Kafka as a Platform: the Ecosystem from the Ground Up



Robin Moffatt

@rmoff



EVENTS

EVENTS

EVENTS

- Something happened
- What happened

Human generated events

A Sale



A Stock movement

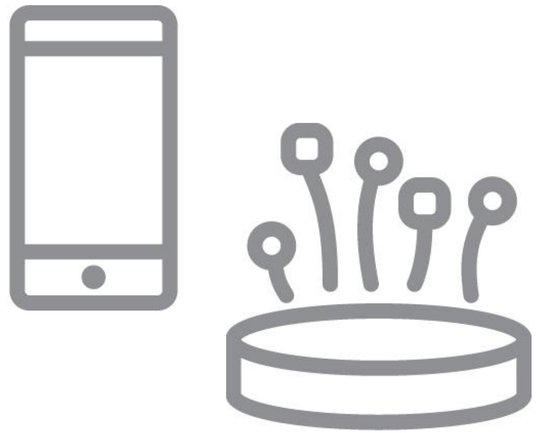


Machine generated events

Networking



IoT



Applications



EVENTS

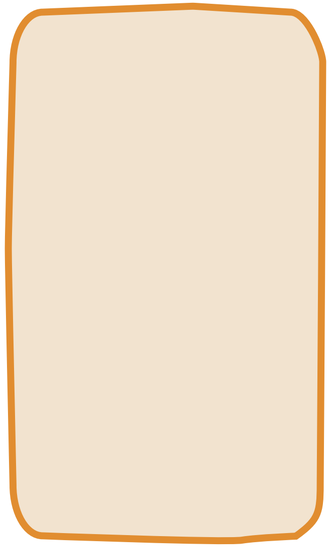
are

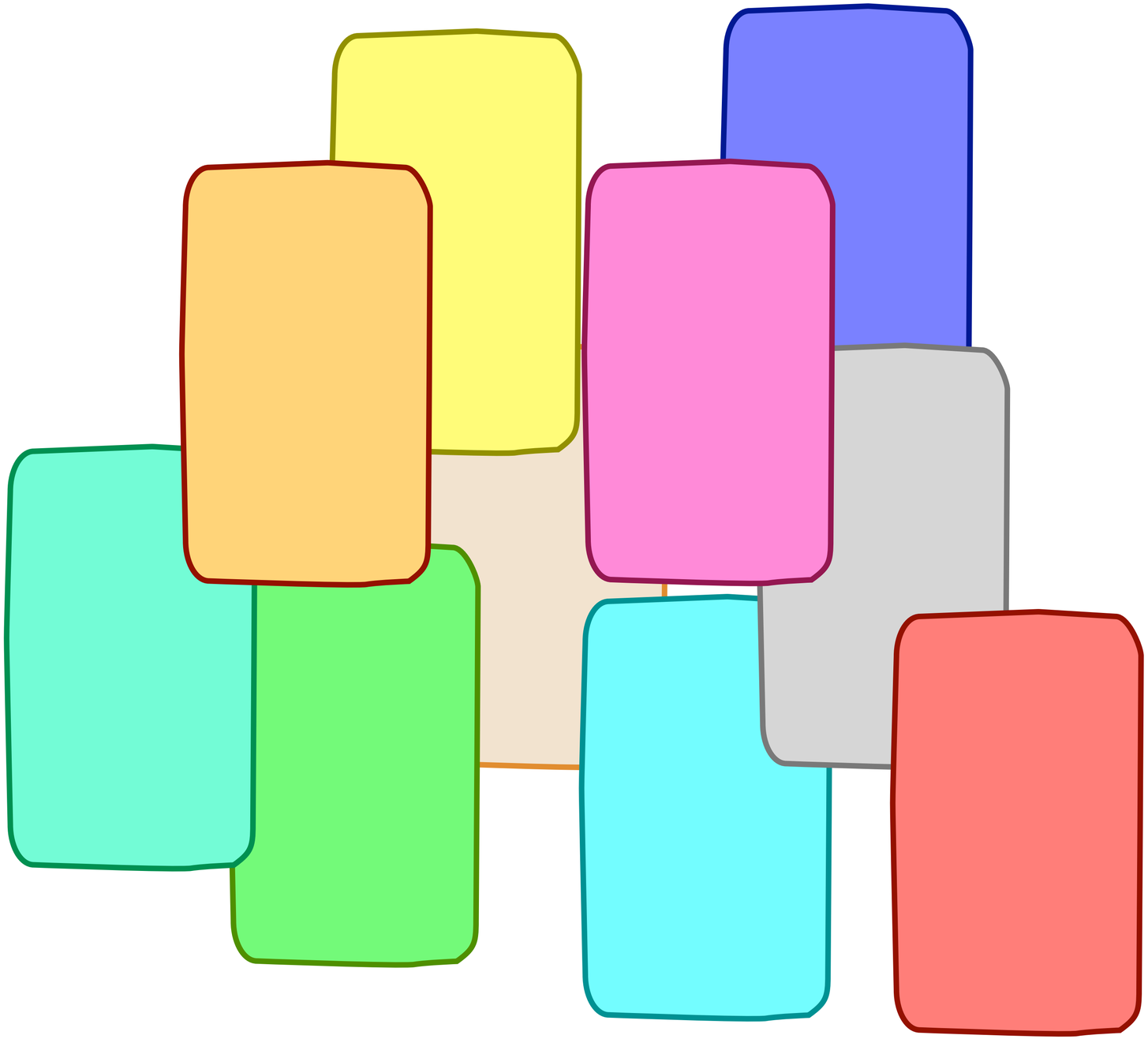
EVERYWHERE

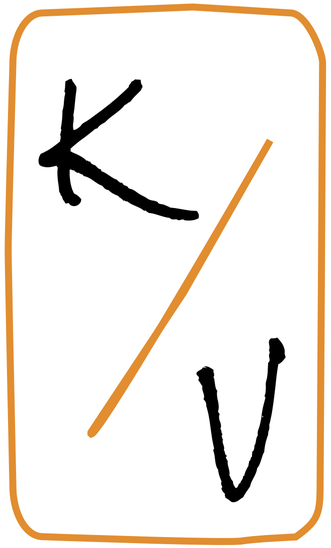
EVENTS

are

very POWERFUL






K
V



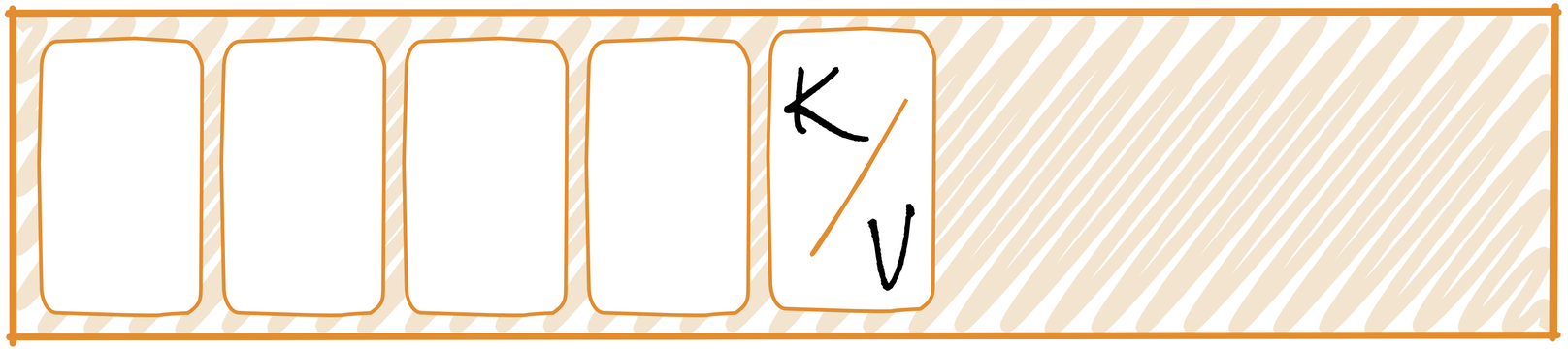
LOG

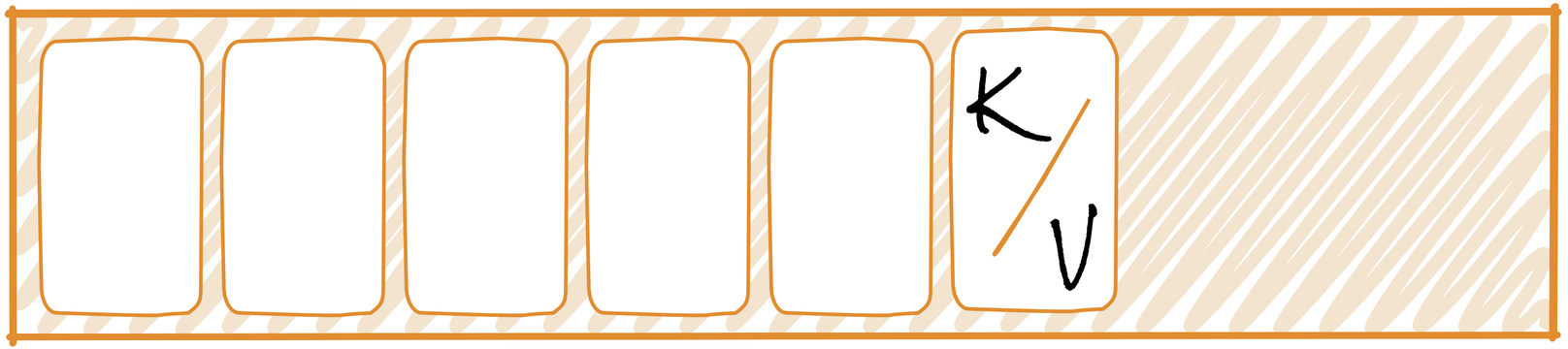

K
V

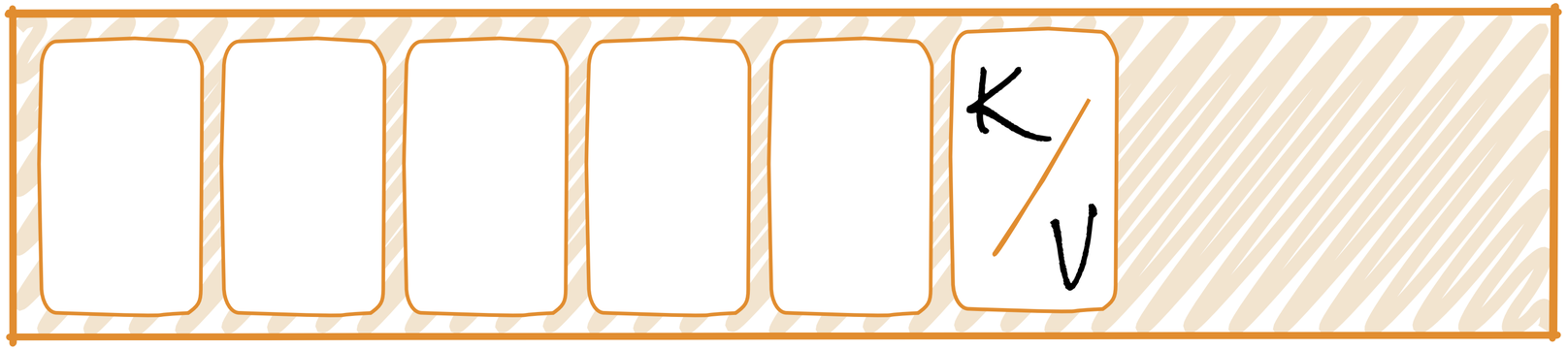

K
V


K
V

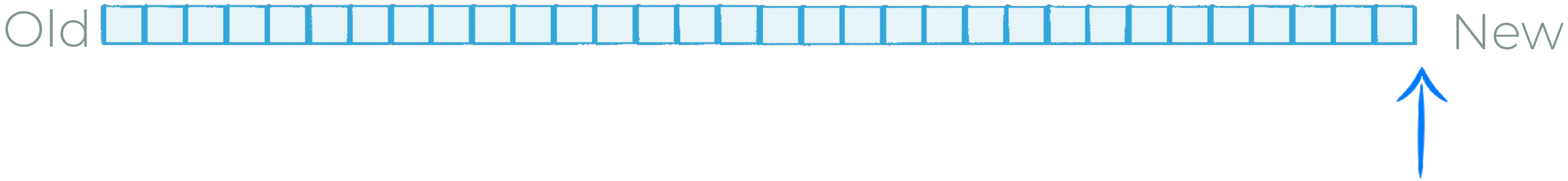

K
V


K
V


K
V


K
V

Immutable Event Log



Events are added at the end of the log



TOPICS

Topics

Clicks



Orders



Customers



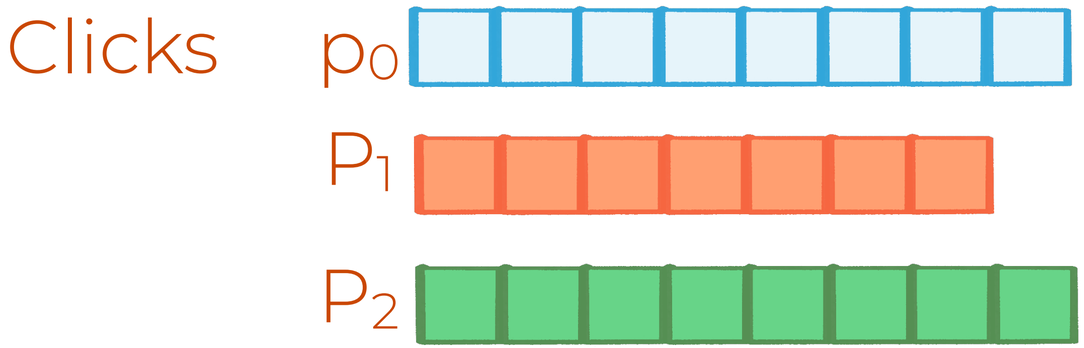
Topics are similar in concept to tables in a database



