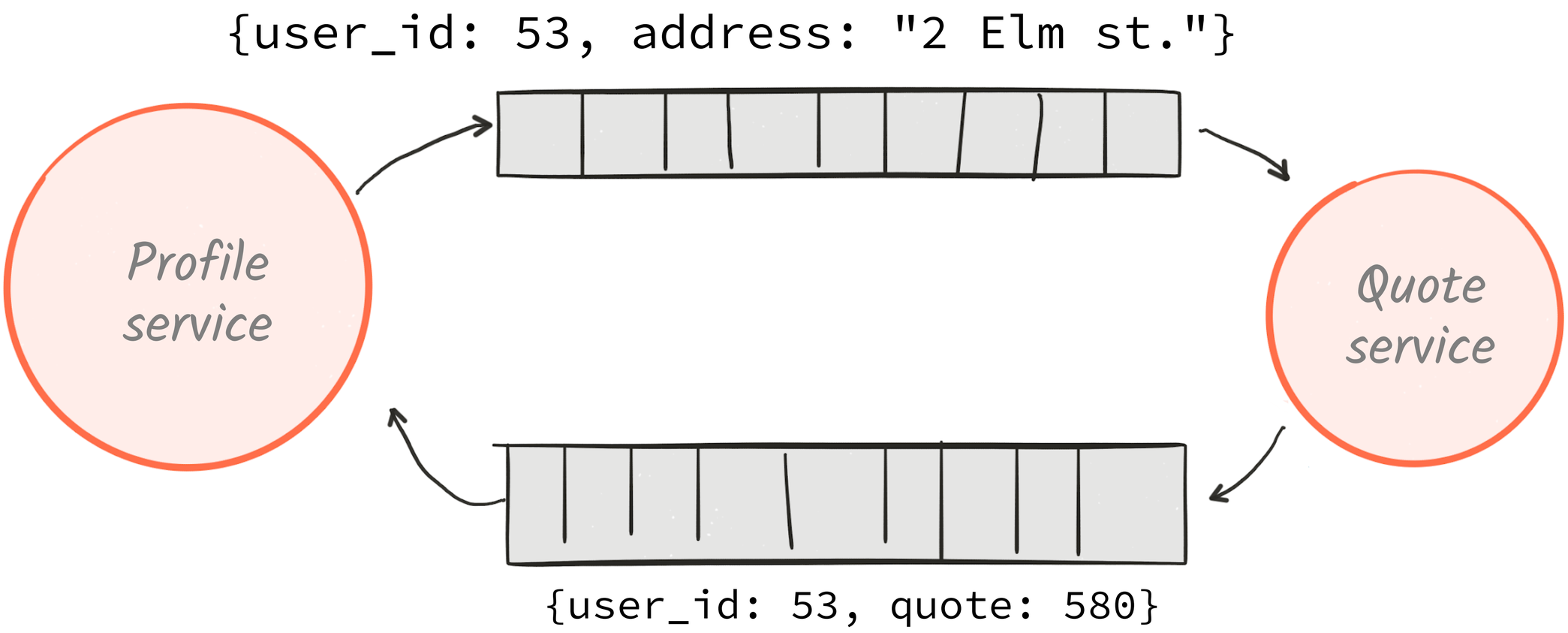
How do you serialise your data?