PARTITIONS

@rmoff

Partitions



Messages are guaranteed to be strictly ordered within a partition



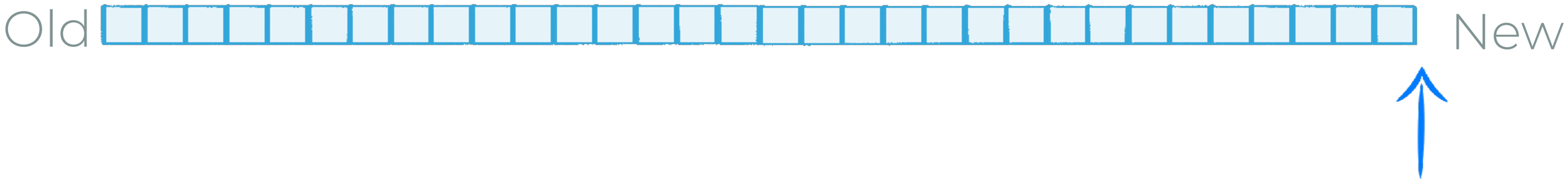
PUB / SUB



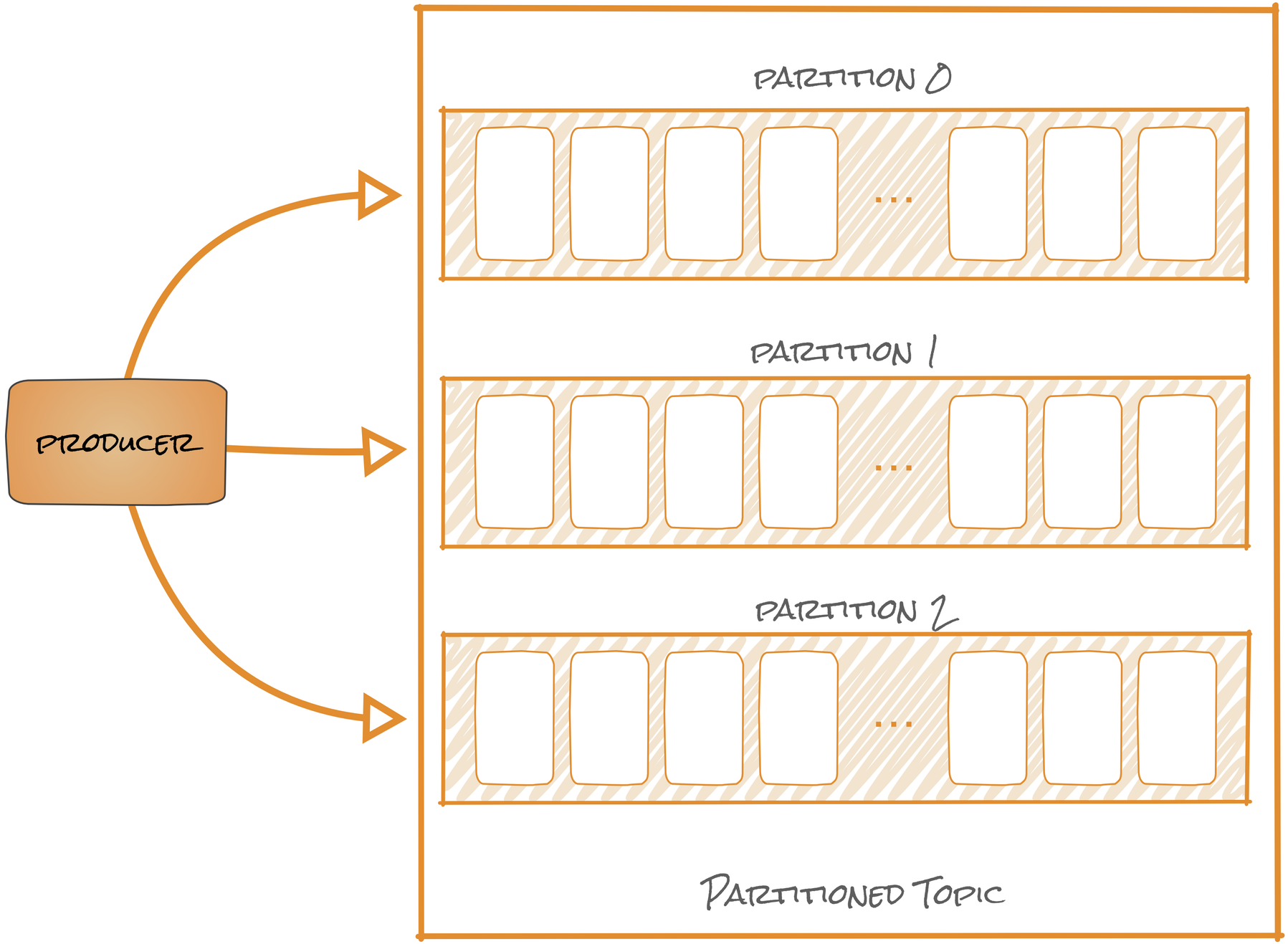
PUB / SUB

@rmoff

Producing data



Messages are added at the end of the log


PRODUCER
PARTITION 0
PARTITION 1
PARTITION 2
PARTITIONED TOPIC

```go
package main

import (
    "gopkg.in/confluentinc/confluent-kafka-go.v1/kafka"
)

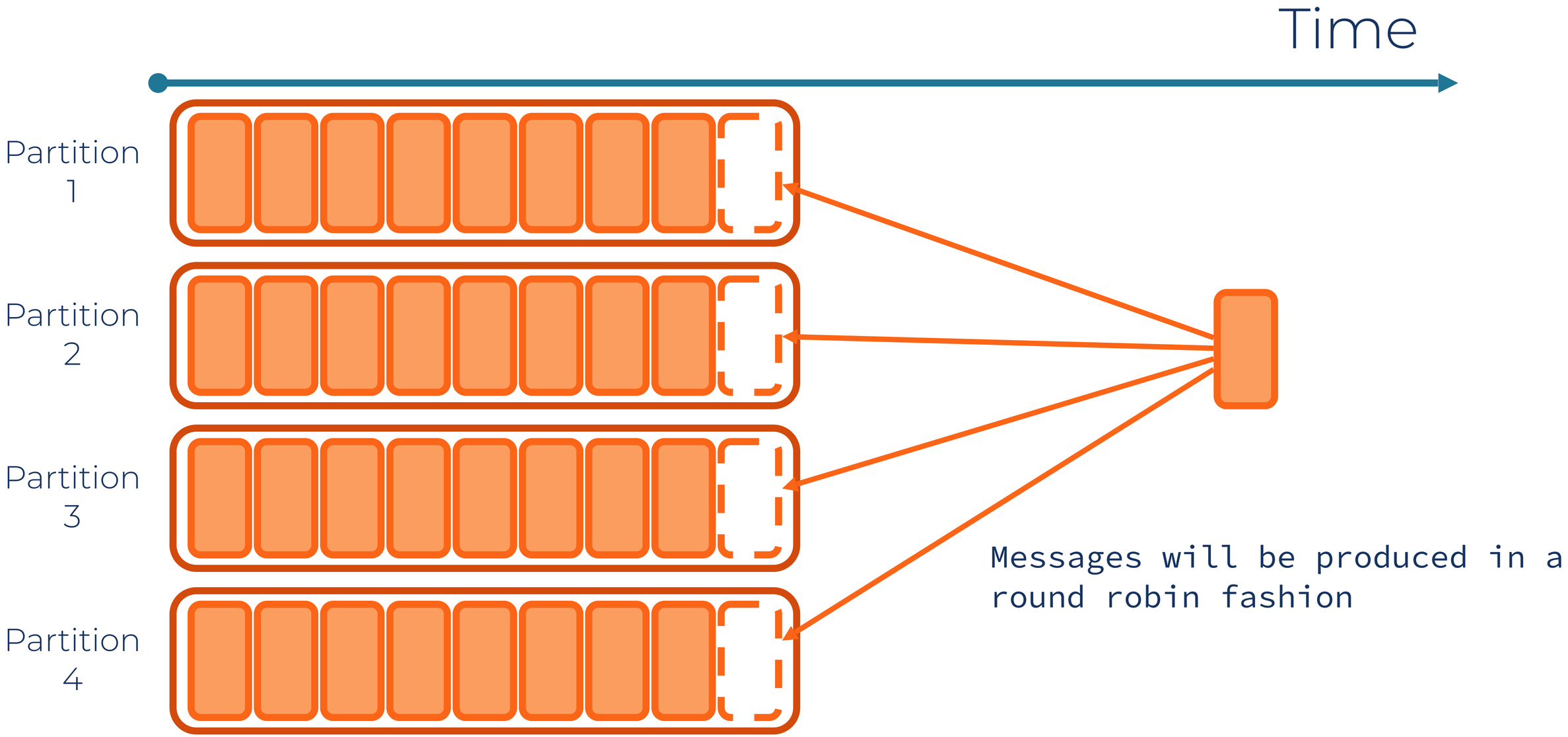
func main() {

    topic := "test_topic"
    p, _ := kafka.NewProducer(&kafka.ConfigMap{
        "bootstrap.servers": "localhost:9092"})
    defer p.Close()

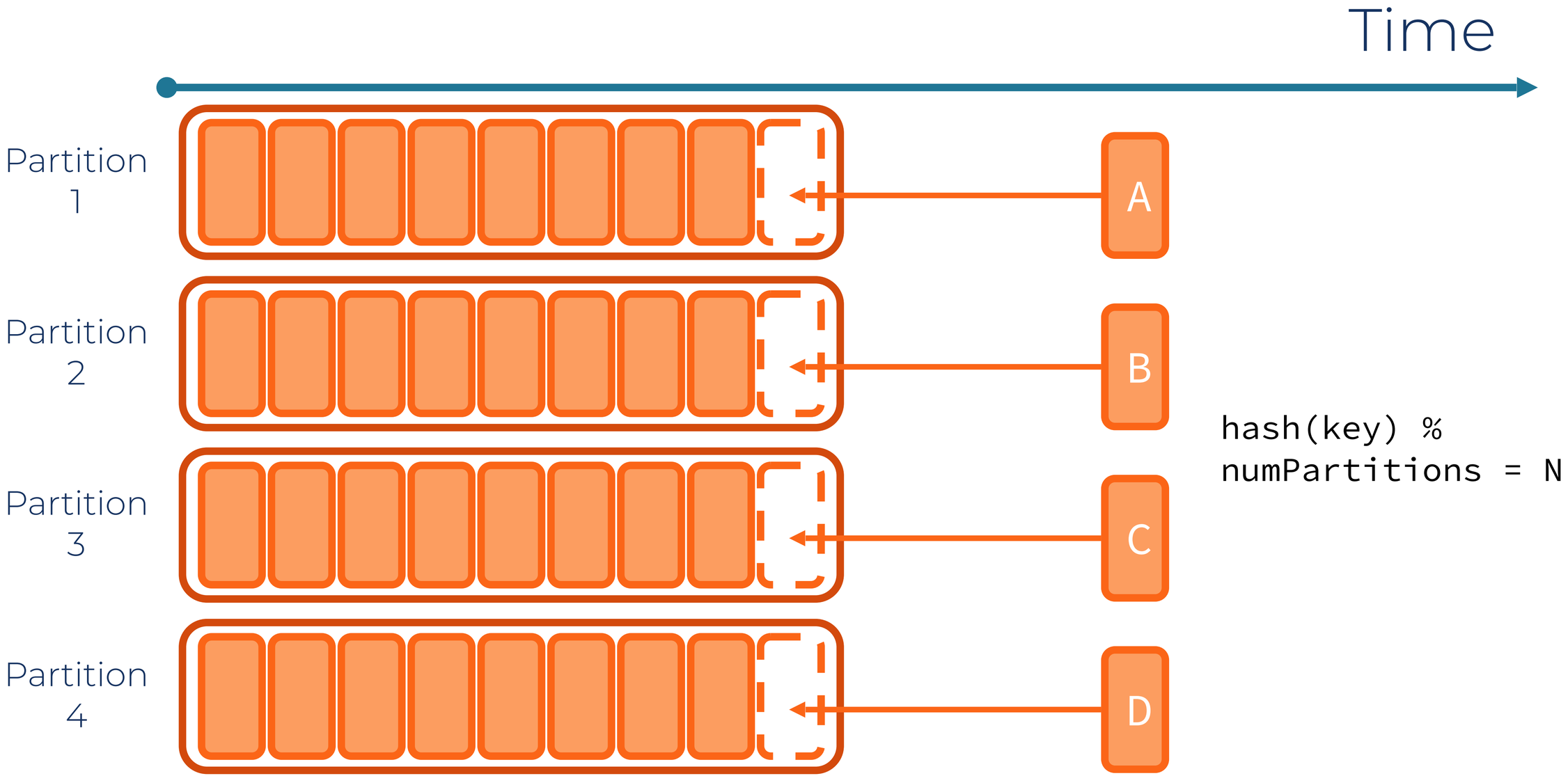
    p.Produce(&kafka.Message{
        TopicPartition: kafka.TopicPartition{Topic: &topic,
            Partition: 0},
        Value: []byte("Hello world")}, nil)

}
```

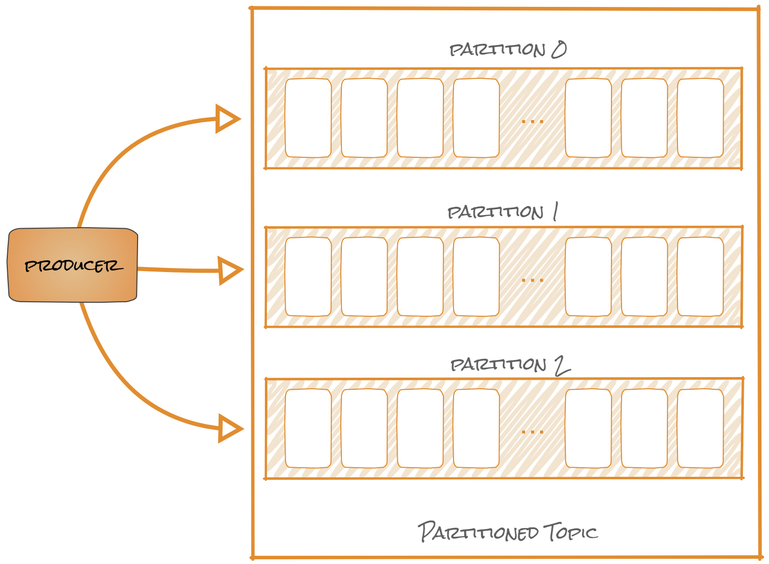
Producing to Kafka - No Key



Producing to Kafka - With Key



Producers



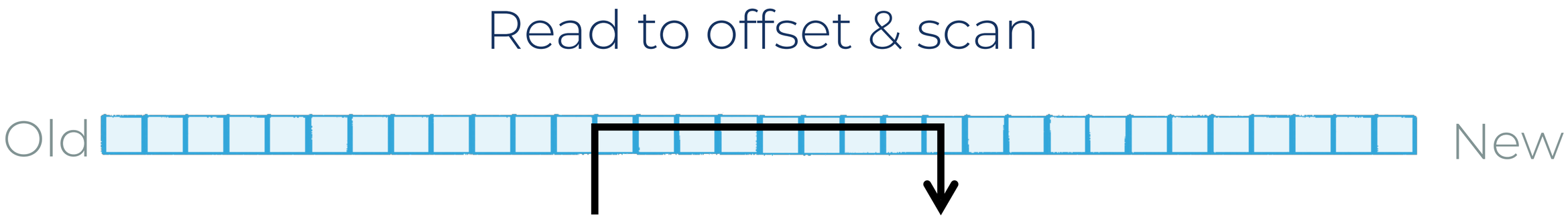
- A client application
- Puts messages into topics
- Handles partitioning, network protocol
- Java, Go, .NET, C/C++, Python
- Also every other language



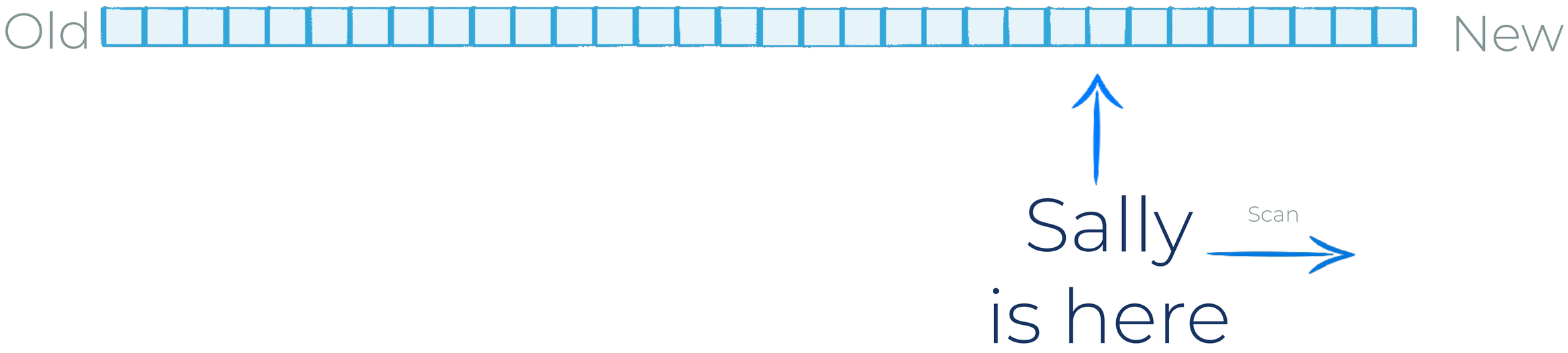
PUB / SUB

@rmoff

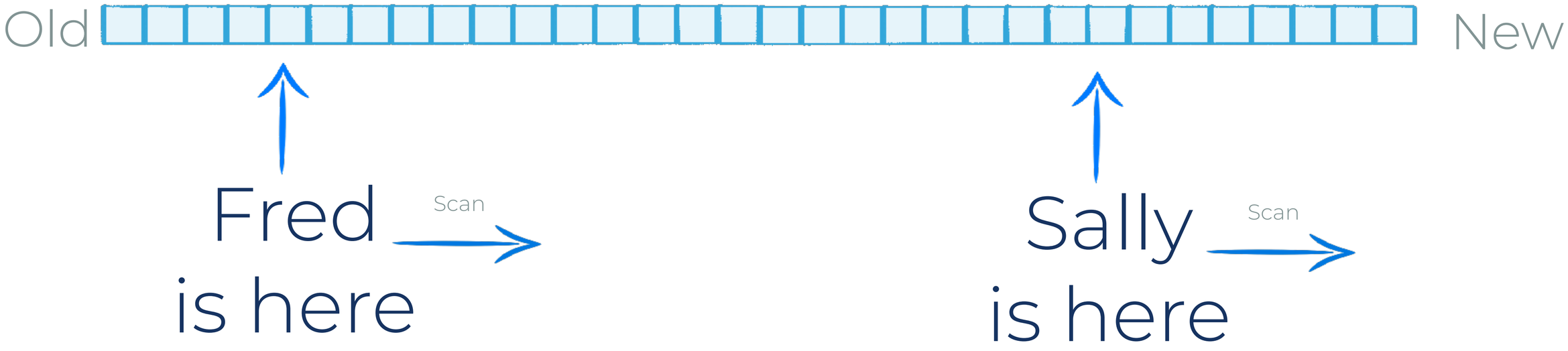
Consuming data - access is only sequential



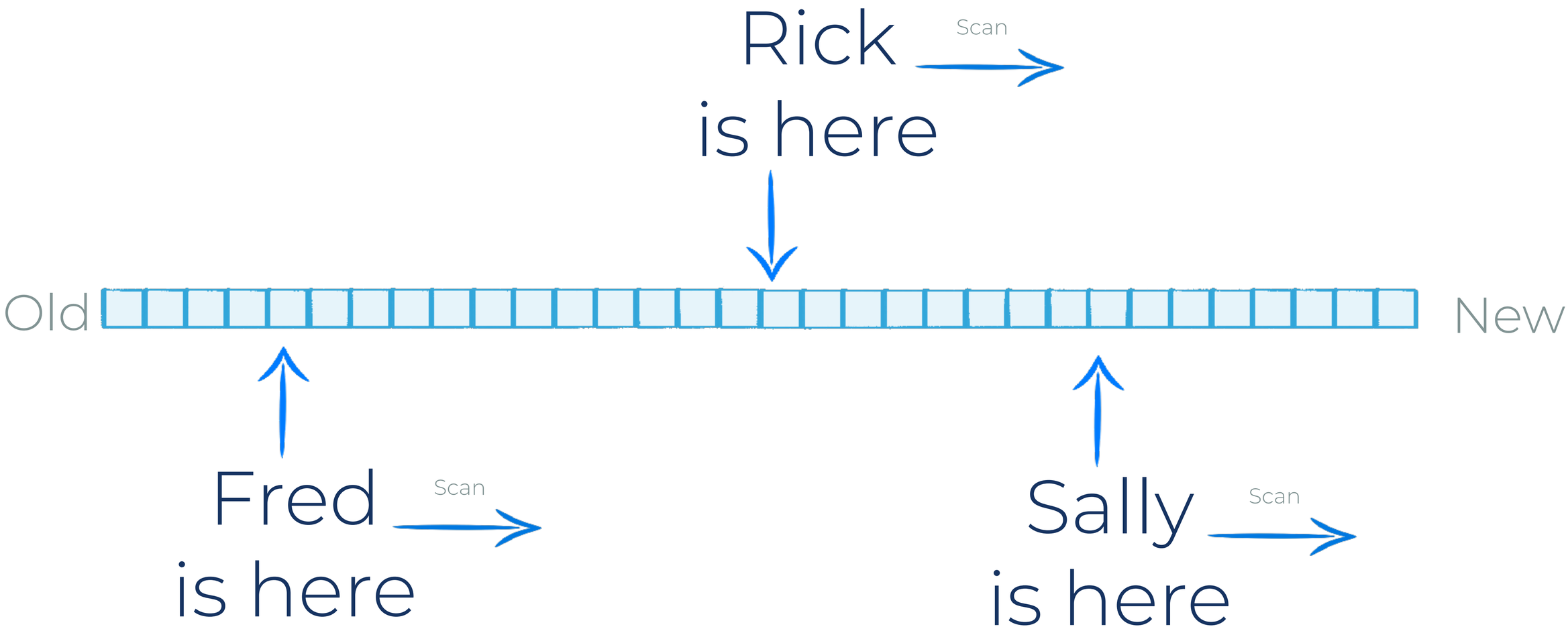
Consumers have a position of their own



Consumers have a position of their own



Consumers have a position of their own

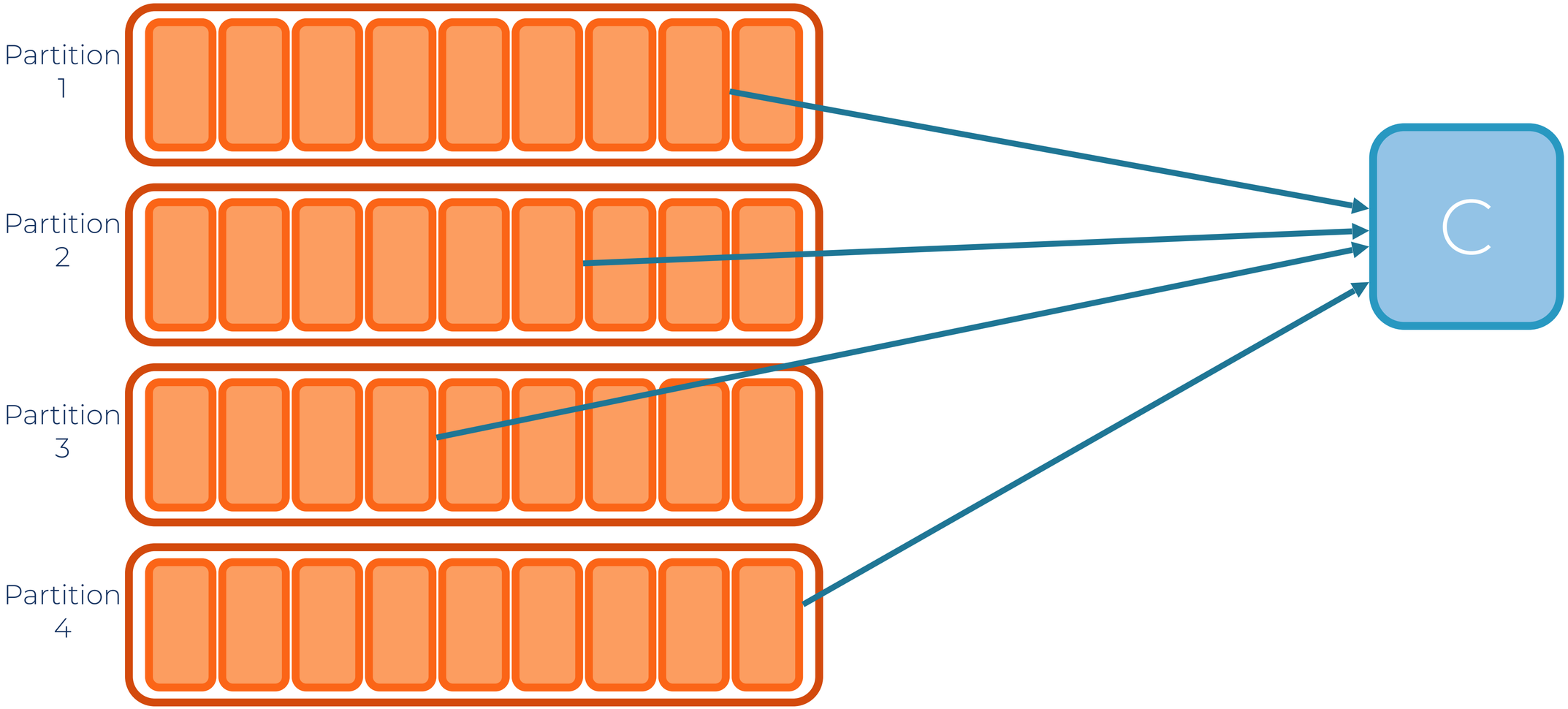


```
c, _ := kafka.NewConsumer(&cm)
defer c.Close()
c.Subscribe(topic, nil)

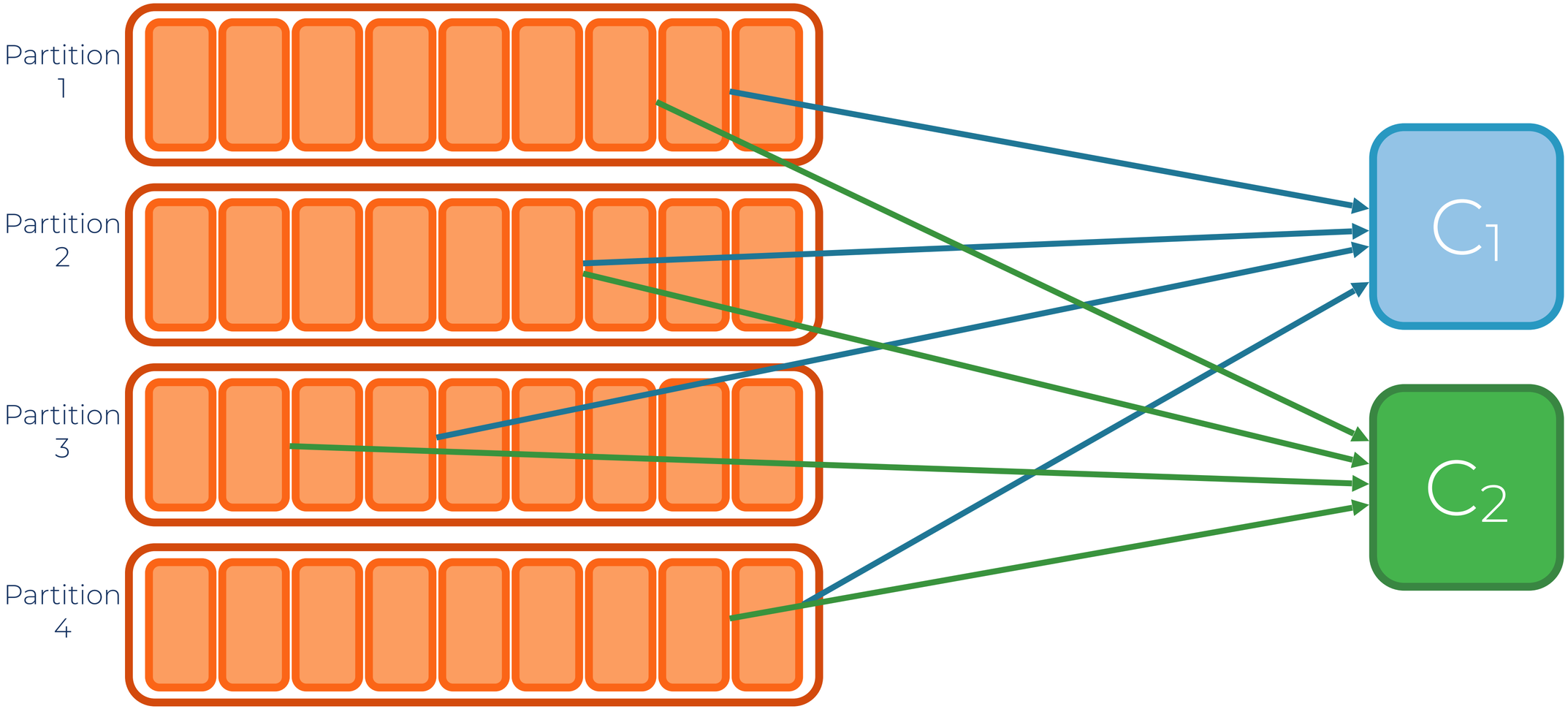
for {
    select {
    case ev := <-c.Events():
        switch ev.(type) {

        case *kafka.Message:
            km := ev.(*kafka.Message)
            fmt.Printf("✅ Message '%v' received from topic
'%v'\n", string(km.Value), string(*km.TopicPartition.Topic))
        }
    }
}
```