Avro Protobuf JSON Schema JSON CSV

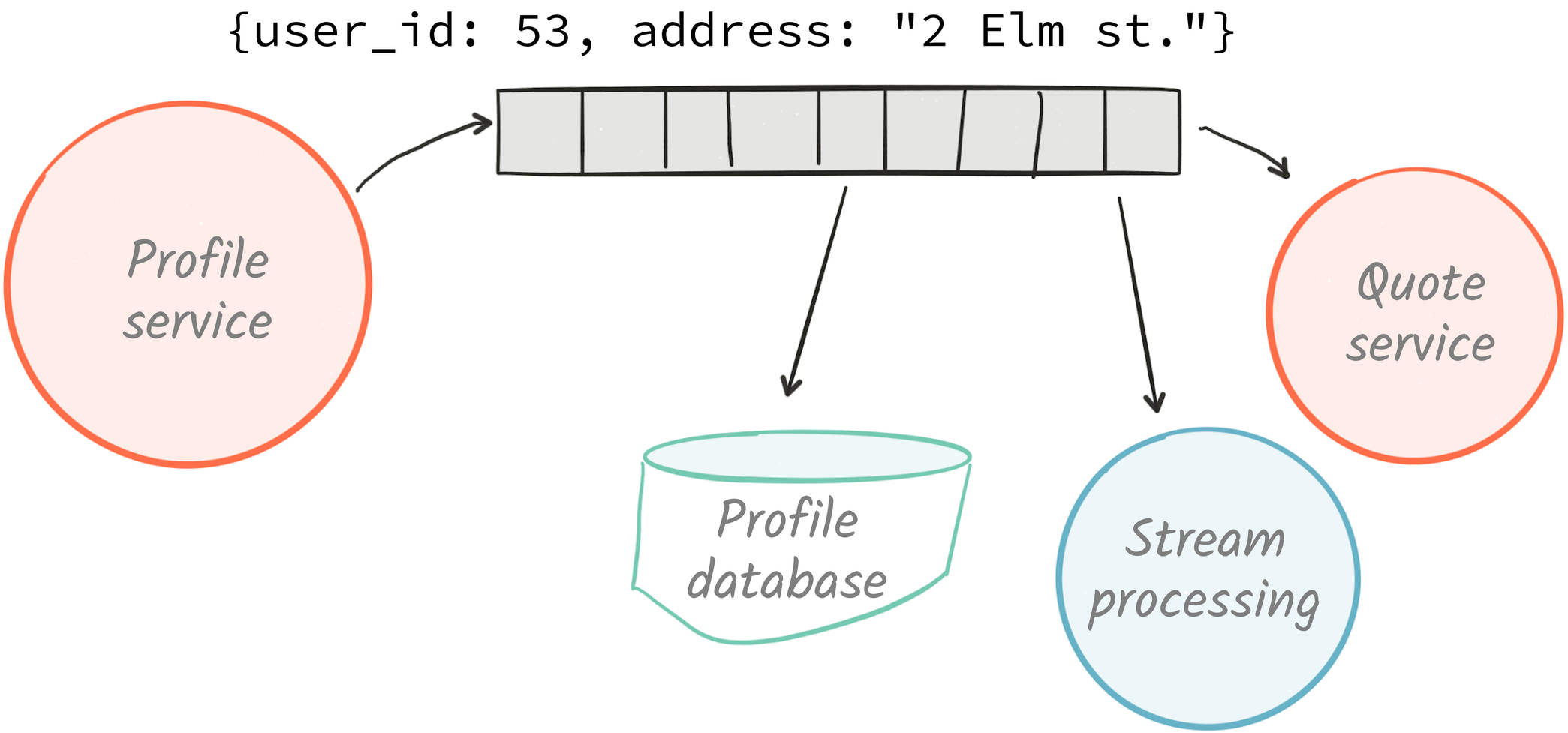
APIs are contracts between services



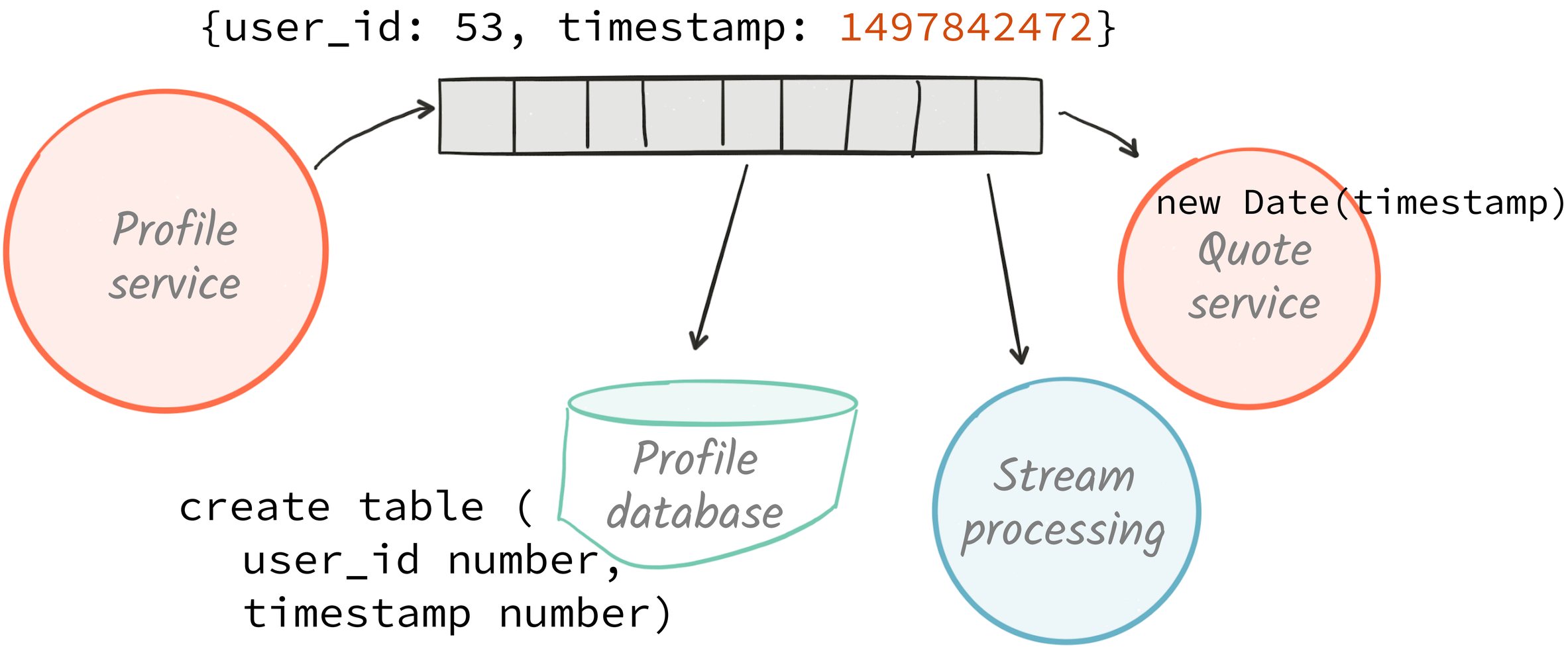
But not all services



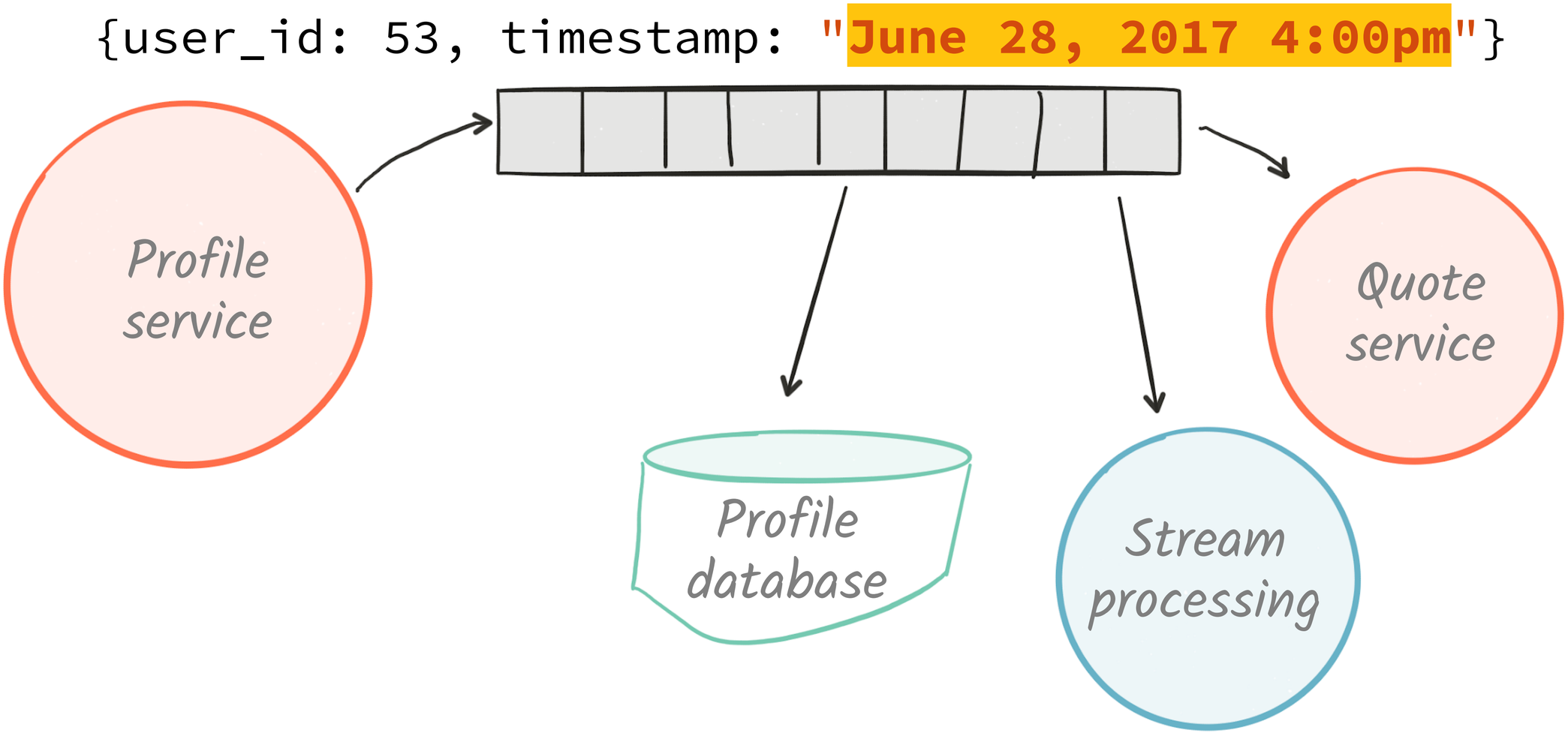
And naturally...



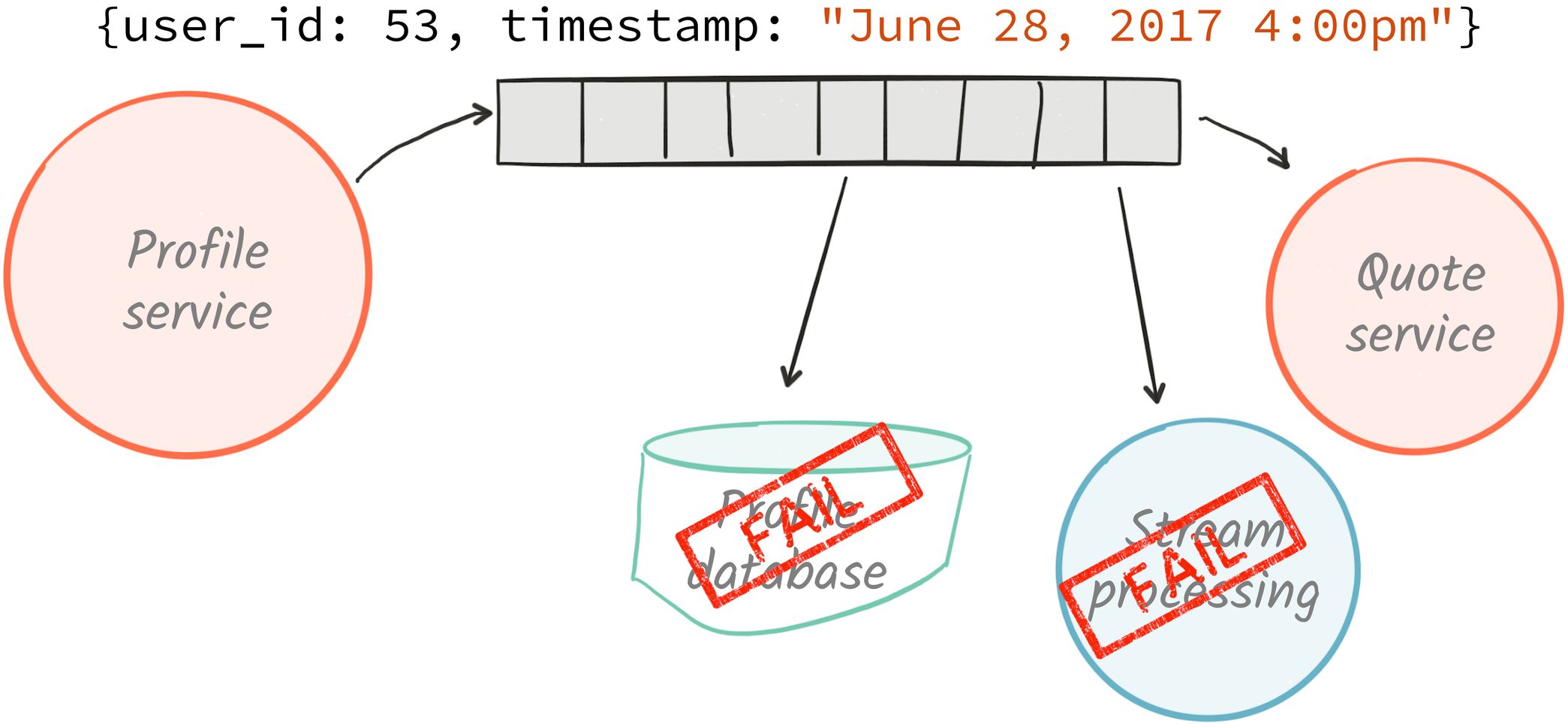
Schemas are about how teams work together



Things change...



Moving fast and breaking things



Lack of schemas - Coupling teams and services

```
2001 2001 Citrus Heights-Sunrise Blvd
Citrus_Hghts 60670001  3400293 34 SAC
Sacramento SV Sacramento Valley SAC
Sacramento County APCD SMA8 Sacramento
Metropolitan Area CA 6920 Sacramento 28 6920
13588 7400 Sunrise Blvd9561038 41 56
38.6988889 121  16 15.98999977
-121.271111  10 4284781 650345  52
```