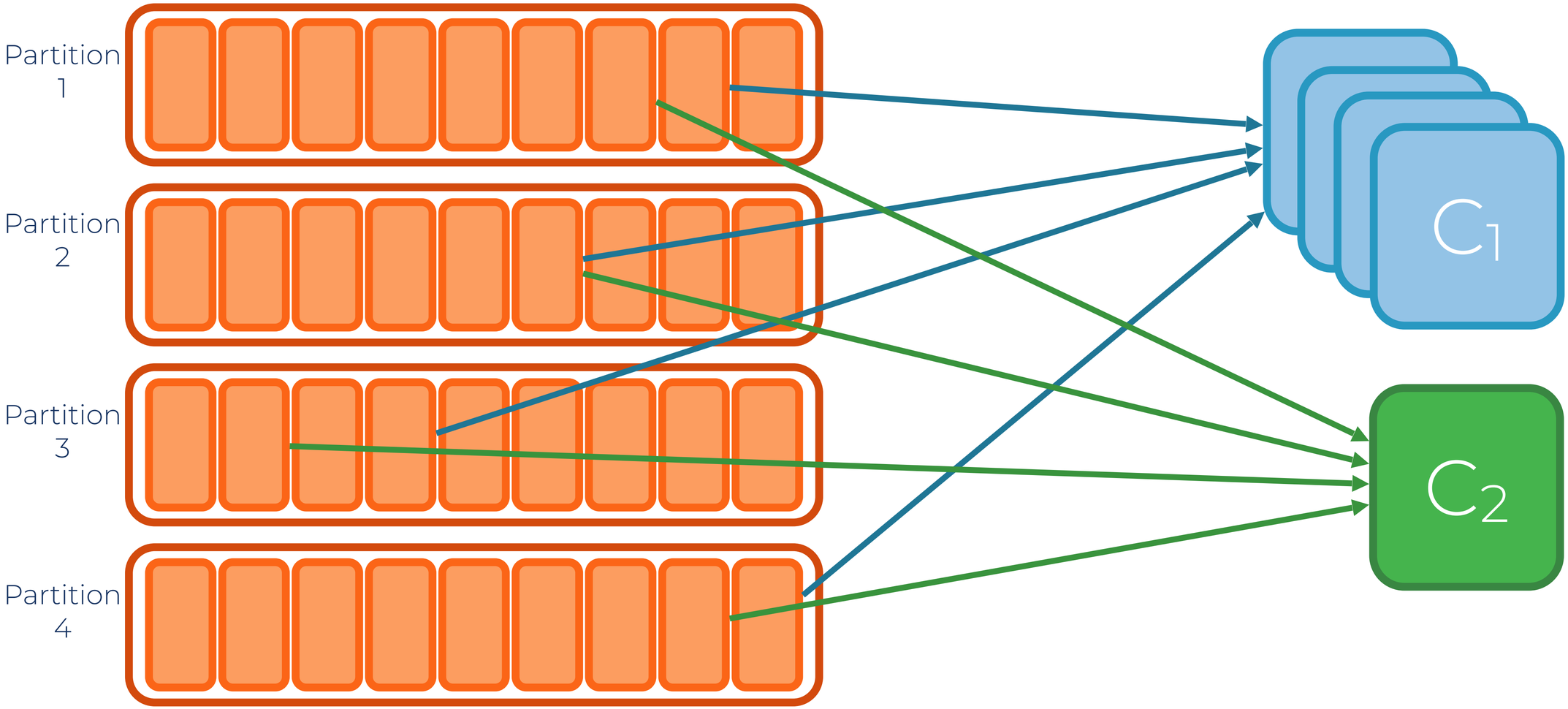
Consuming From Kafka - Single Consumer



Consuming From Kafka - Multiple Consumers

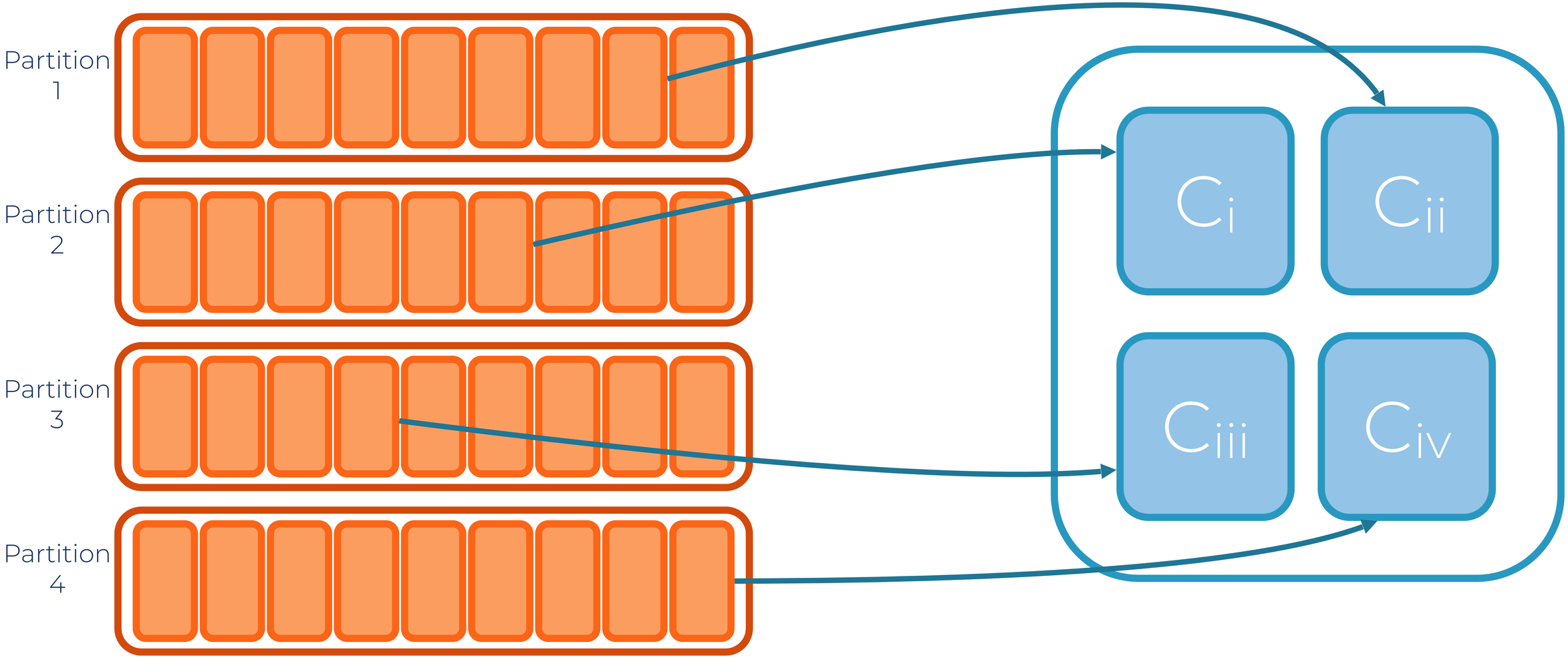


Consuming From Kafka - Grouped Consumers

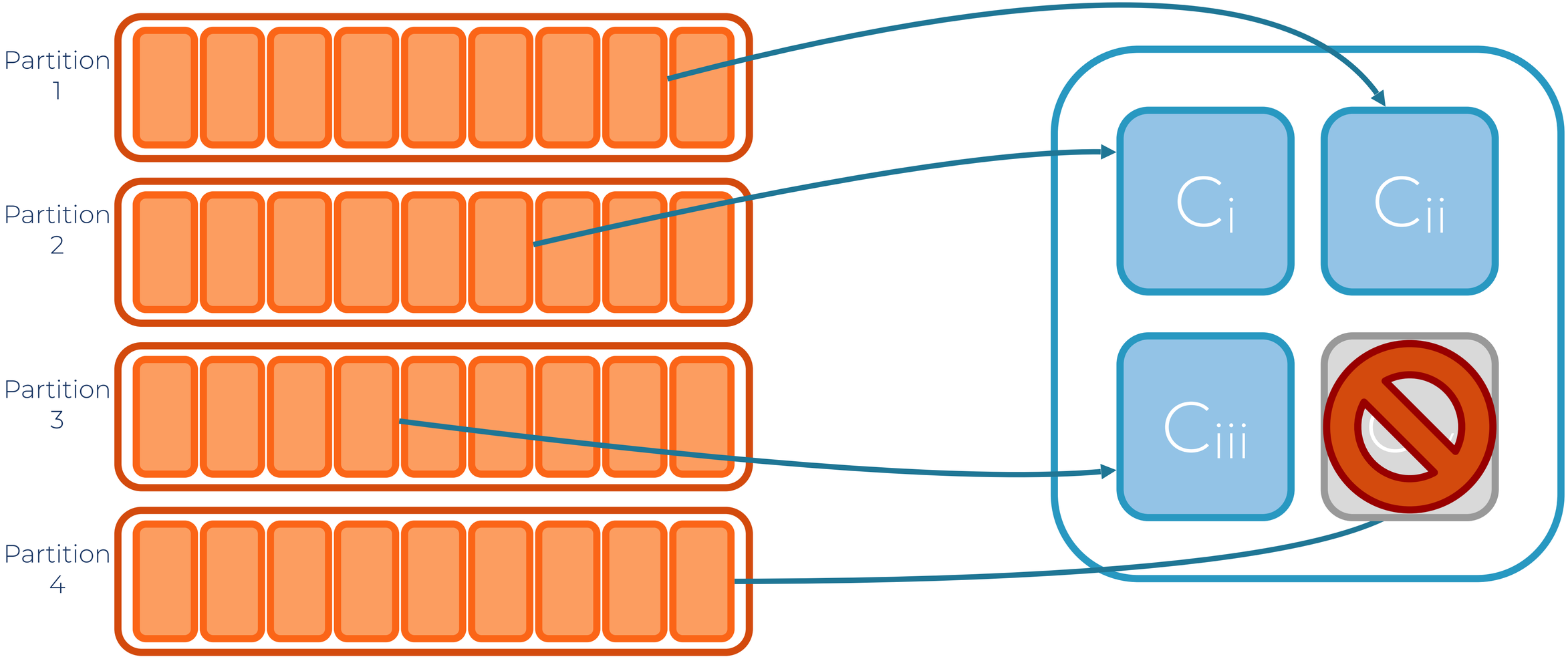



CONSUMER GROUP
COORDINATOR
CONSUMERS
CONSUMER GROUP

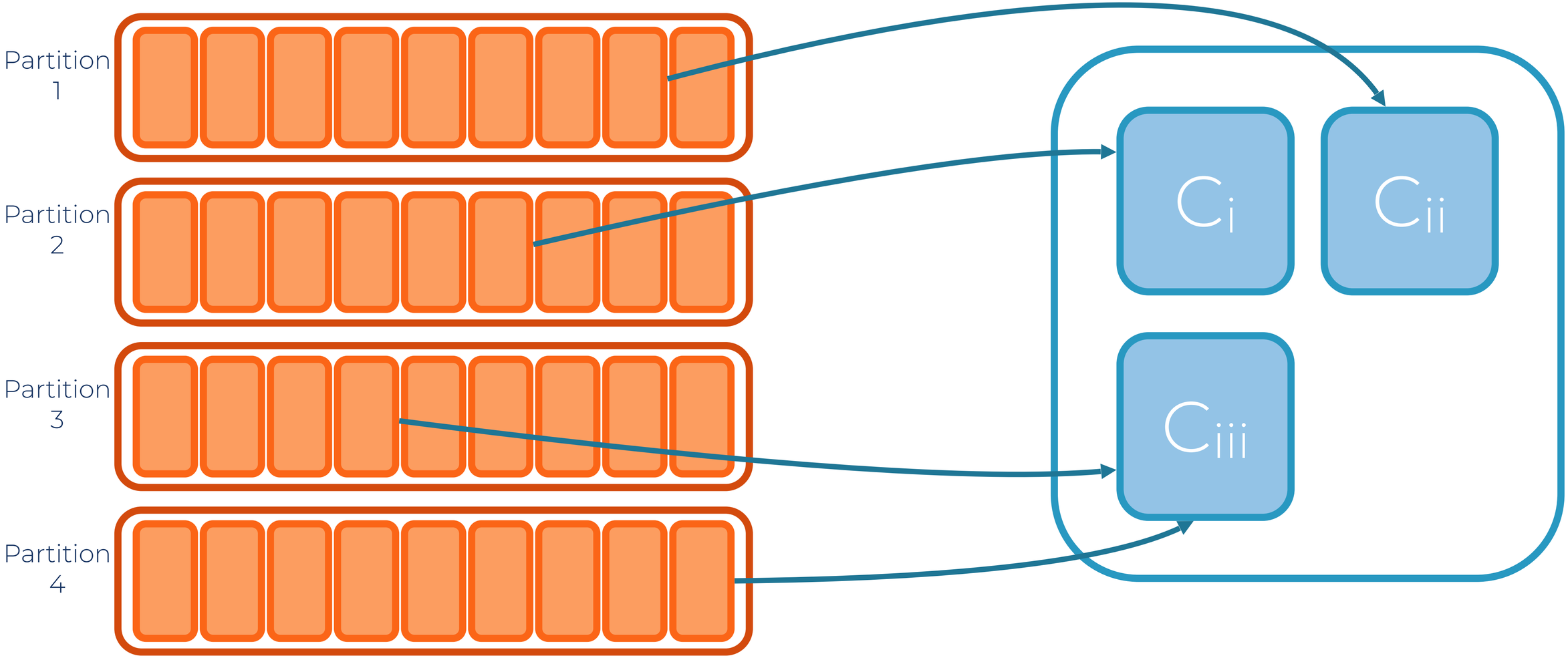
Consuming From Kafka - Grouped Consumers



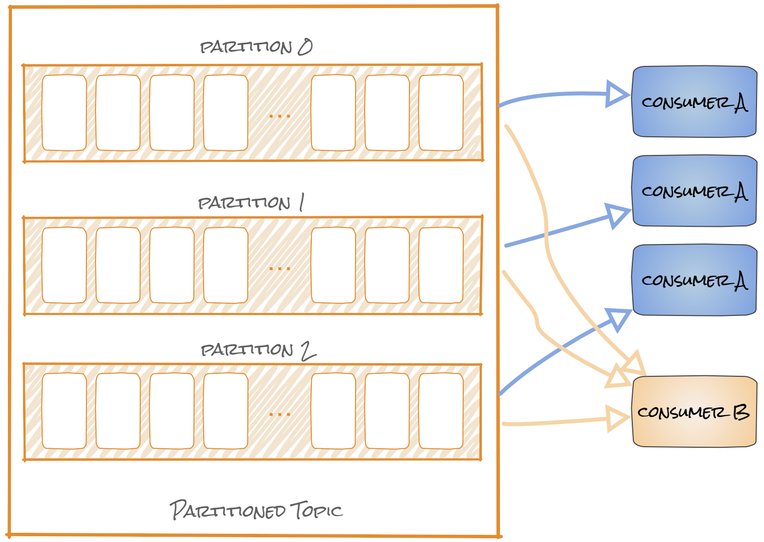
Consuming From Kafka - Grouped Consumers



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

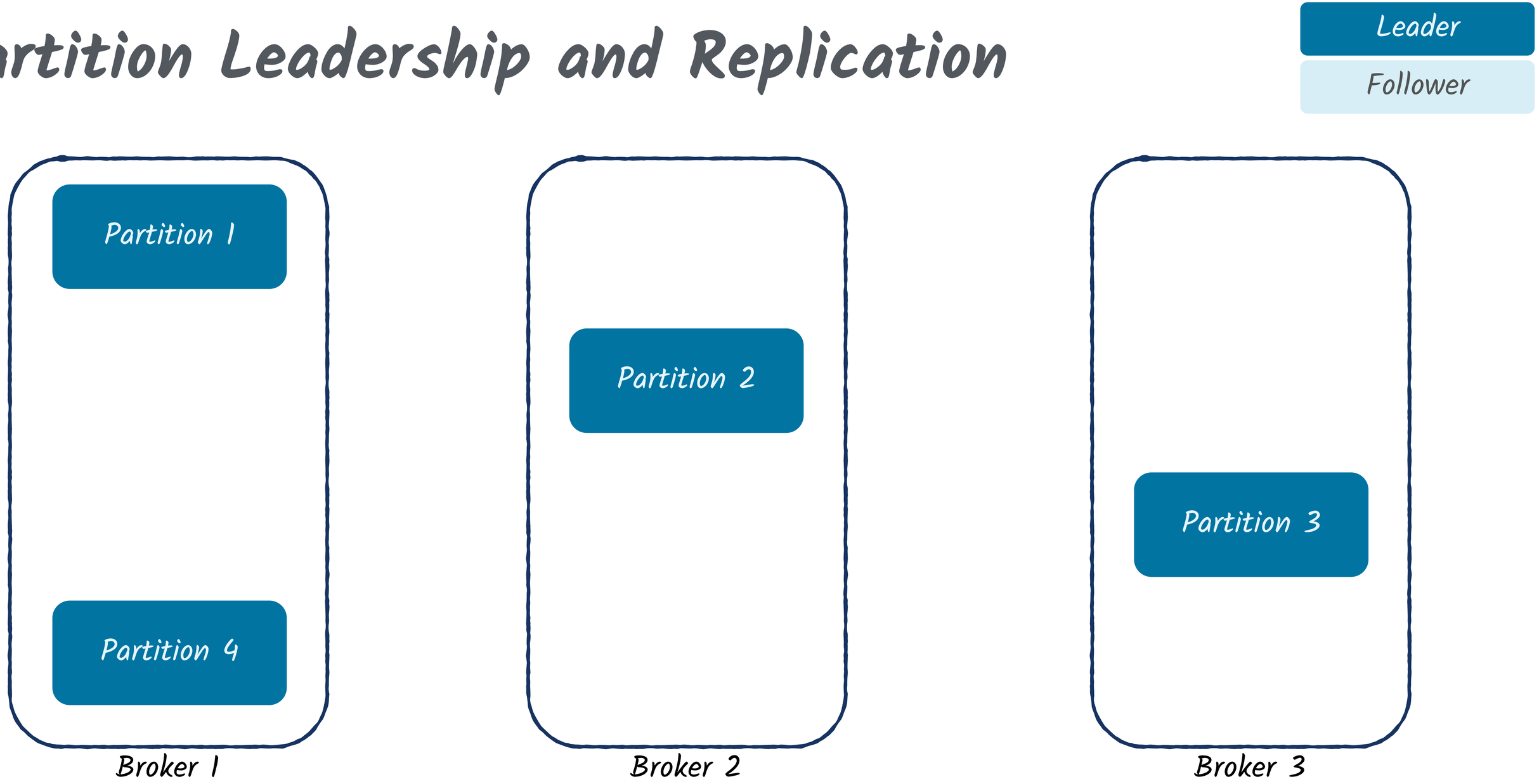


BROKERS

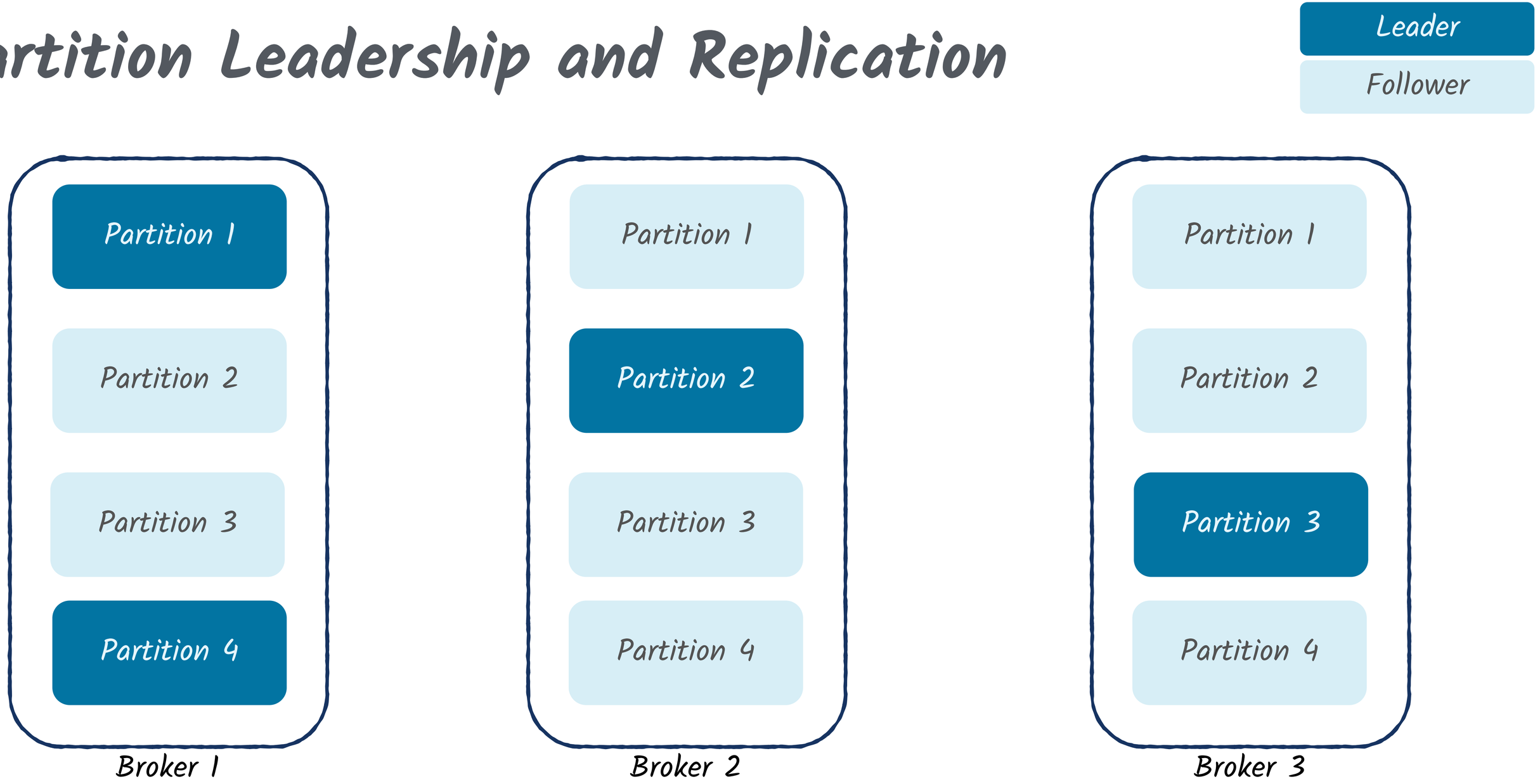
and REPLICATION

@rmoff

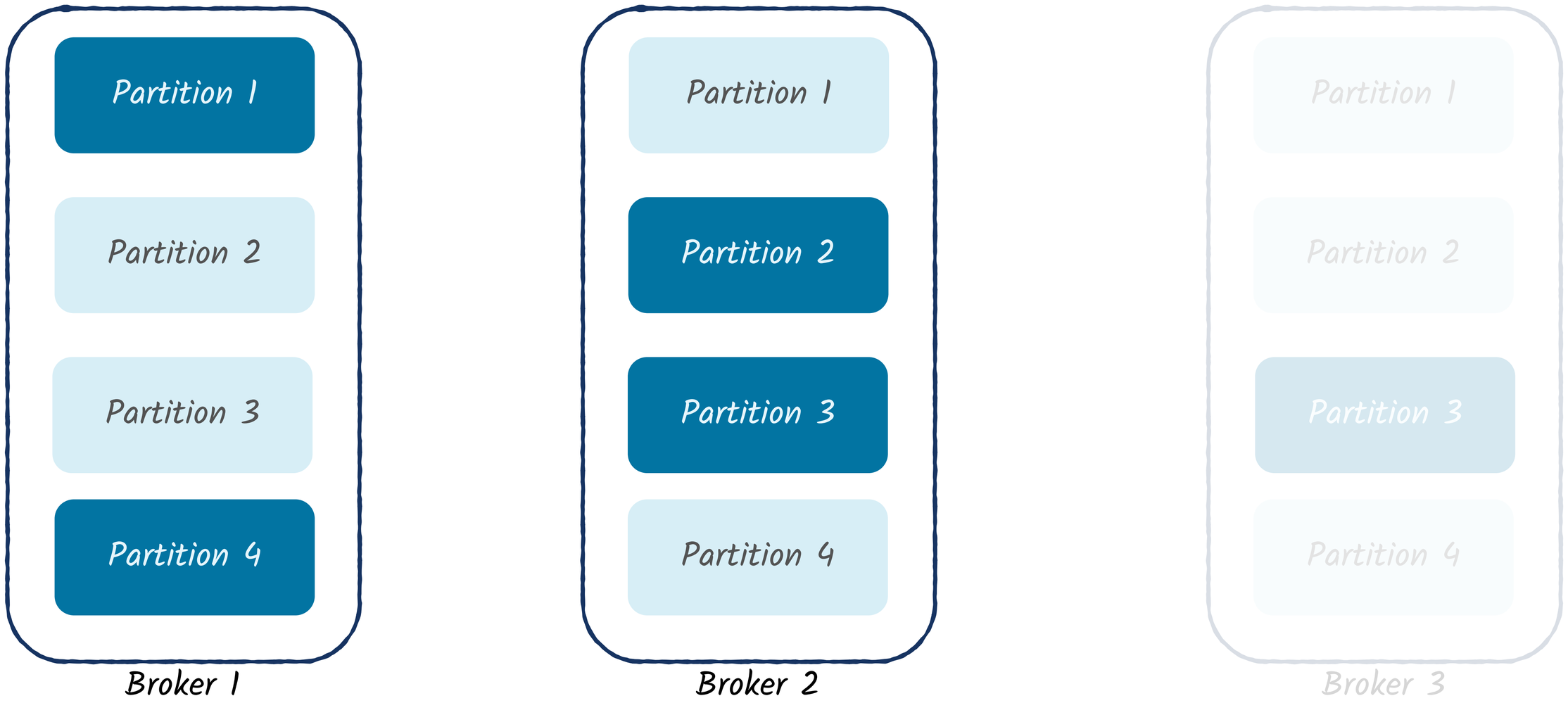
Partition Leadership and Replication



Partition Leadership and Replication



Partition Leadership and Replication





CONFLUENT

DEMO

@rmoff



So far, this is

Pretty good

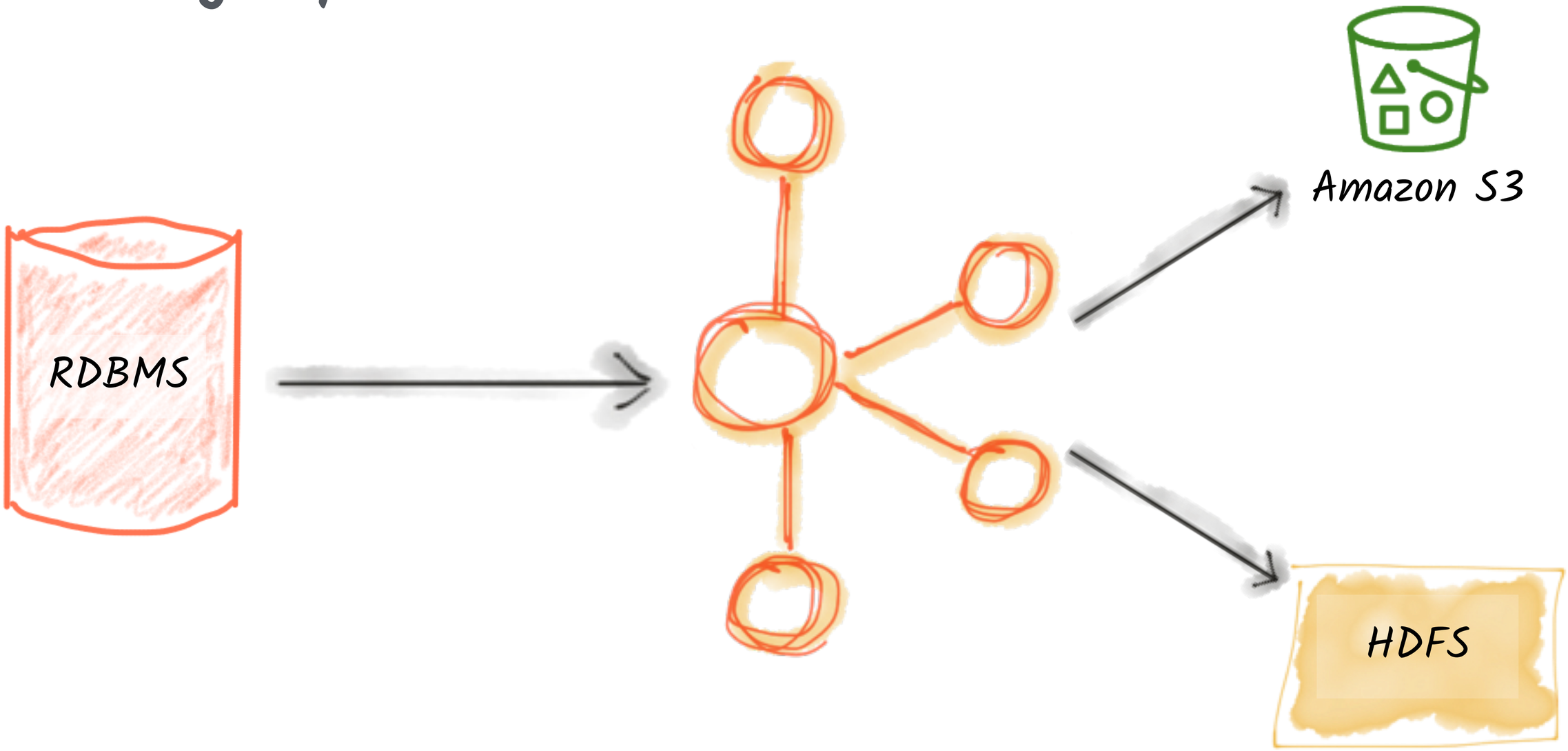


So far, this is

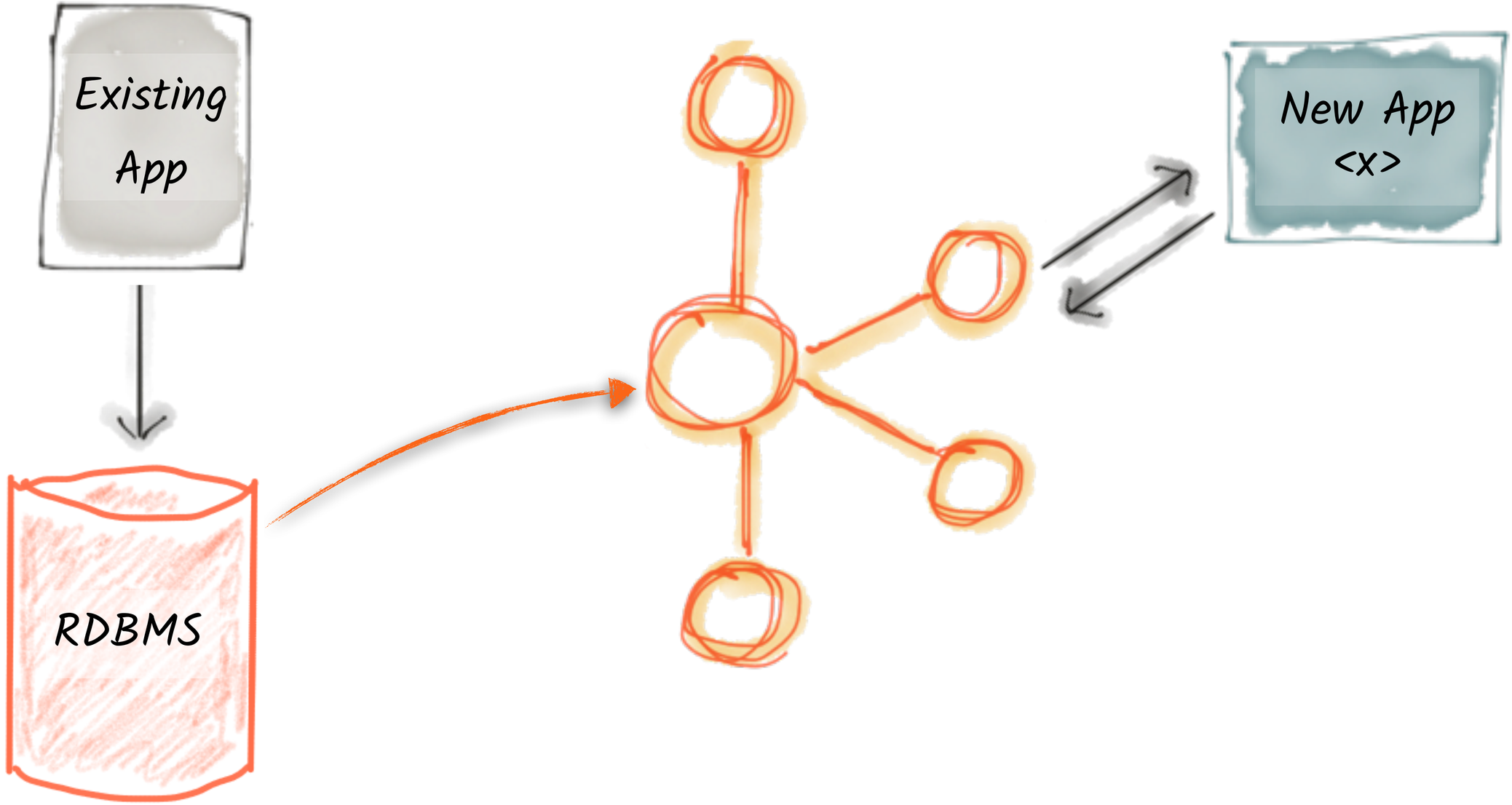
Pretty good

but I've not finished yet...

Streaming Pipelines



Evolve processing from old systems to new

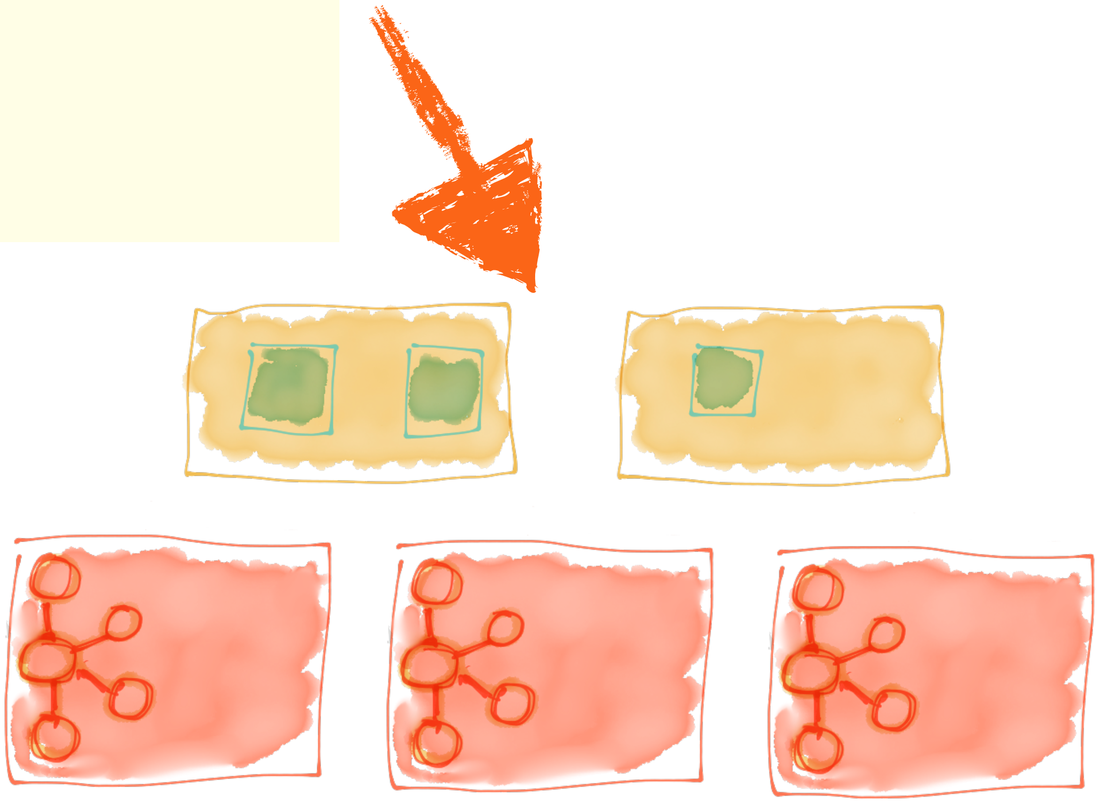




Streaming Integration with Kafka Connect



Sources



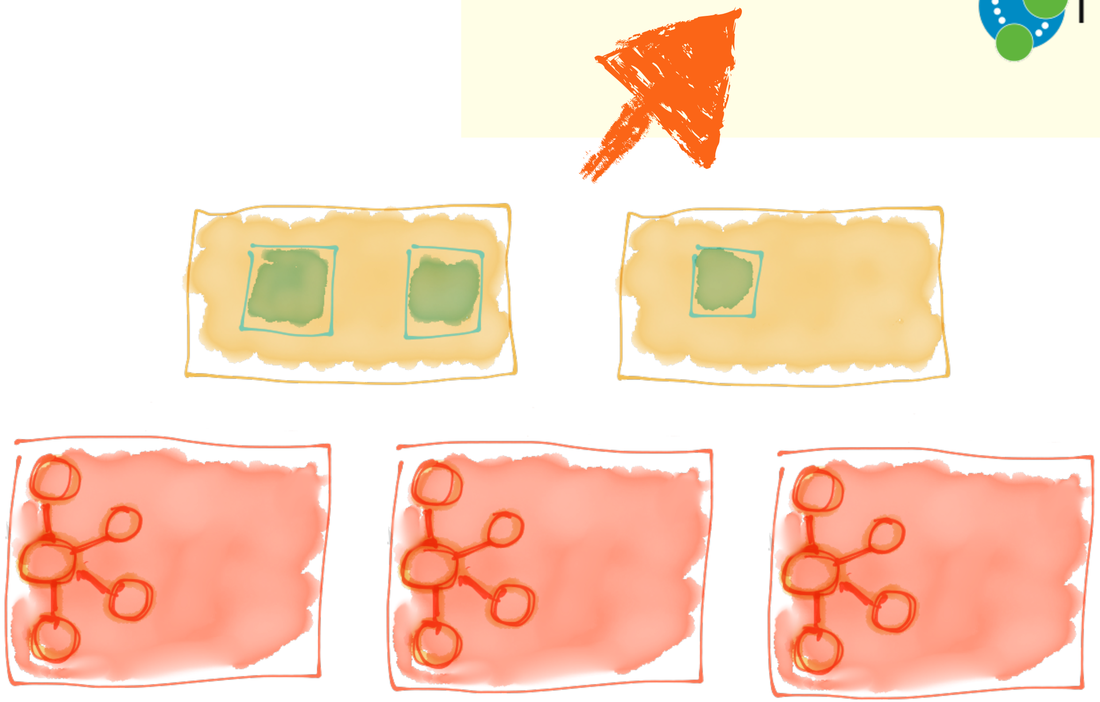
Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

Sinks





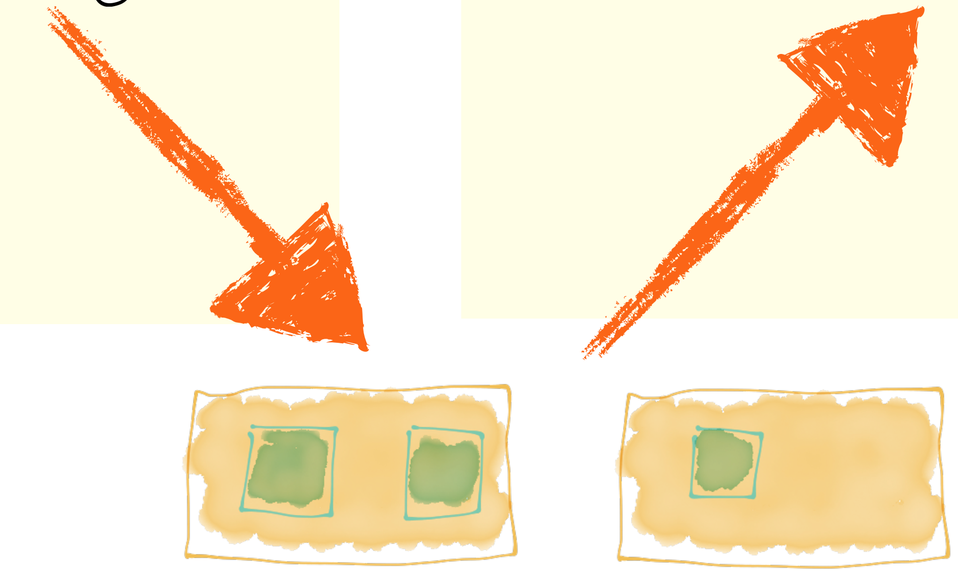
Kafka Connect

Kafka Brokers

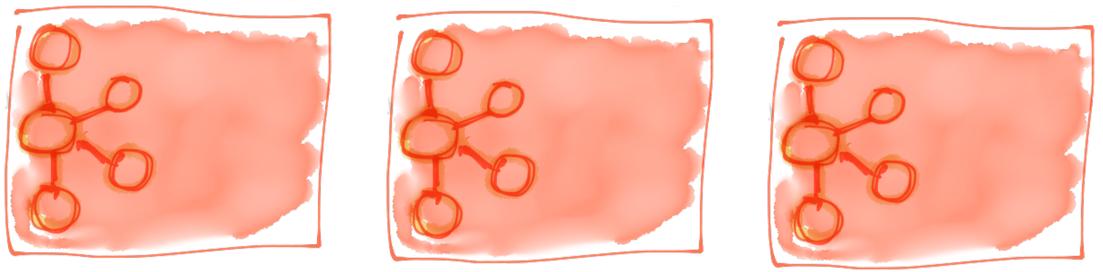
Streaming Integration with Kafka Connect