Serialisation & Schemas

Avro Protobuf JSON Schema JSON CSV

Serialisation & Schemas

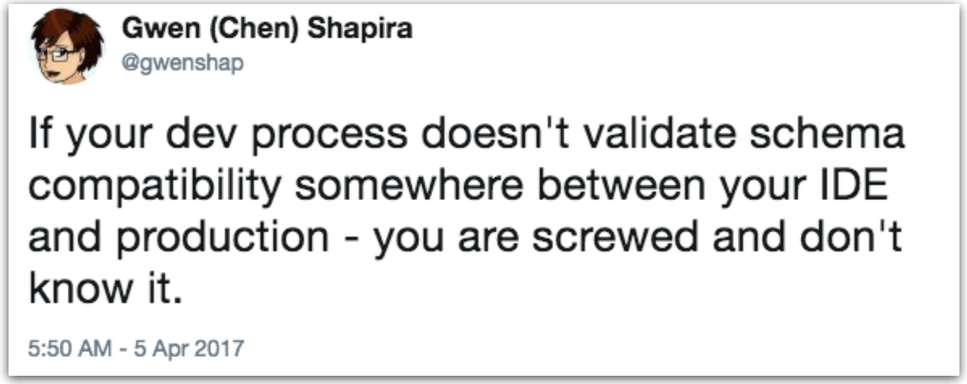
Avro 👍

Protobuf 👍

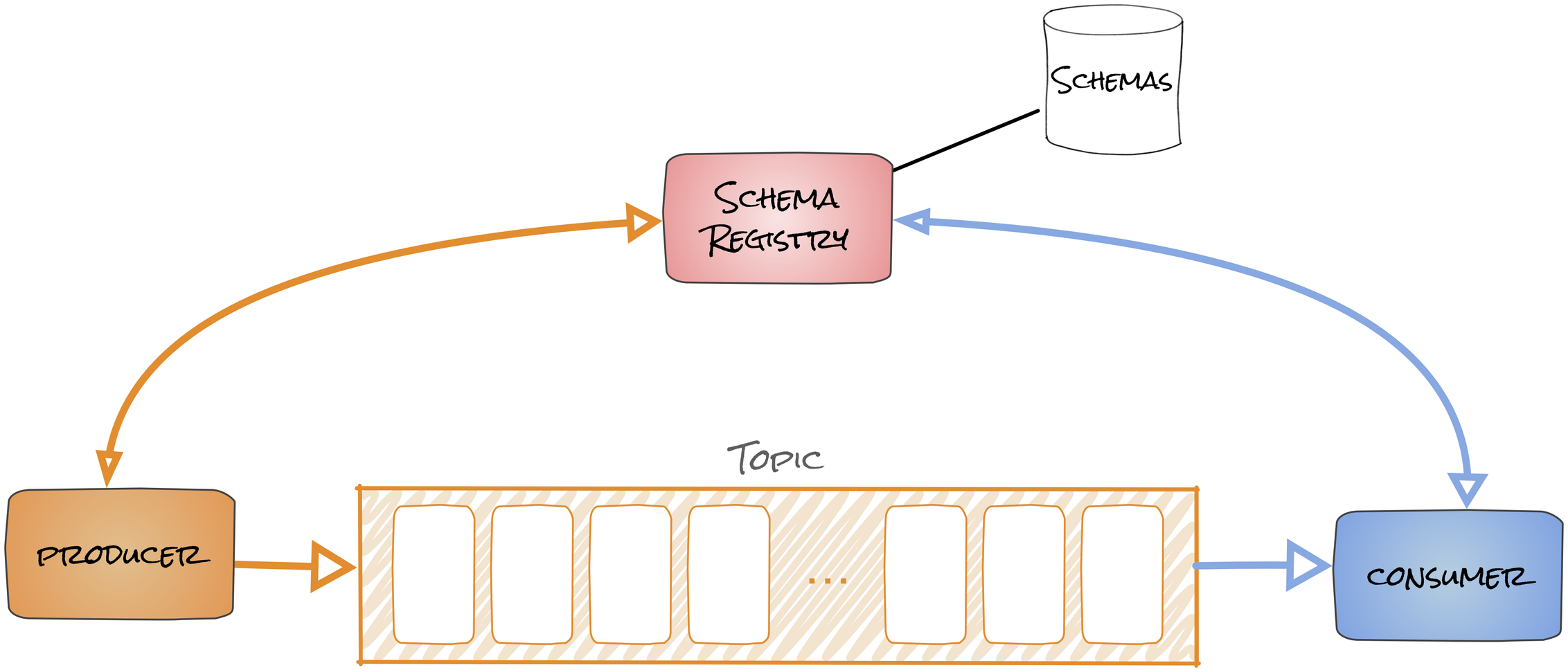
JSON Schema 👍

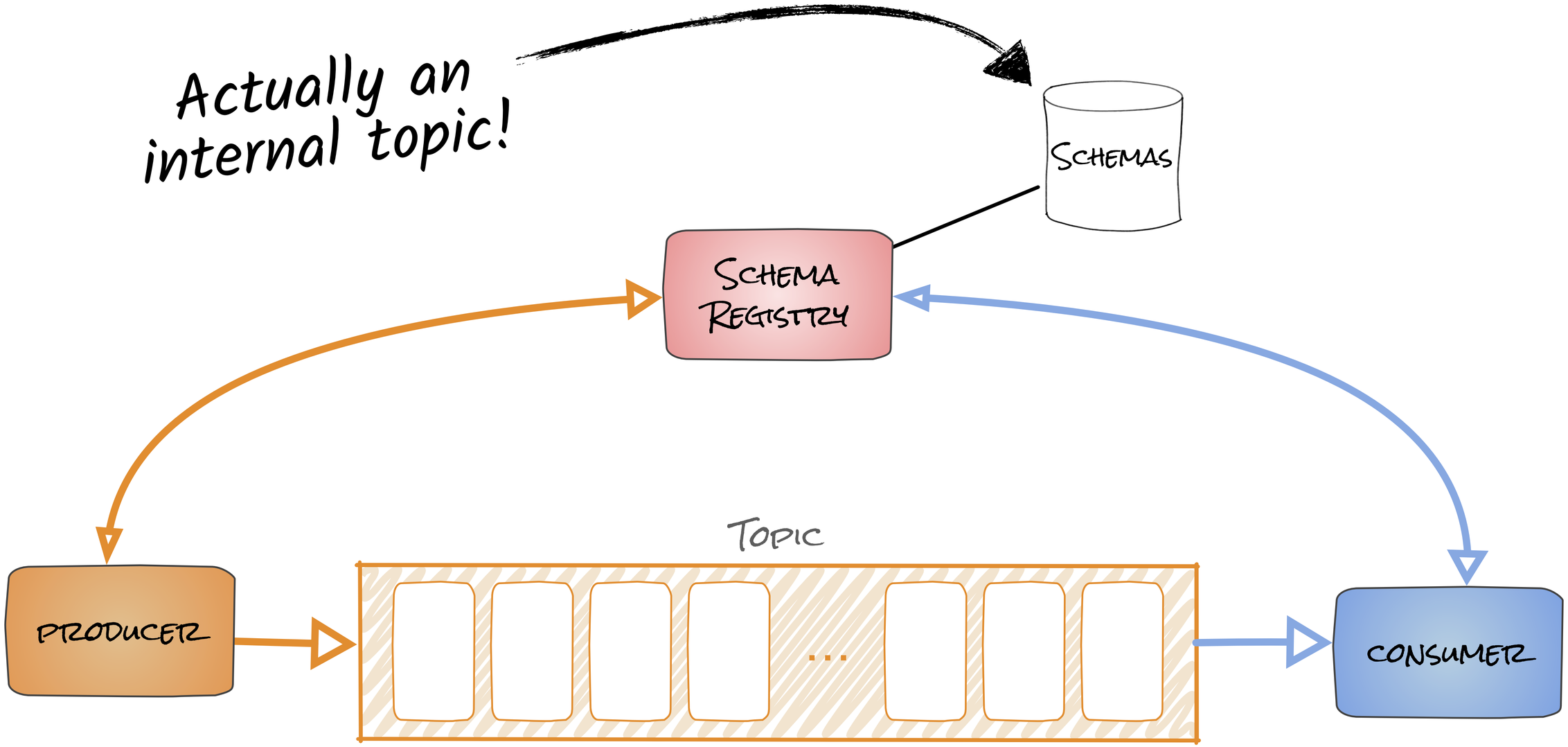
JSON 😬

CSV 



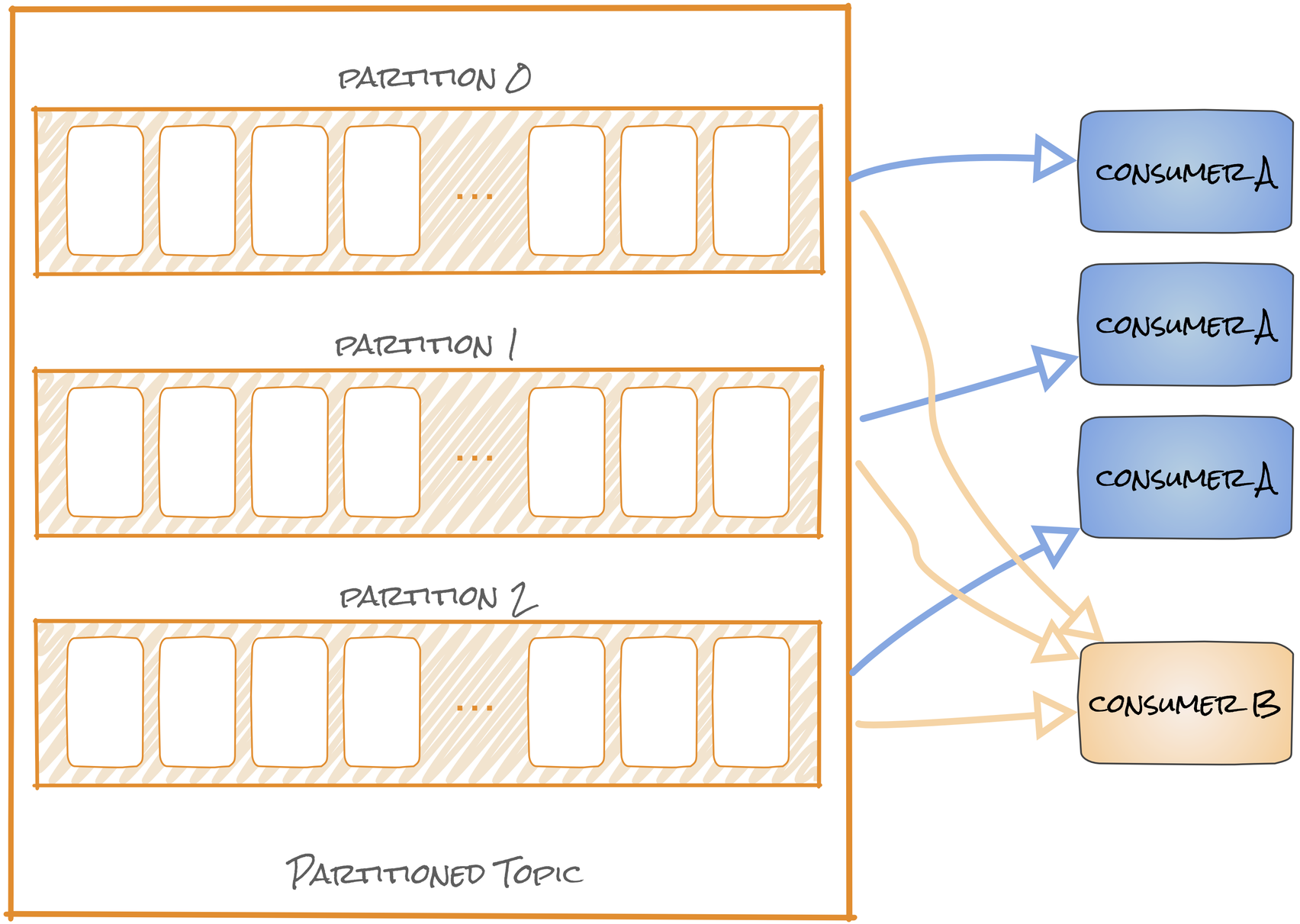
https://qconnewyork.com/system/files/presentation-slides/qcon_17_-_schemas_and_apis.pdf


SCHEMAS
SCHEMA REGISTRY
TOPIC
PRODUCER
CONSUMER


Actually an internal topic!
Schemas
Schema Registry
Topic
Producer
Consumer

Producers contain serializers

```
props.put("key.serializer",
    "org.apache.kafka.serializers.StringSerializer");
props.put("value.serializer",
    "io.confluent.kafka.serializers.KafkaAvroSerializer");
props.put("schema.registry.url",
          "http://schema-registry:8081");
…
producer<String, LogLine> producer = new
  KafkaProducer<String, LogLine>(props);
```


PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC
CONSUMER A
CONSUMER A
CONSUMER A
CONSUMER B

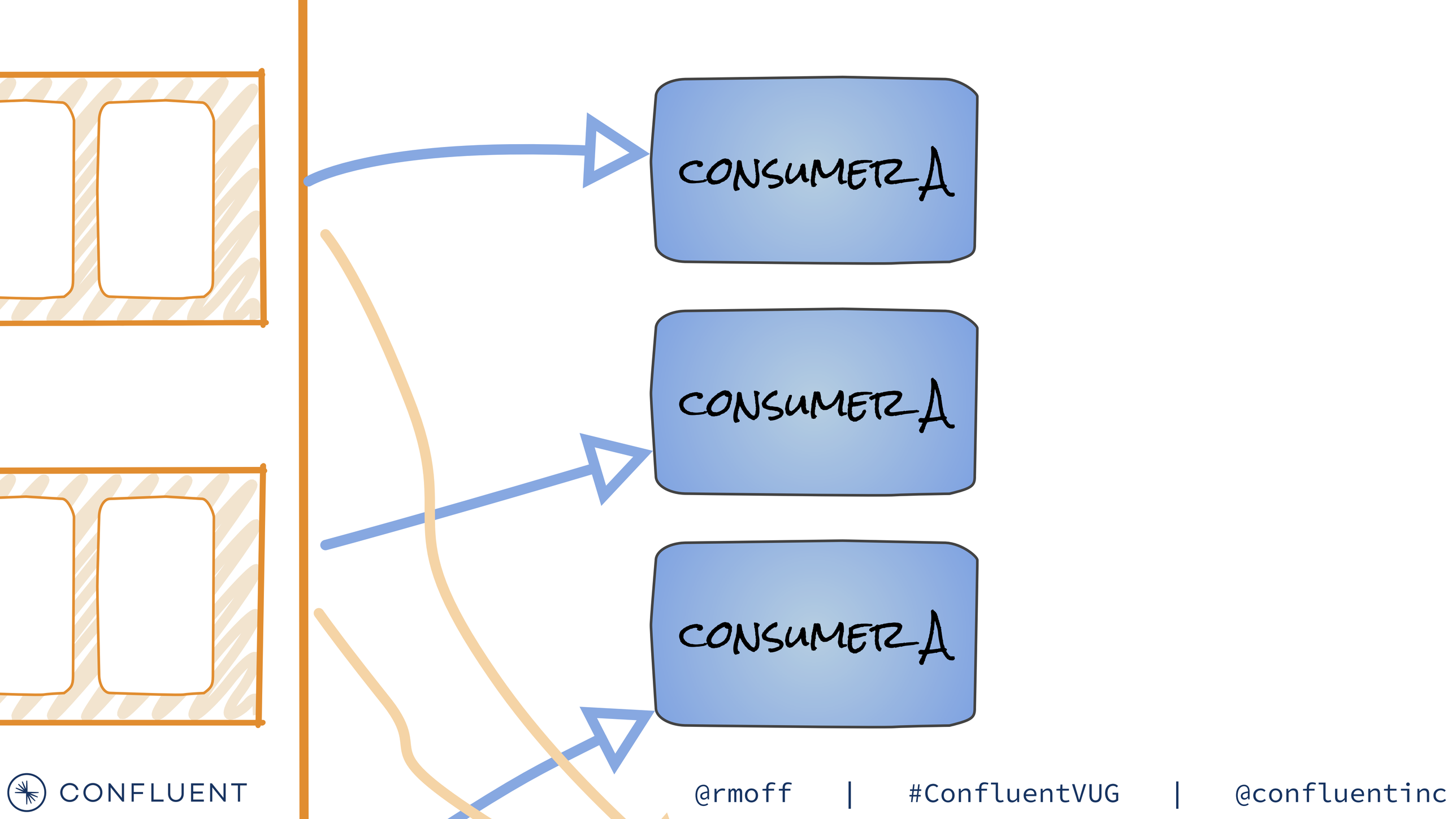
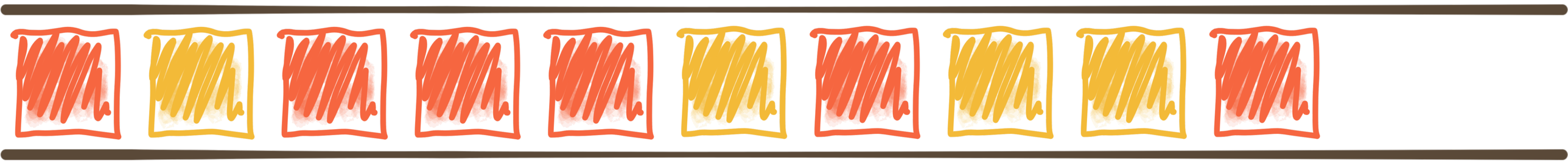

CONSUMER A
CONSUMER A
CONSUMER A



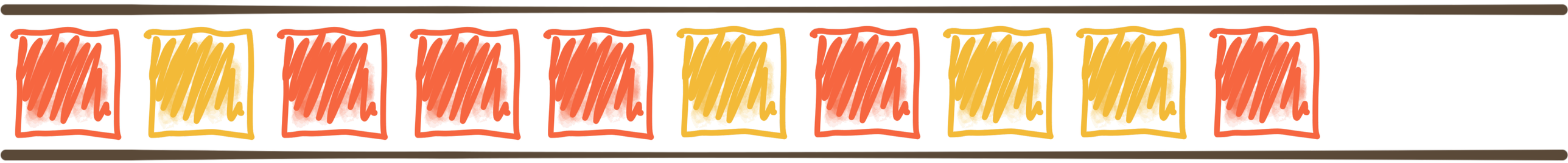
Photo by Franck V. on Unsplash

Streams of events



Time

Stream Processing



Stream: widgets



Stream: widgets_red

Stream Processing with Kafka Streams

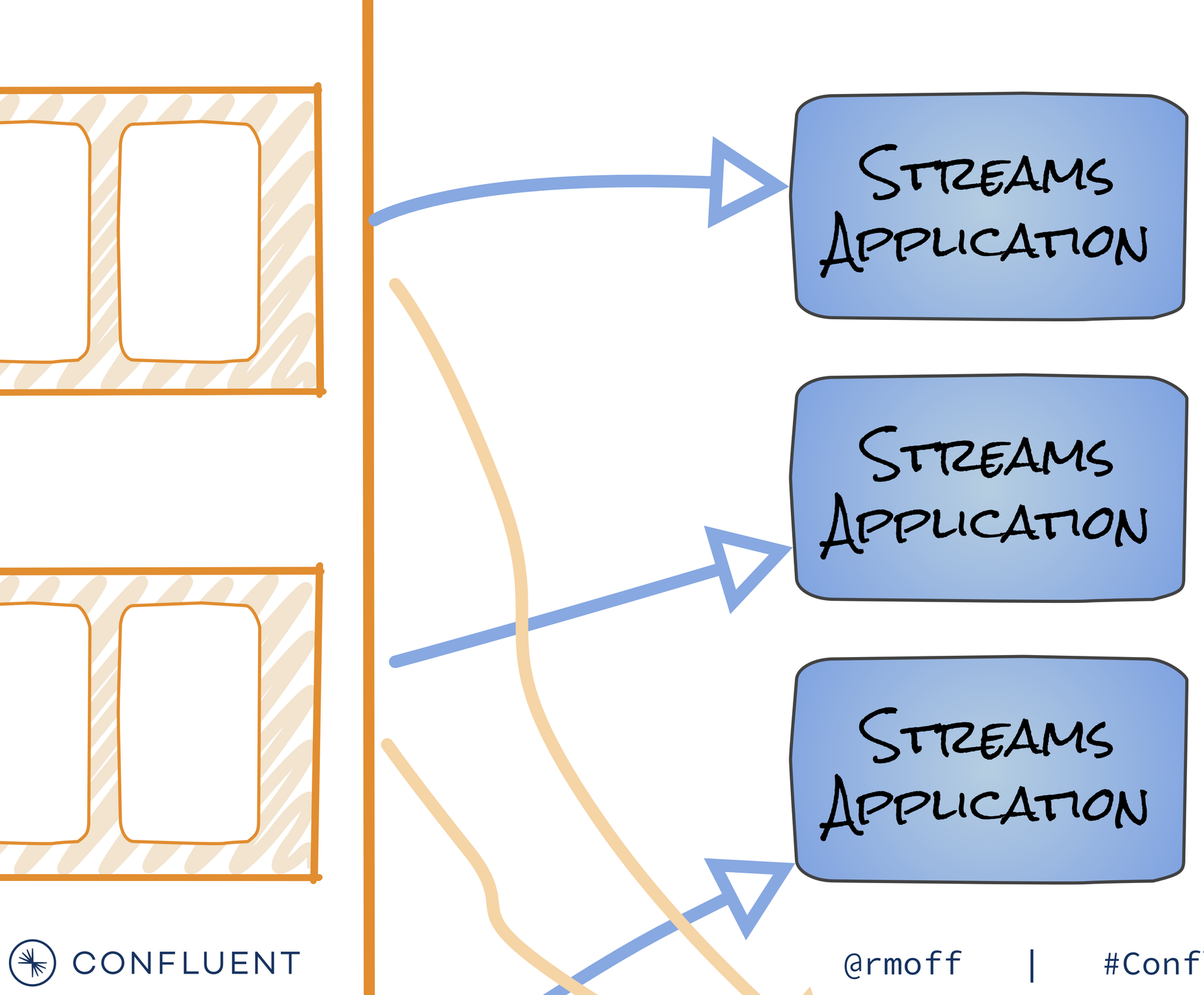


Stream:
widgets

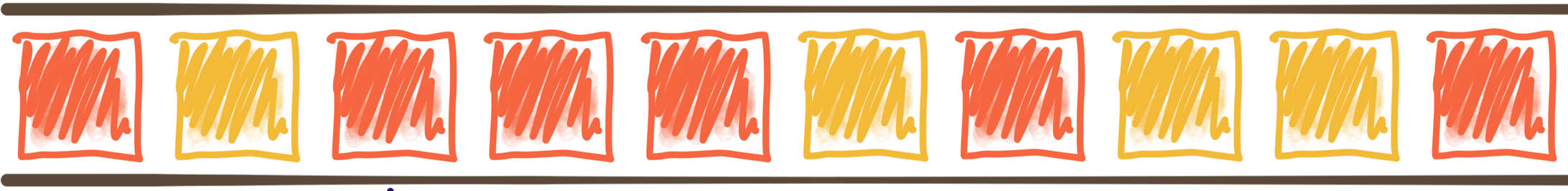
```
final StreamsBuilder builder = new StreamsBuilder()
    .stream("widgets", Consumed.with(stringSerde, widgetsSerde))
    .filter( (key, widget) -> widget.getColour().equals("RED") )
    .to("widgets_red", Produced.with(stringSerde, widgetsSerde));
```