Kafka Connect

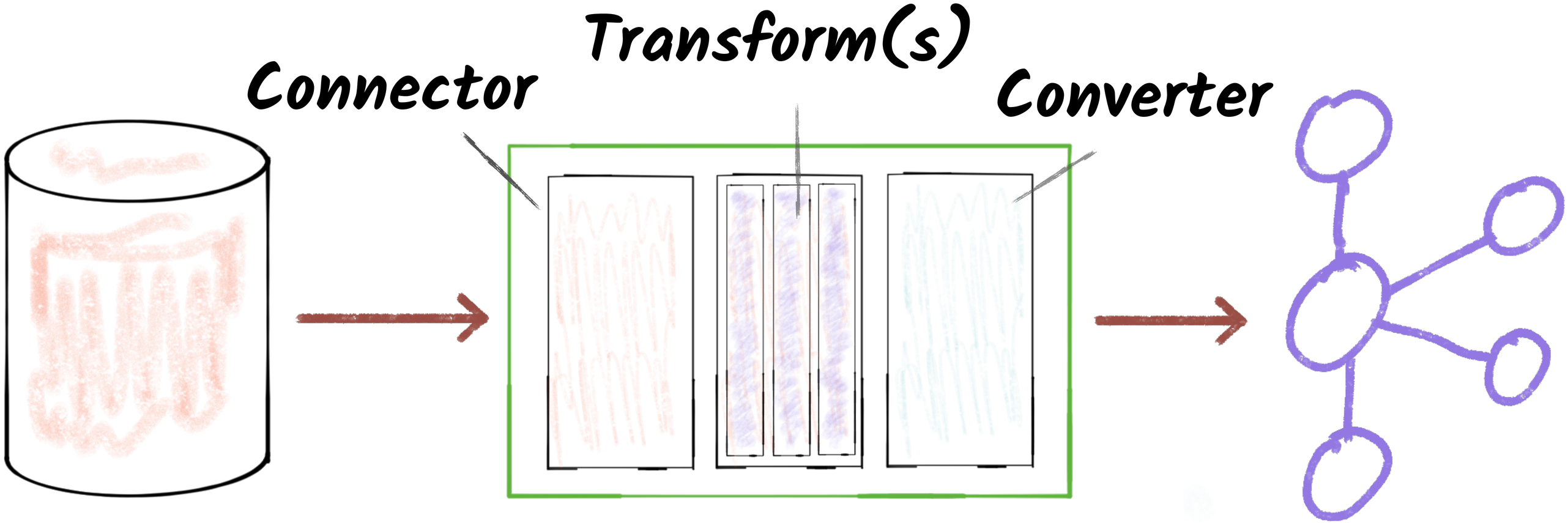


Kafka Brokers

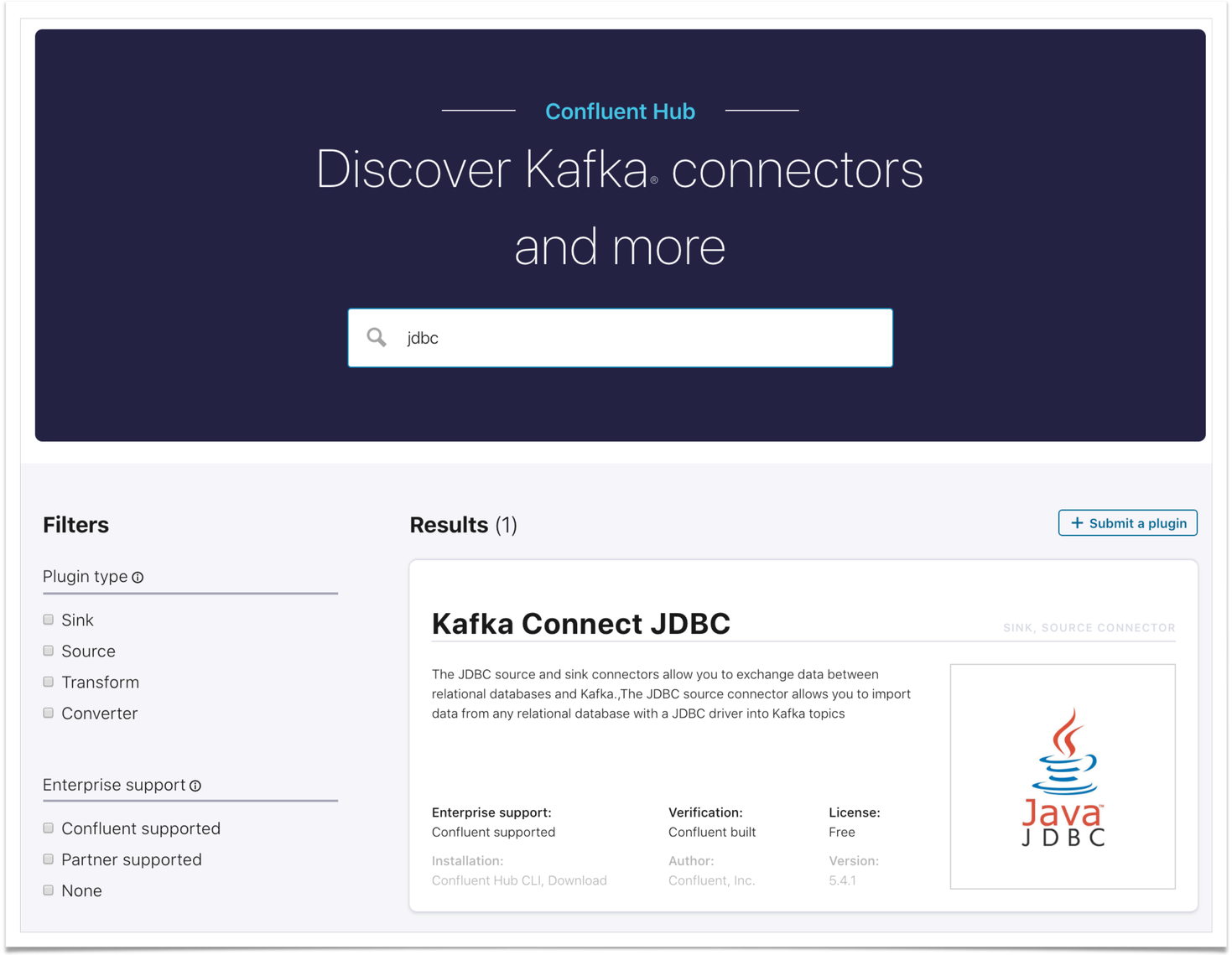
Look Ma, No Code!