Stream:
widgets_red


Streams Application
Streams Application
Streams Application

Stream Processing with ksqlDB



Stream:
widgets

ksqlDB

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



Stream:
widgets_red



Photo by Franck V. on Unsplash



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



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



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

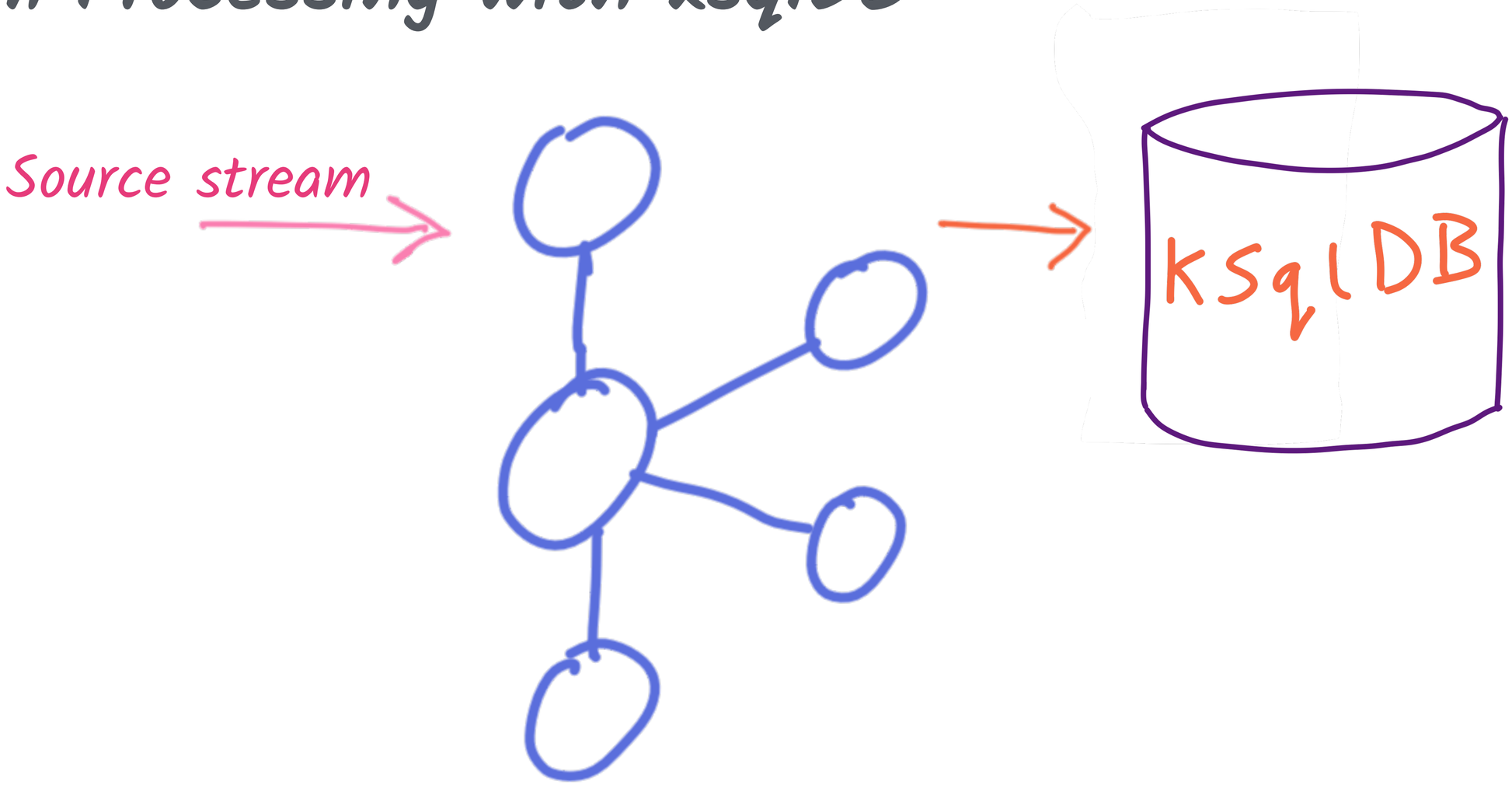


Object store, data warehouse, RDBMS

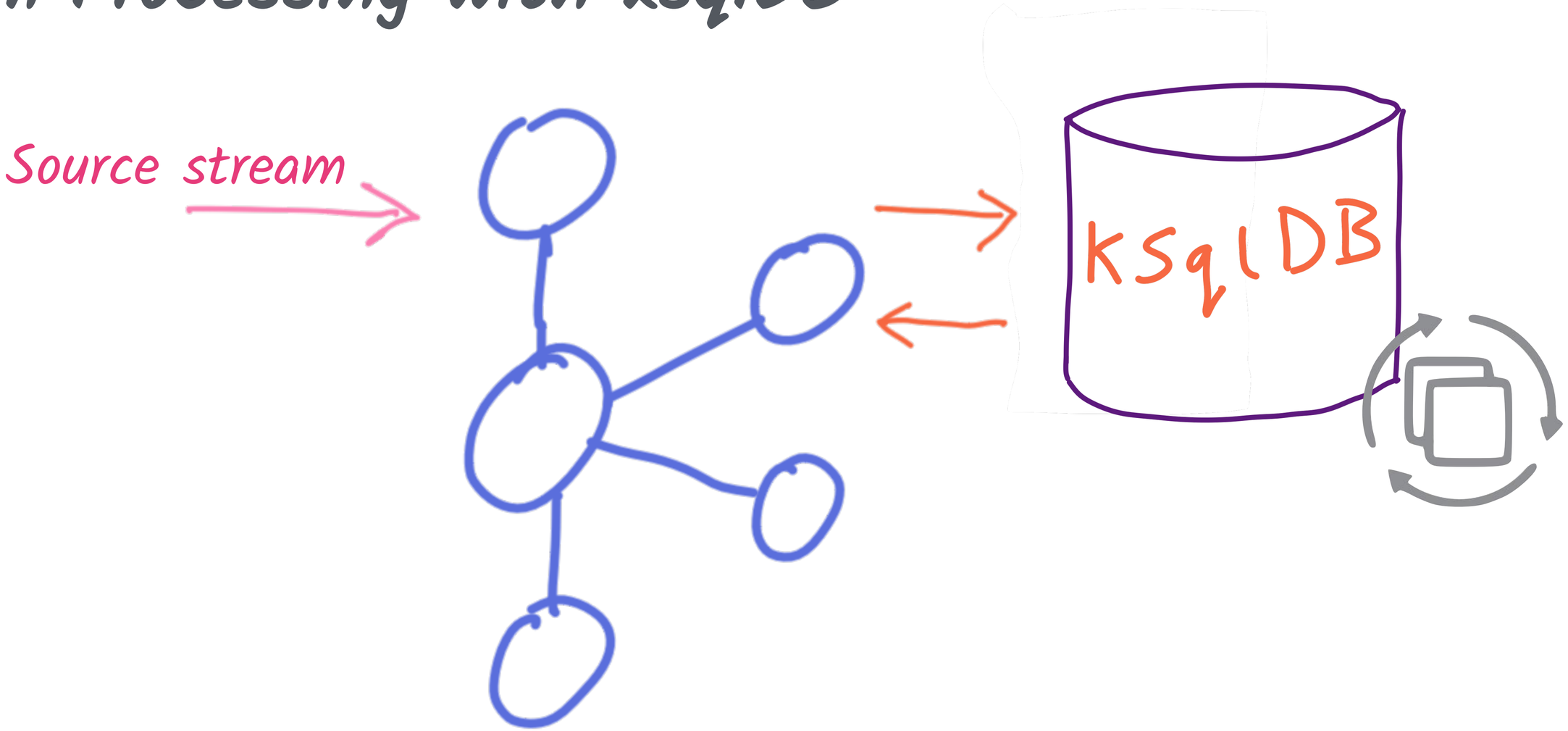
```
CREATE SINK CONNECTOR dw WITH (
'connector.class' = 'S3Connector',
'topics' = 'widgets'
…);
```

Photo by Franck V. on Unsplash

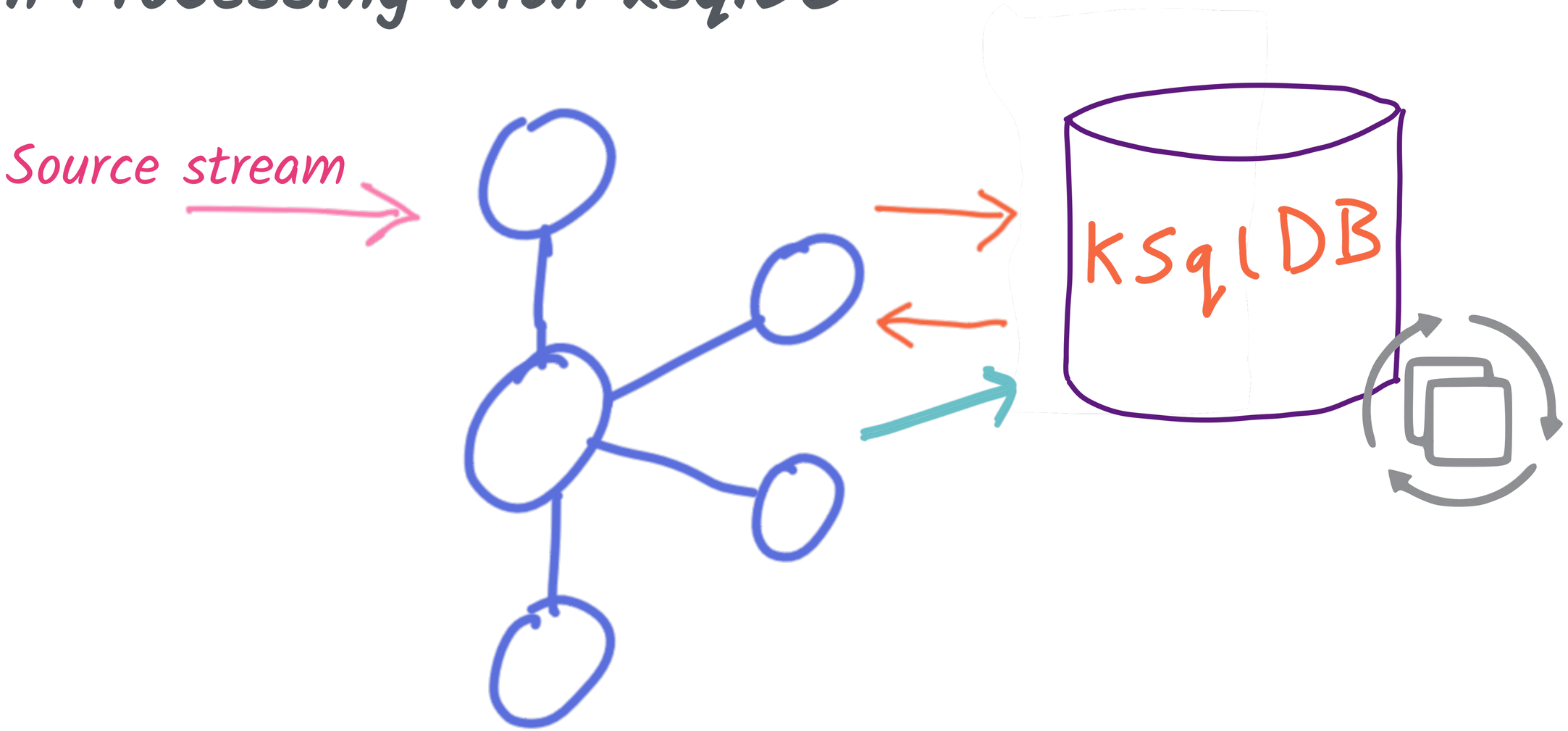
Stream Processing with ksqlDB



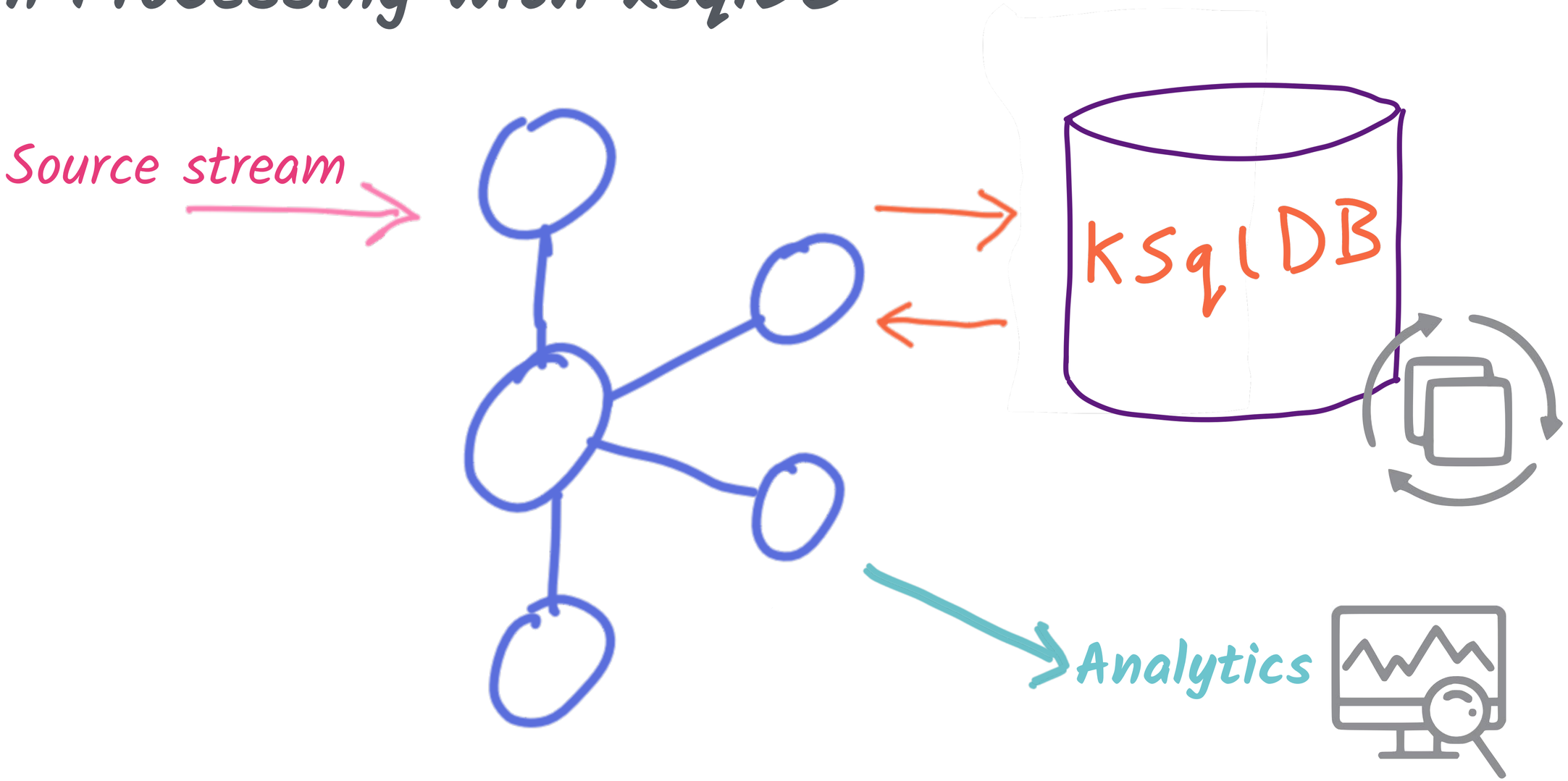
Stream Processing with ksqlDB



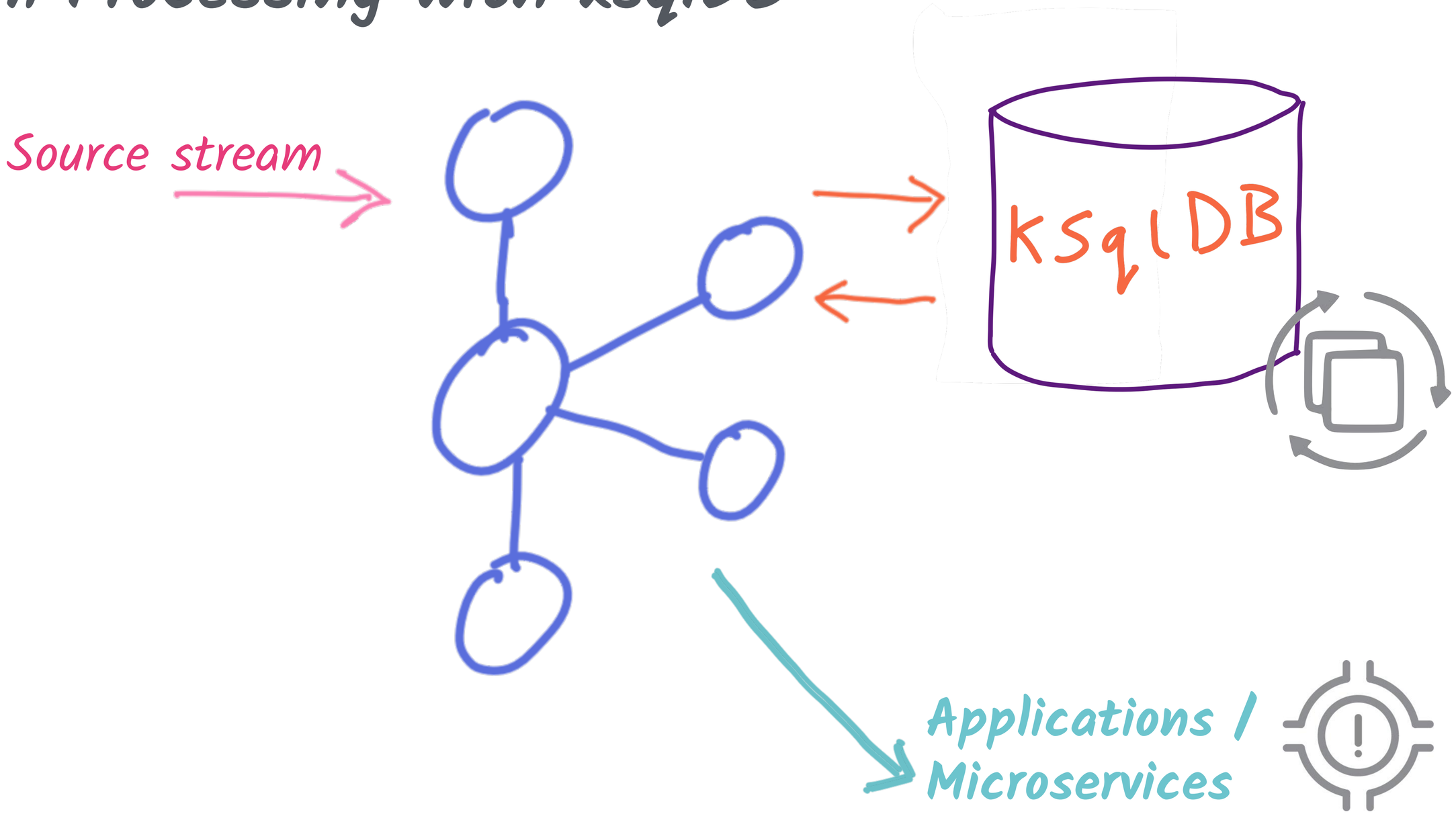
Stream Processing with ksqlDB



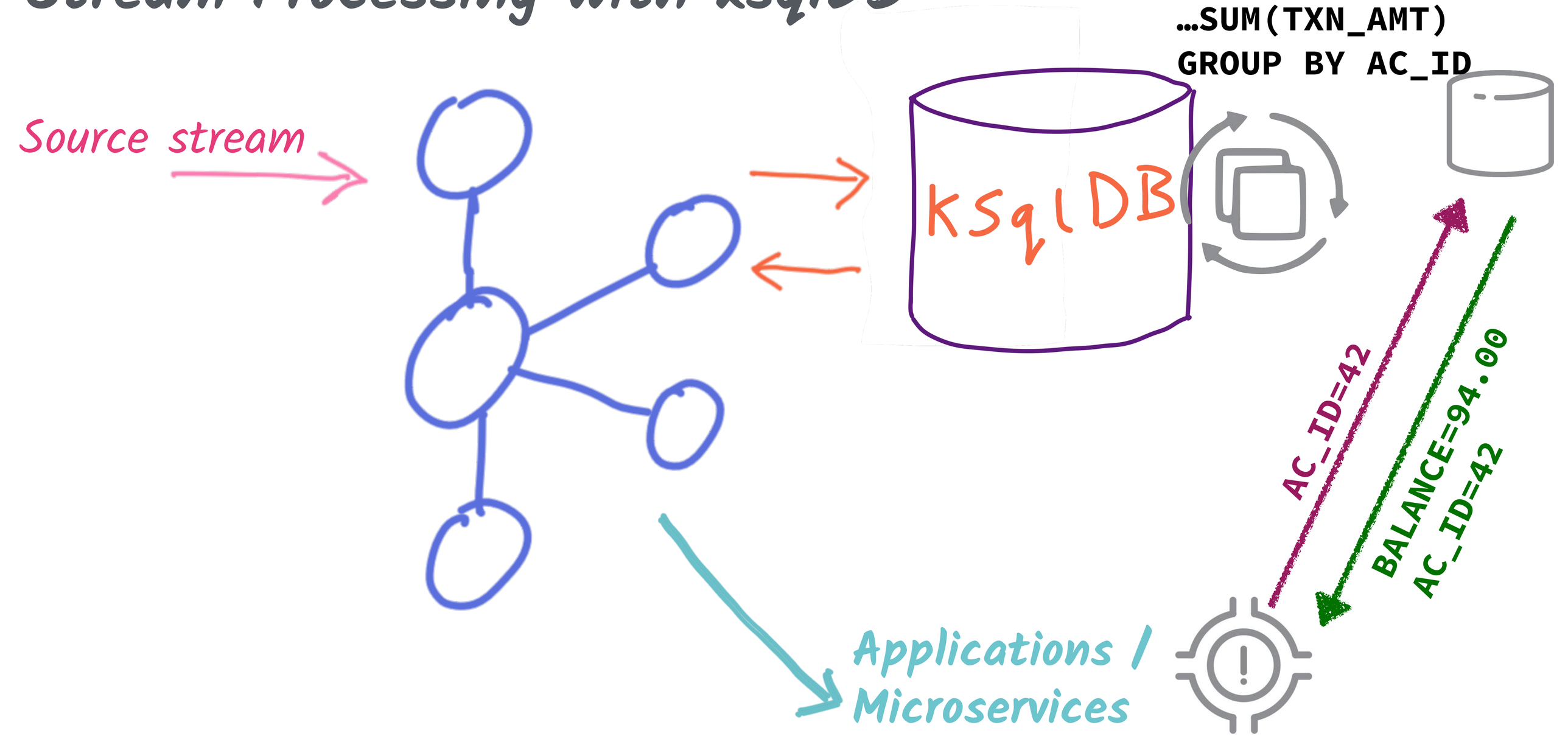
Stream Processing with ksqlDB



Stream Processing with ksqlDB



Stream Processing with ksqlDB



CONFLUENT

ksqlDB

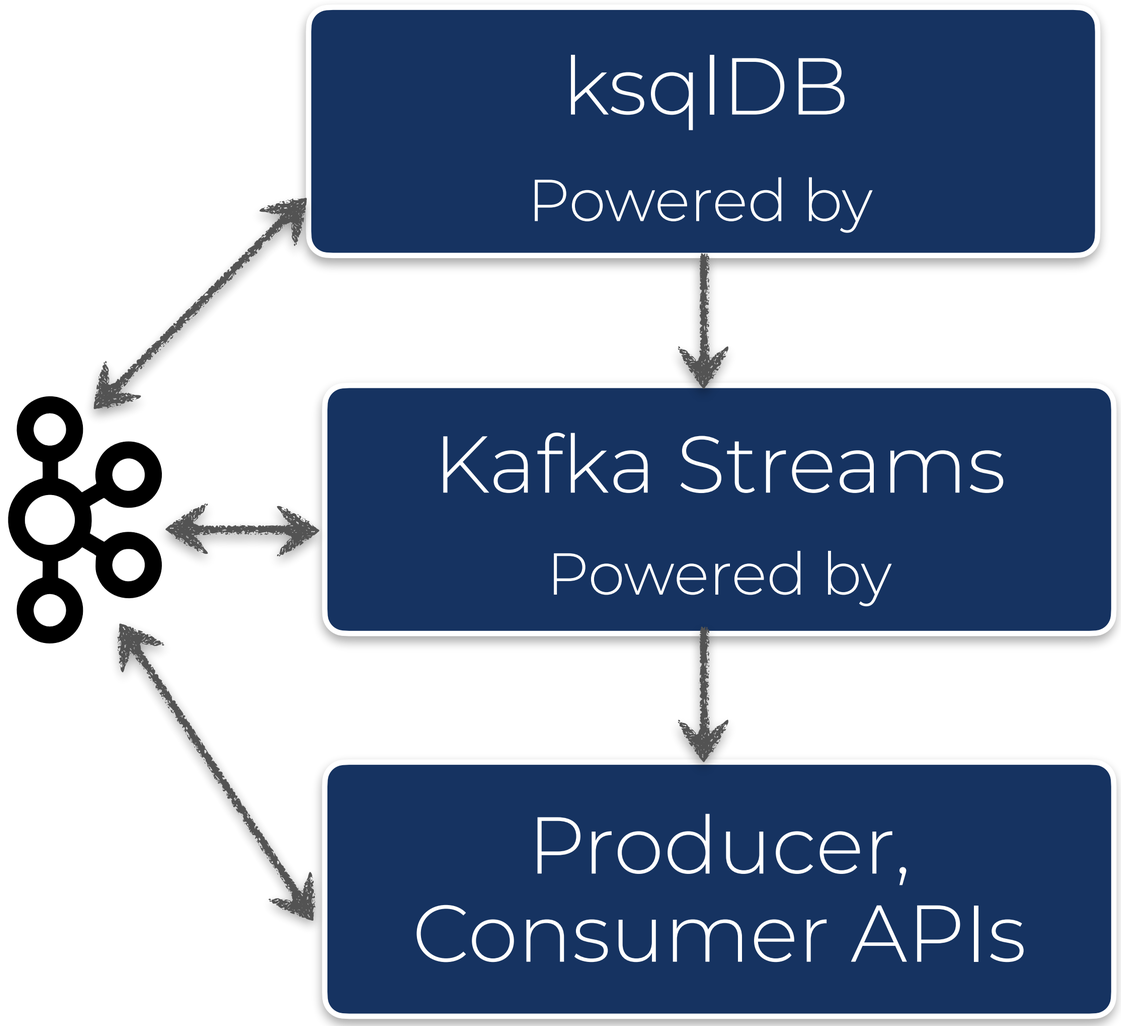
or

Kafka Streams?