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

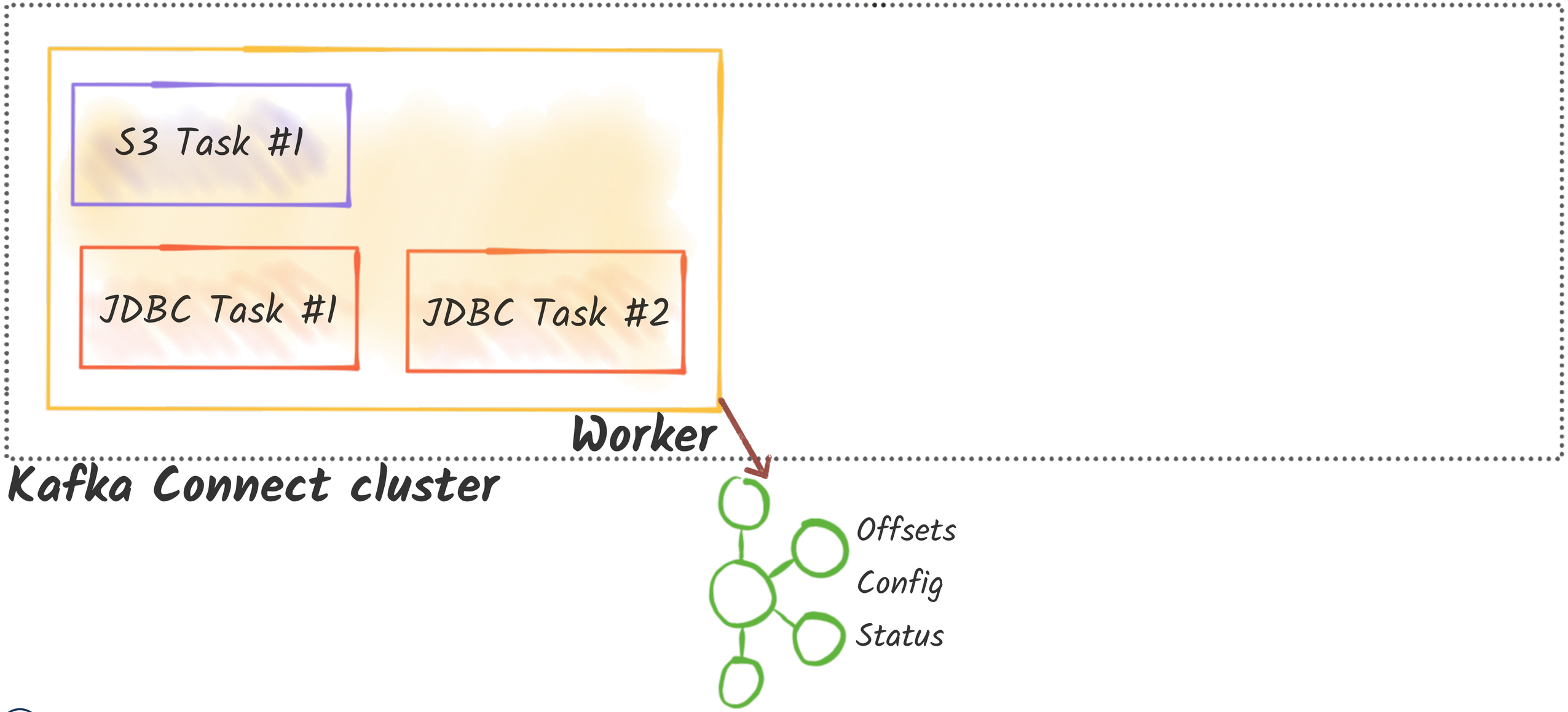
Extensible



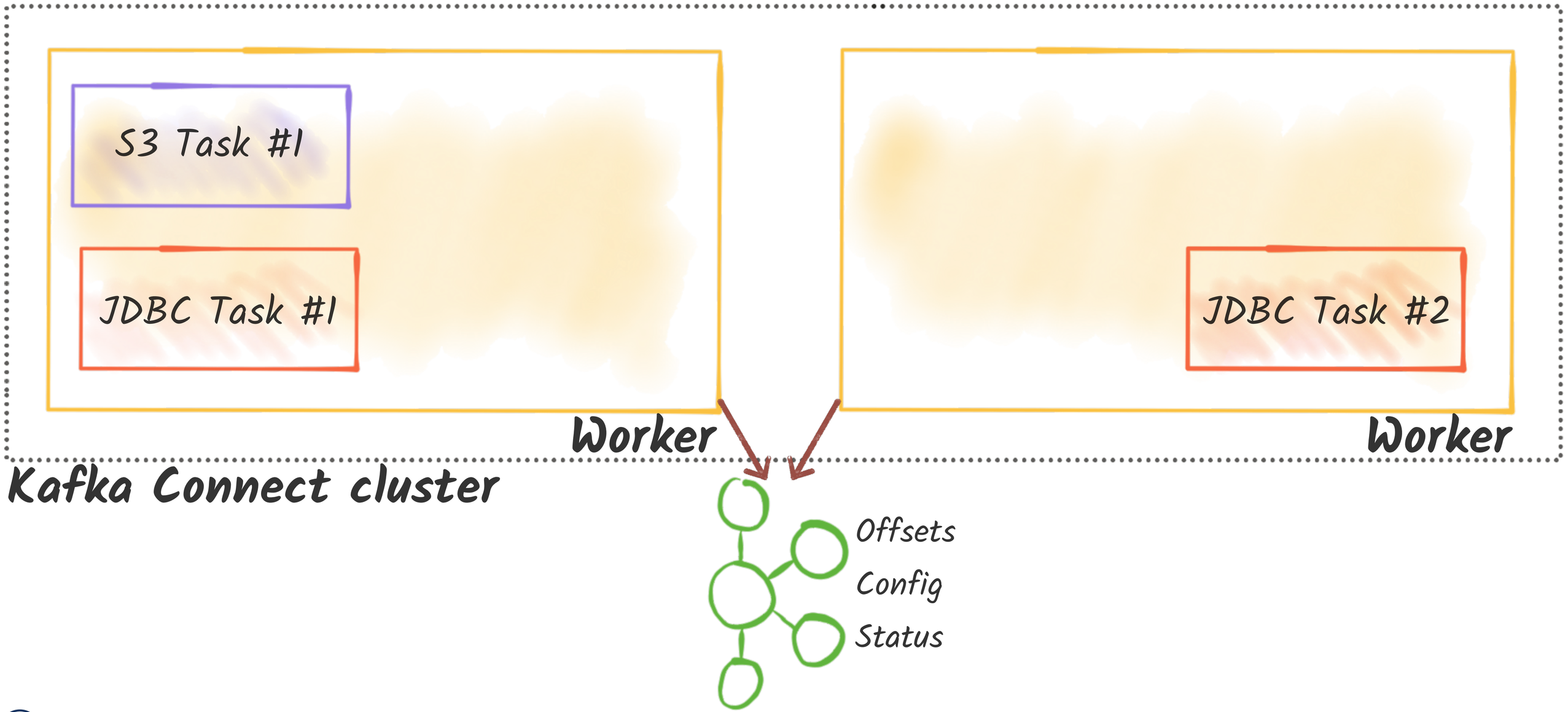
hub.confluent.io



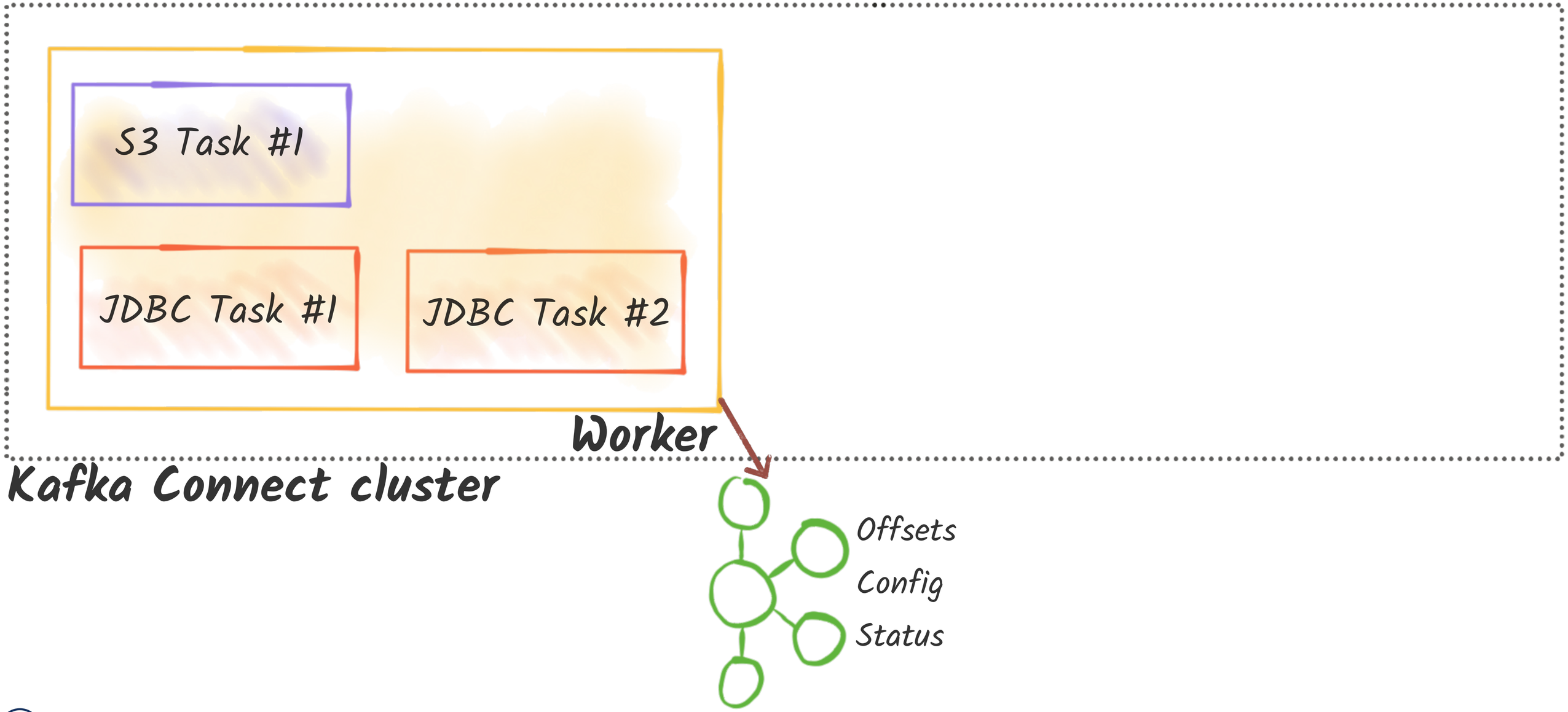
Single Kafka Connect node



Kafka Connect - scalable and fault-tolerant



Automatic fault tolerance

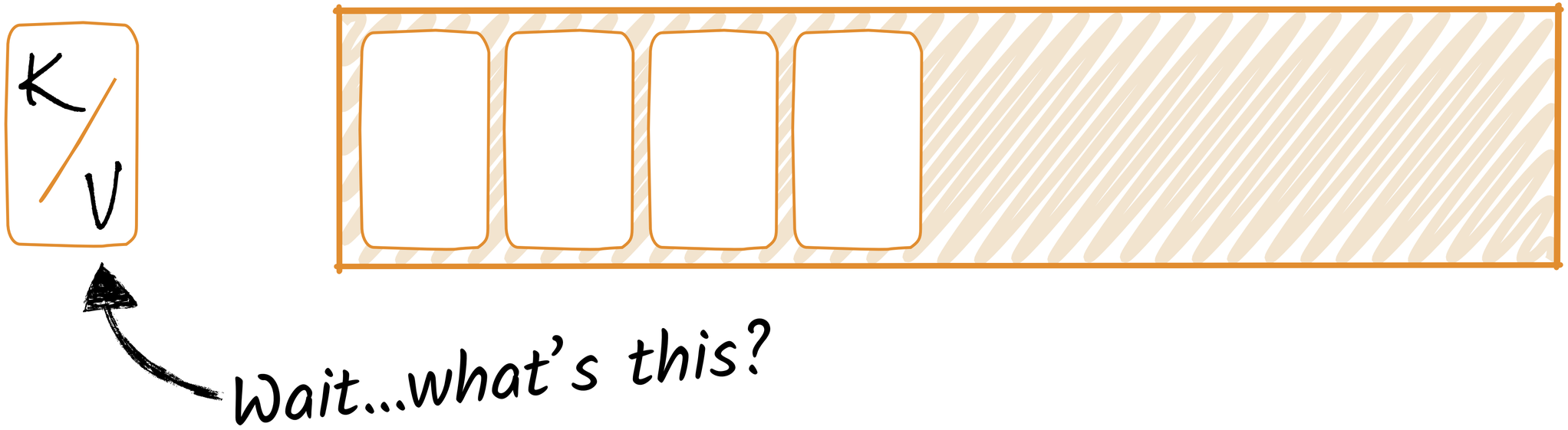



K
V


K
V


K
V

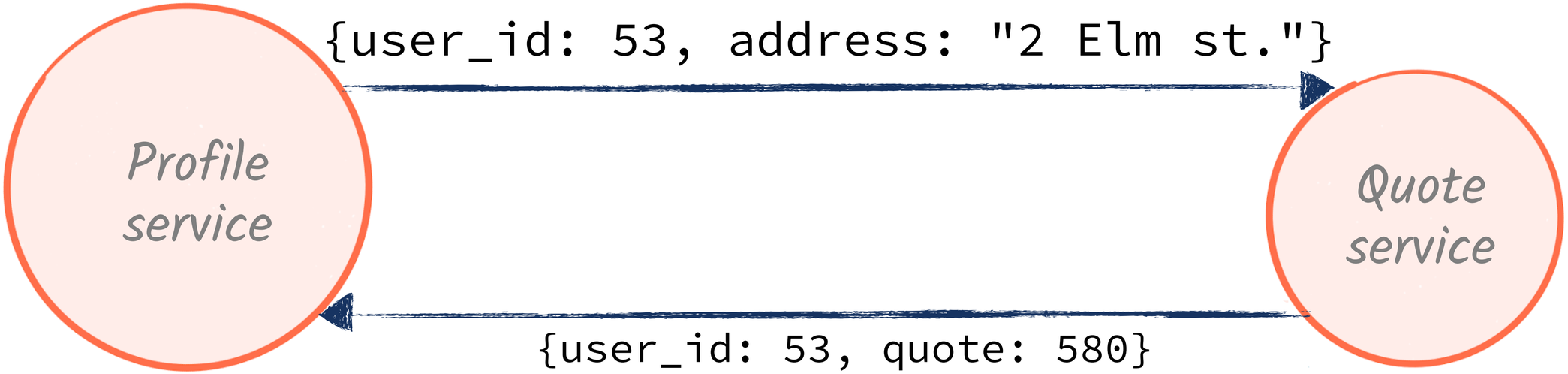

K
V


K
V
Wait...what's this?

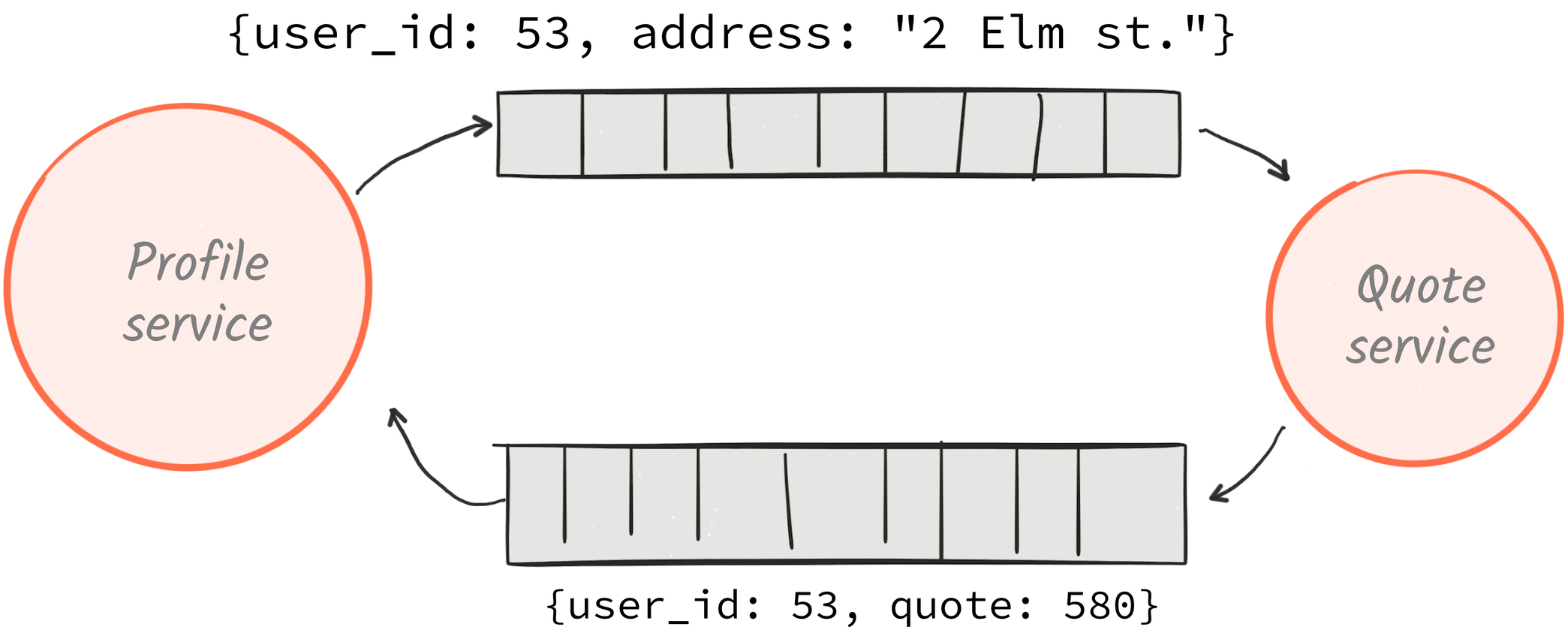
How do you serialise your data?

Avro Protobuf JSON Schema JSON CSV

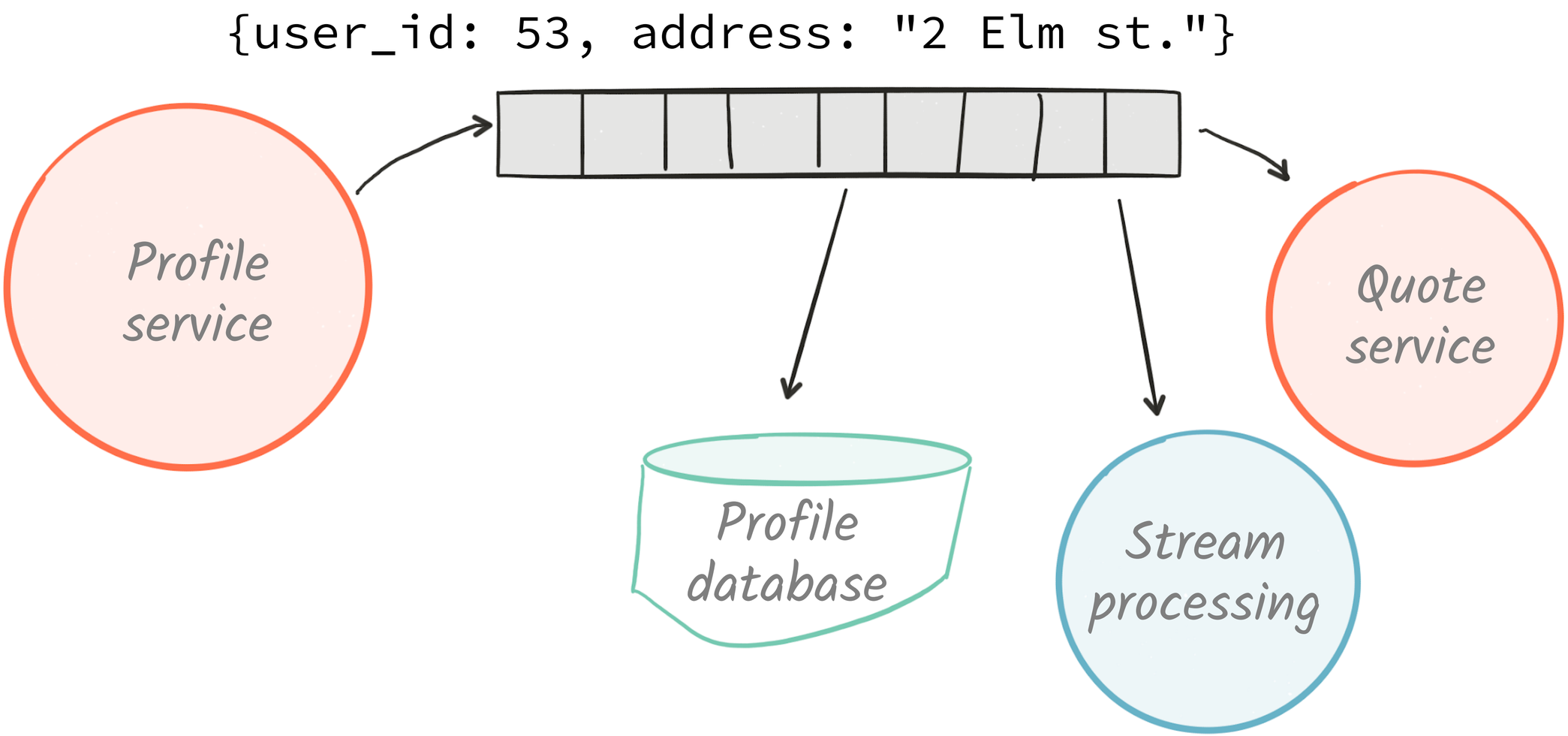
APIs are contracts between services



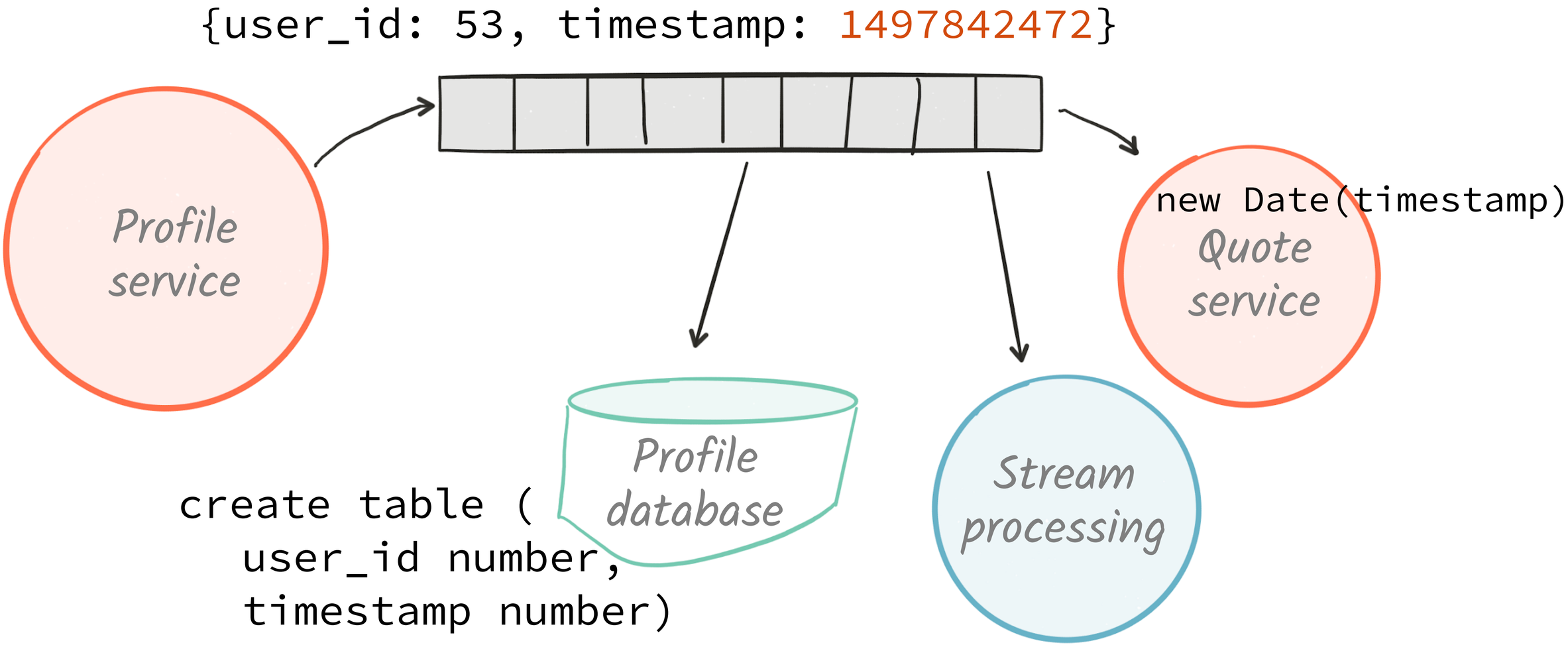
But not all services



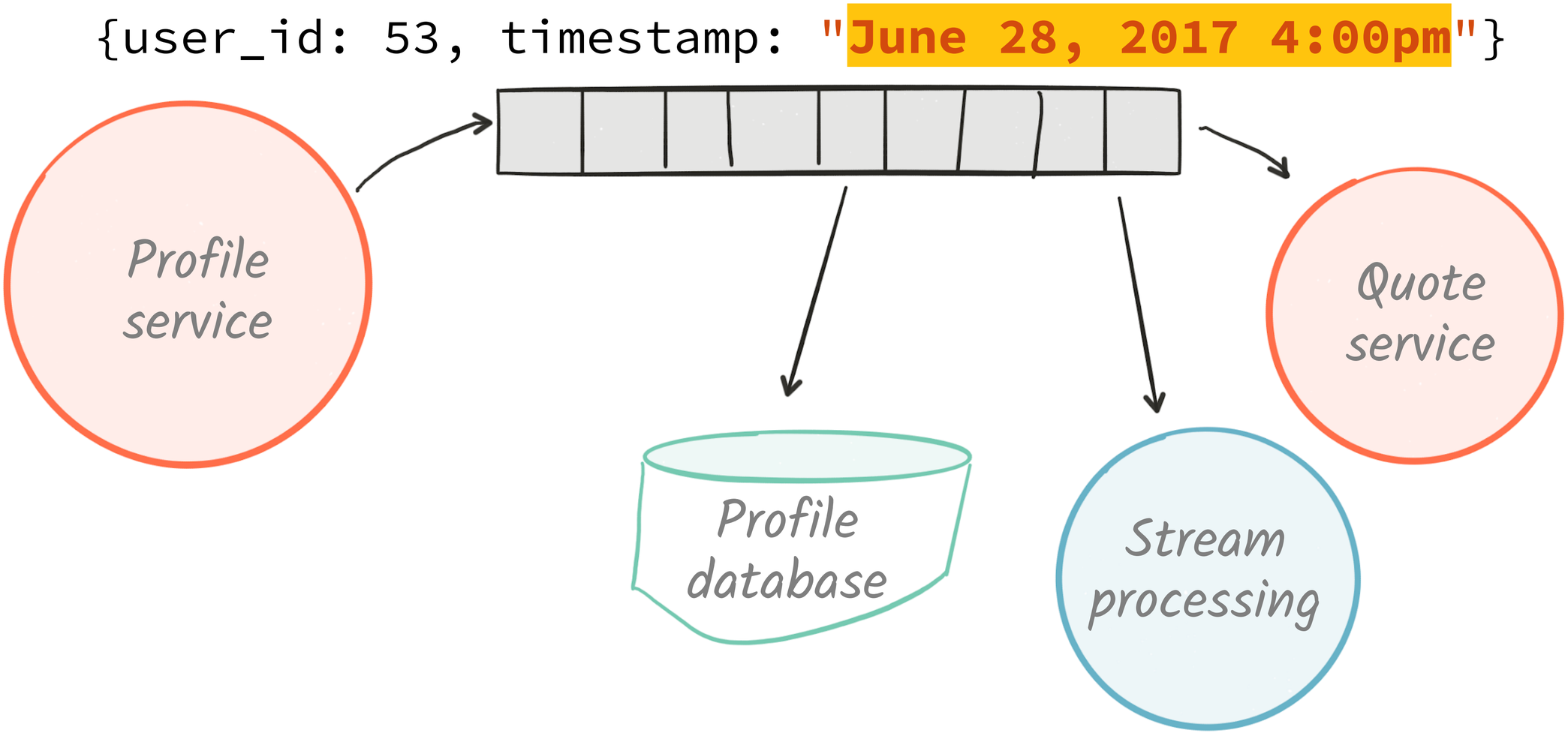
And naturally...