Photo by Ramiz Dedaković on Unsplash

Standing on the Shoulders of Streaming Giants



Ease of use

Flexibility

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

ksqlDB UDFs

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

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

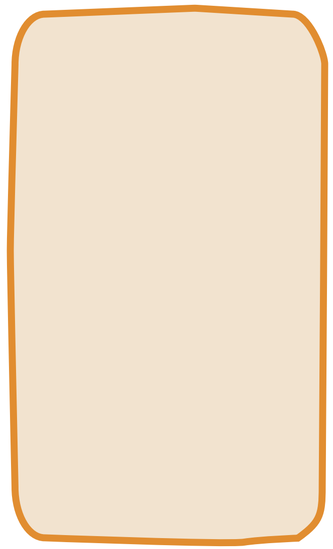


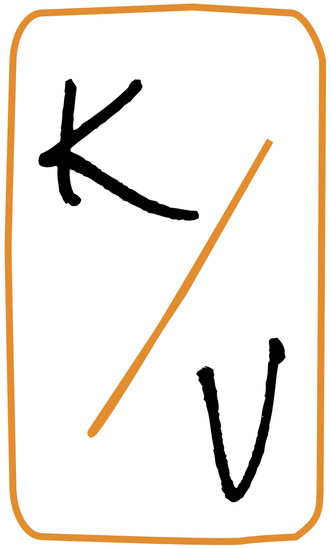
CONFLUENT

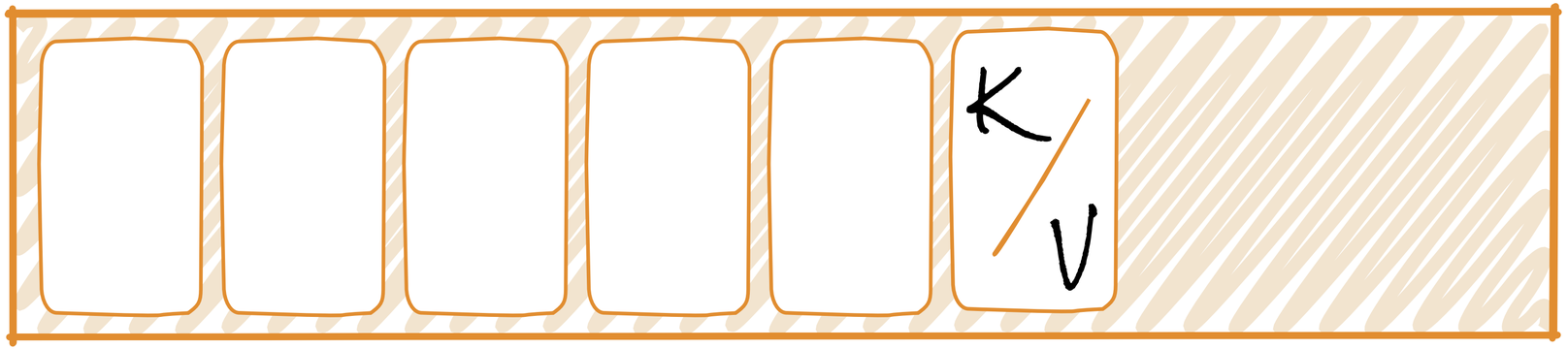
DEMO

@rmoff | #ConfluentVUG | @confluentinc

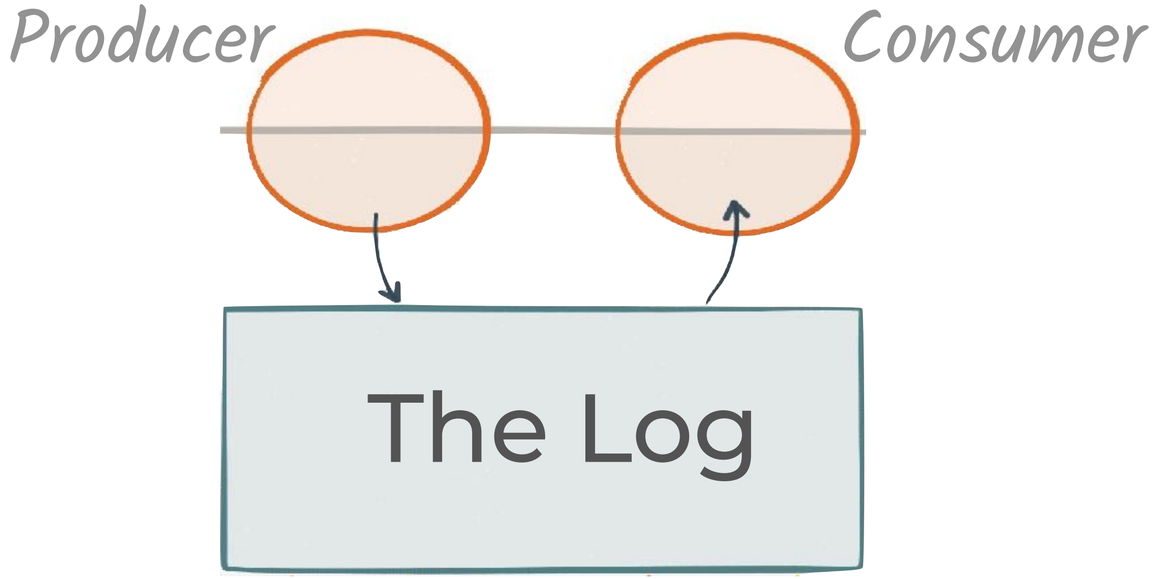
Summary

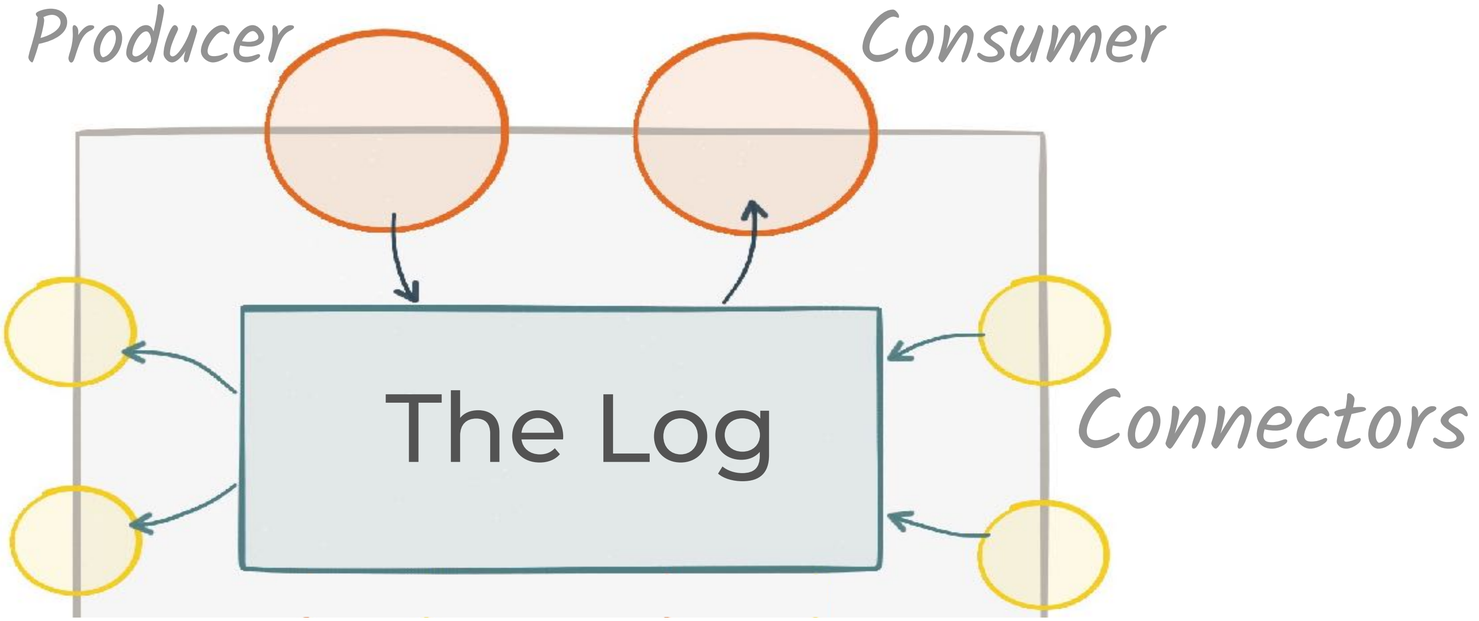


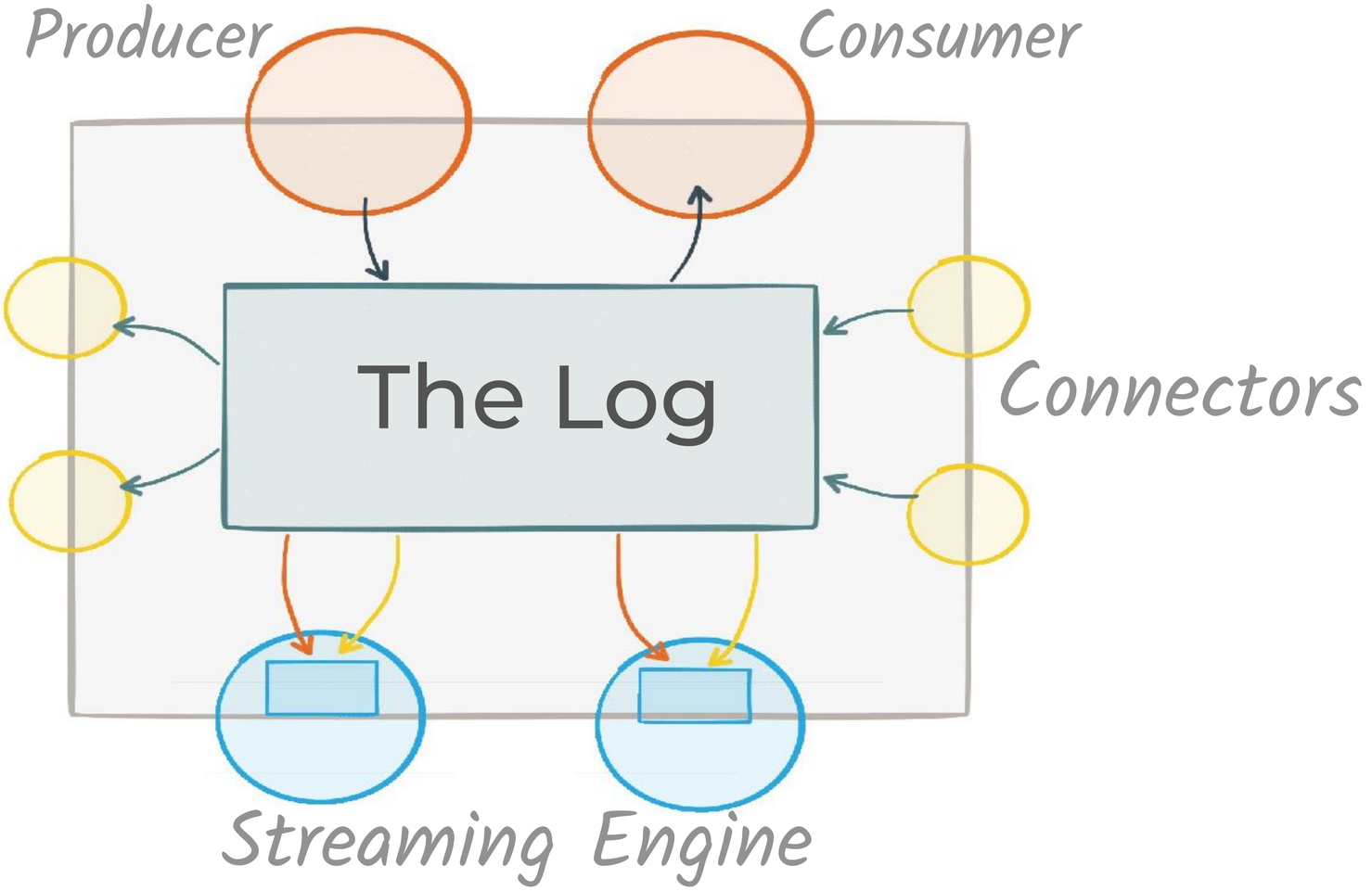

K
V


K
V

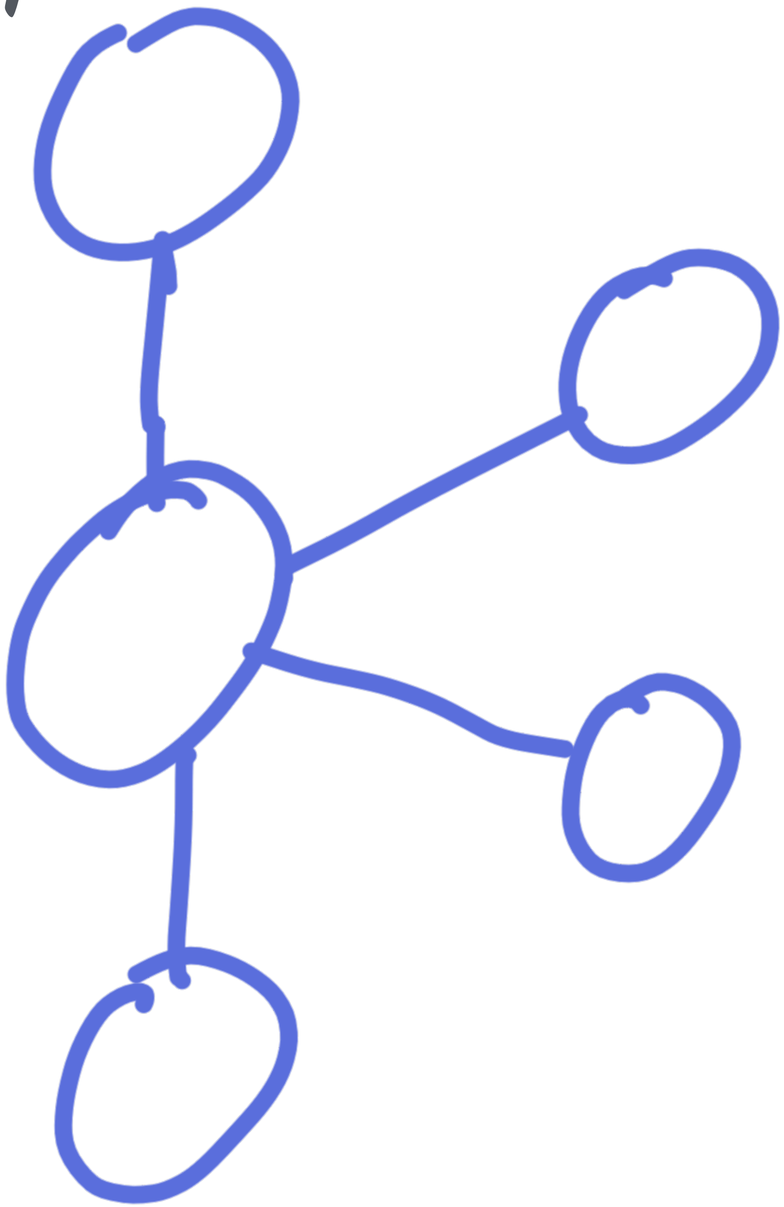
The Log

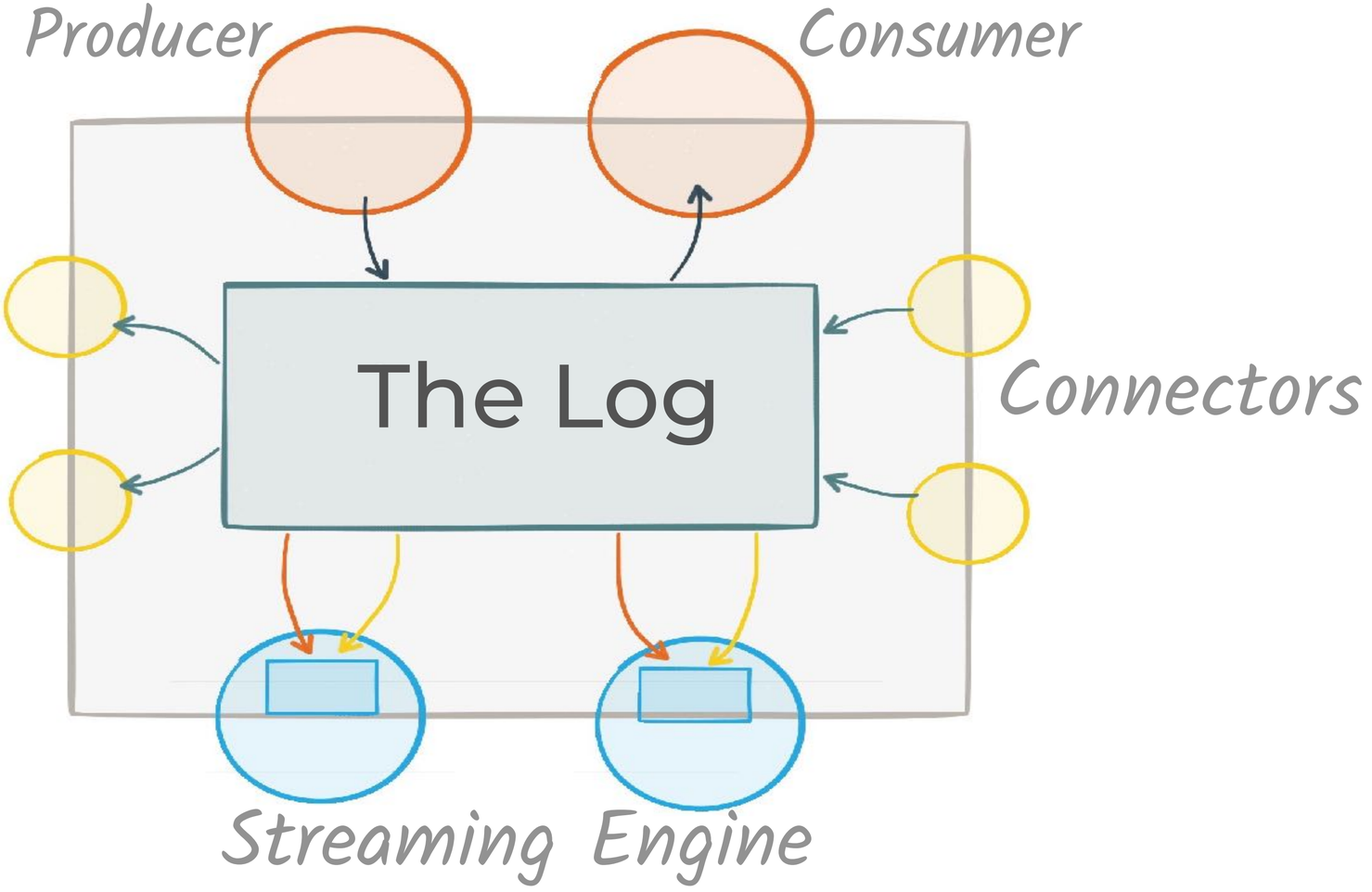

Producer
Consumer
The Log


Producer
Consumer
The Log
Connectors


Producer
Consumer
The Log
Connectors
Streaming Engine

Apache Kafka





EVENTS

are

EVERYWHERE

EVENTS

are

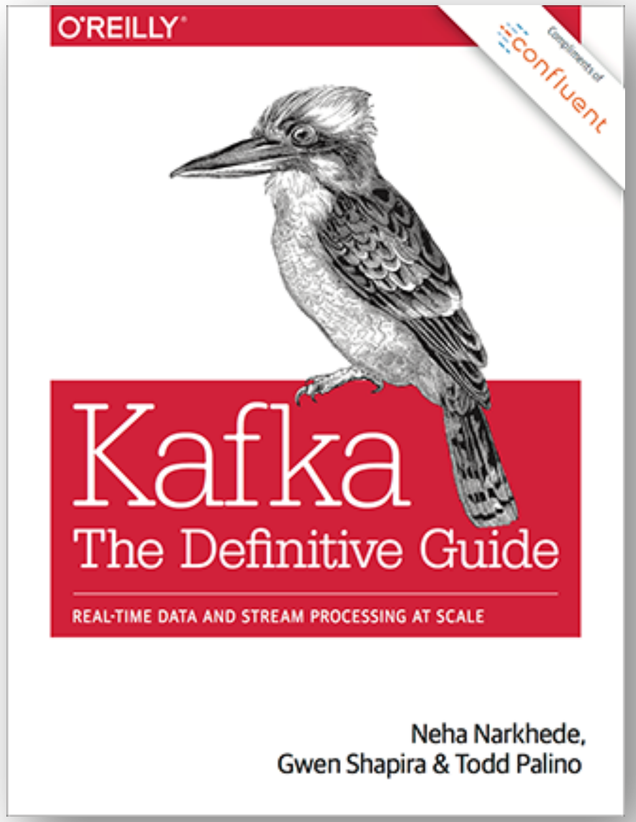
very

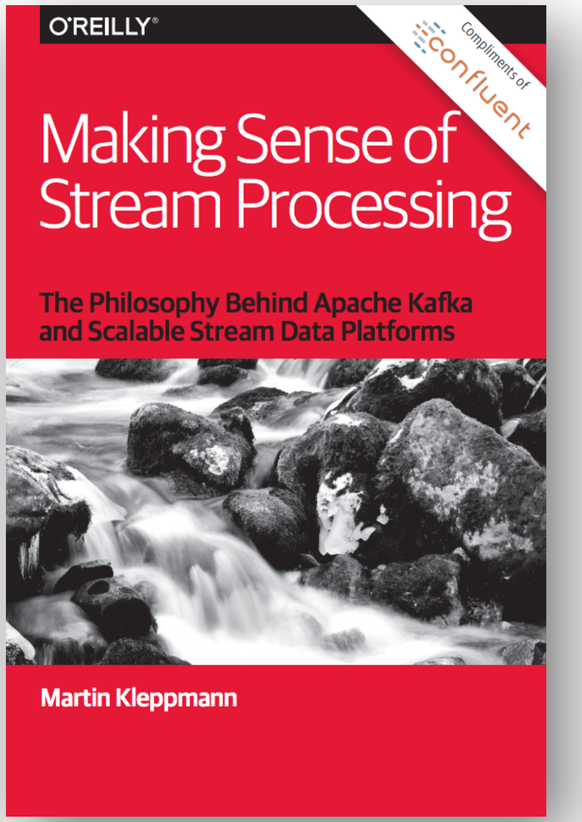
POWERFUL

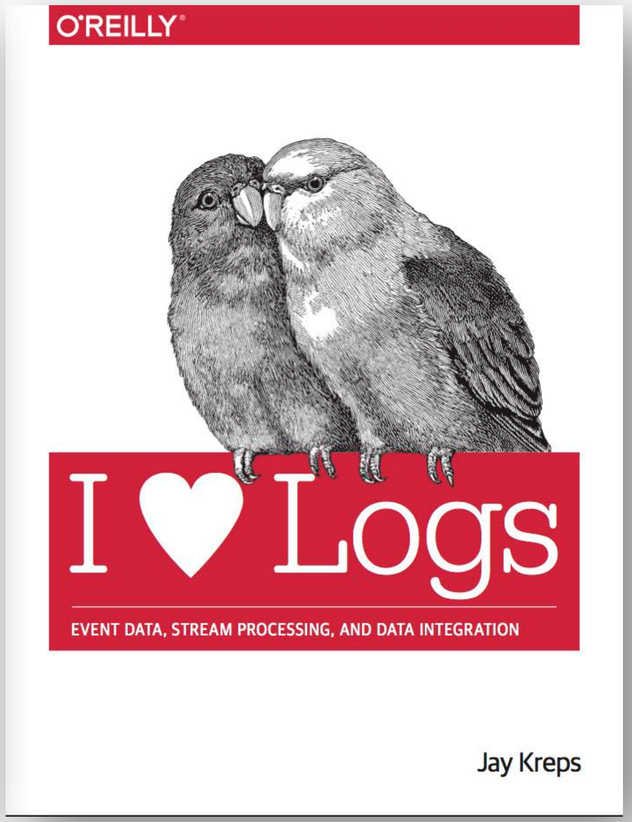
Standby for resource links...

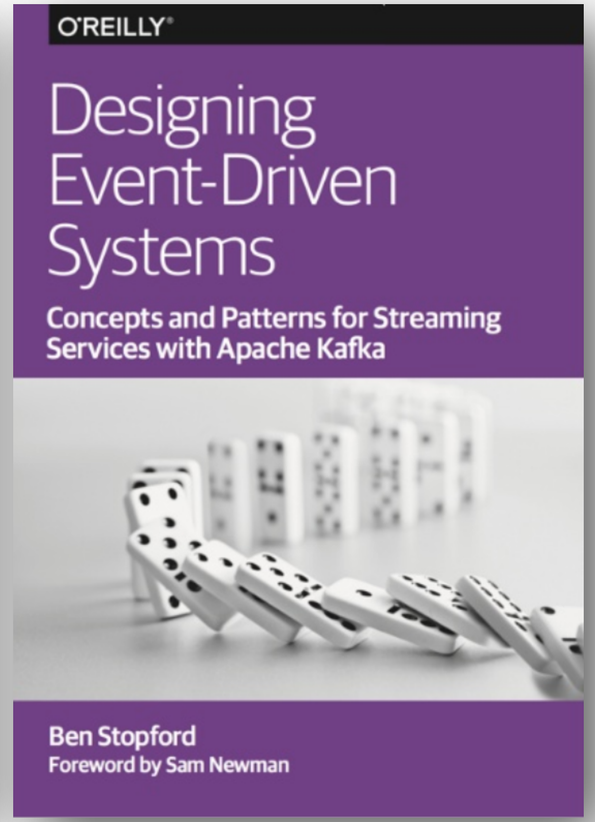
Free Books!

https://rmoff.dev/6hz









Learn Kafka.

Start building with Apache Kafka at Confluent Developer.



developer.confluent.io

Confluent Community Slack group



cnfl.io/slack



Fully Managed Kafka as a Service

@rmoff #ConfluentVUG

#EOF

https://events.confluent.io/meetups