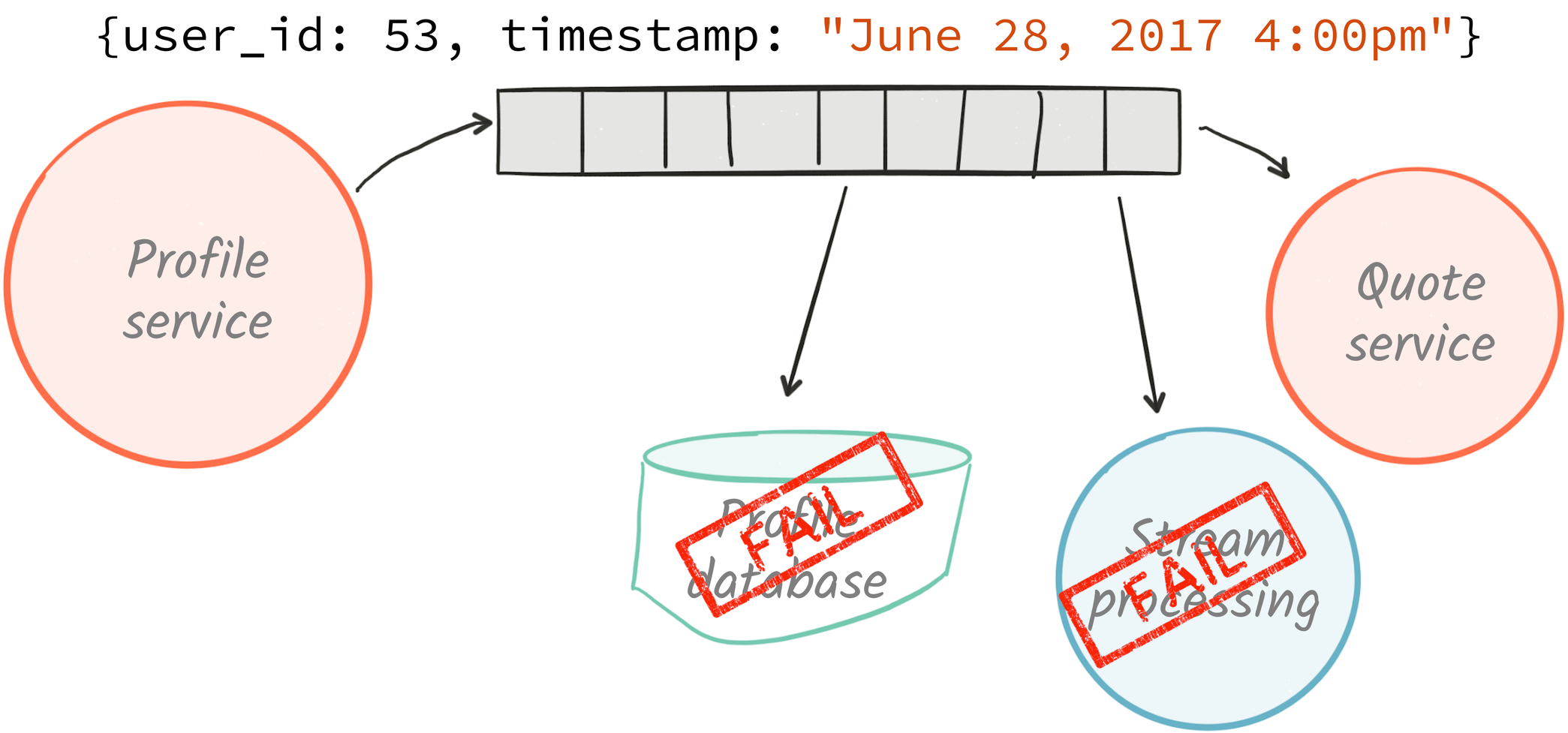
Schemas are about how teams work together



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

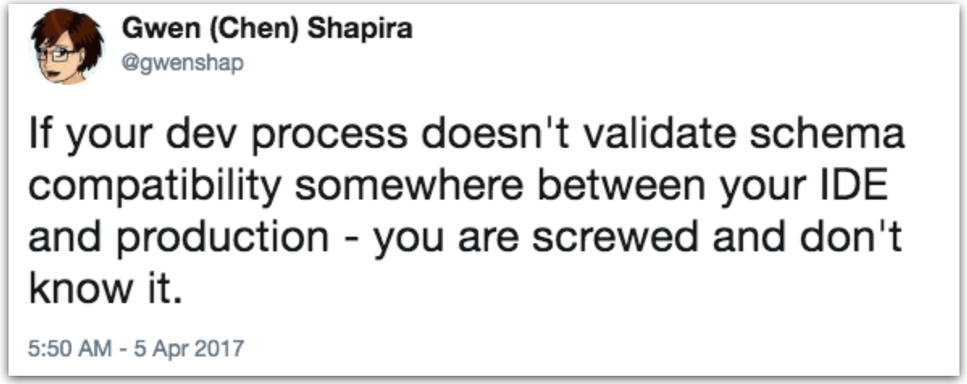
Avro 👍

Protobuf 👍

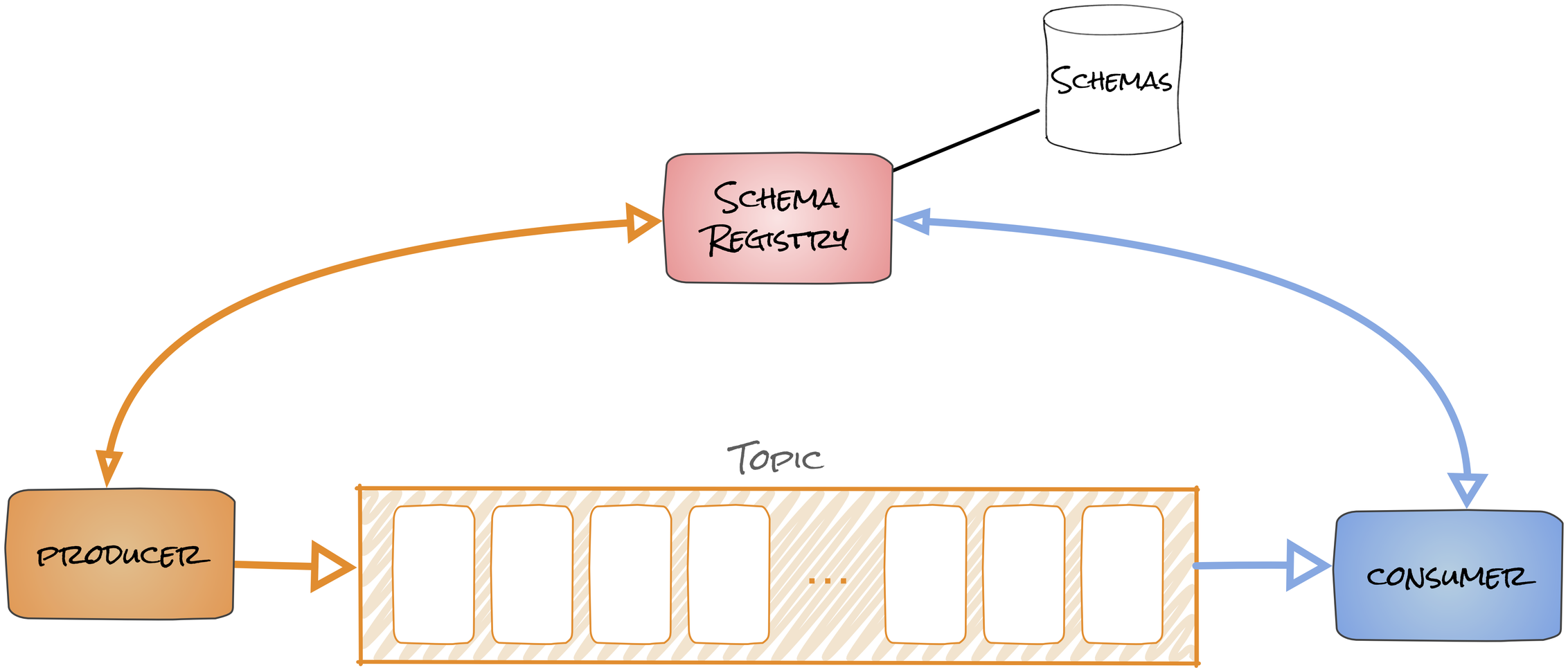
JSON Schema 👍

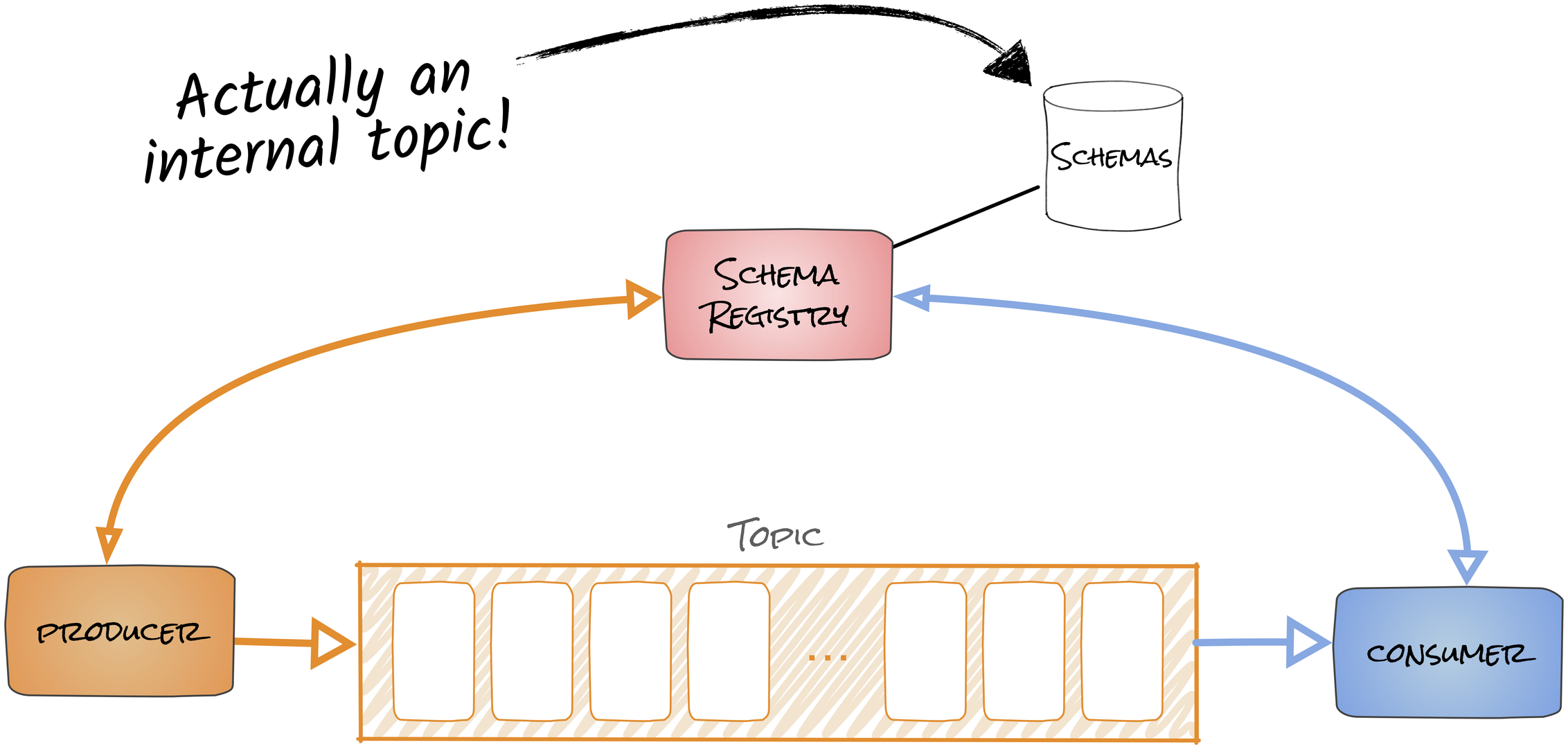
JSON 😬

CSV 



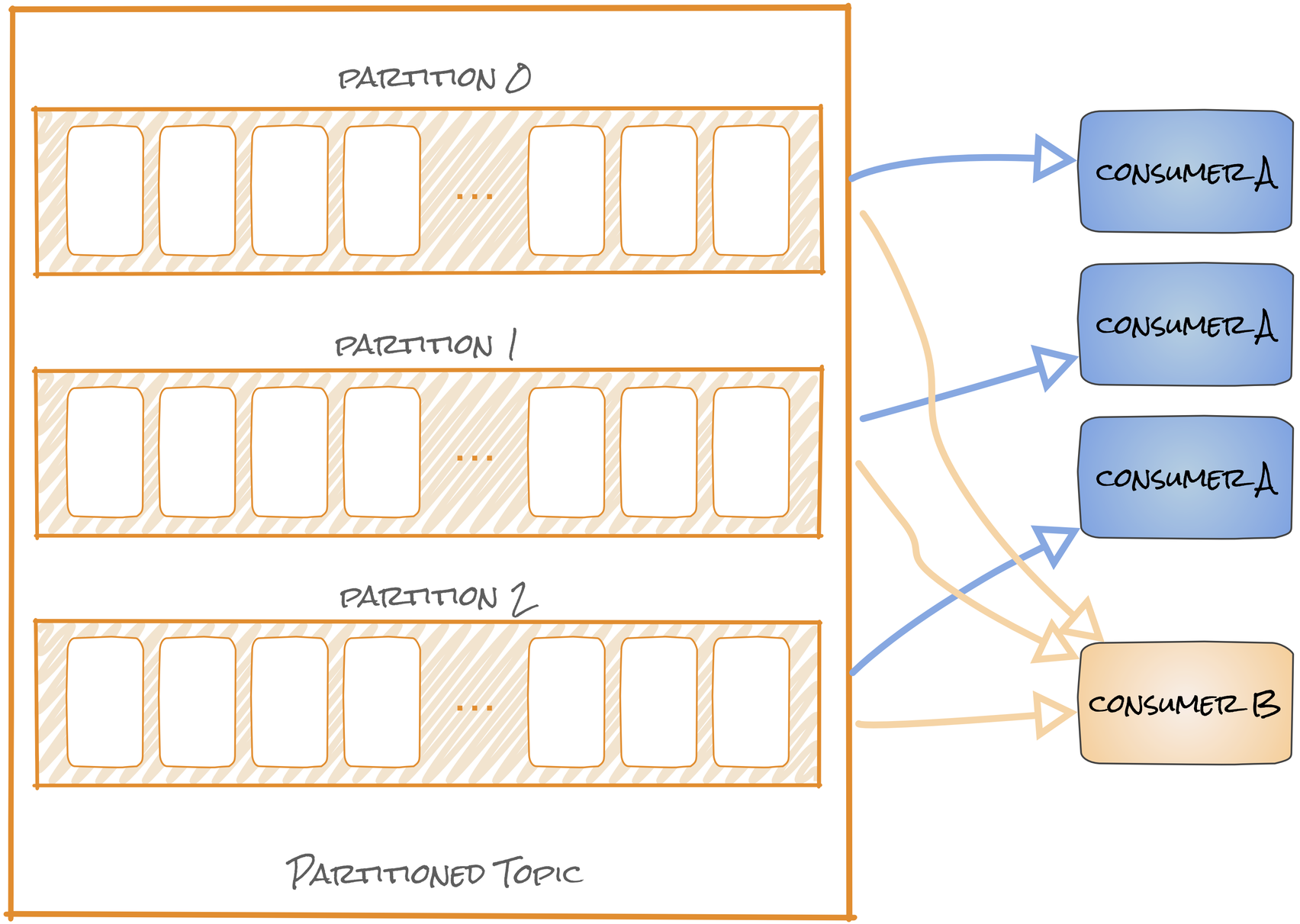
https://rmoff.dev/qcon-schemas


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

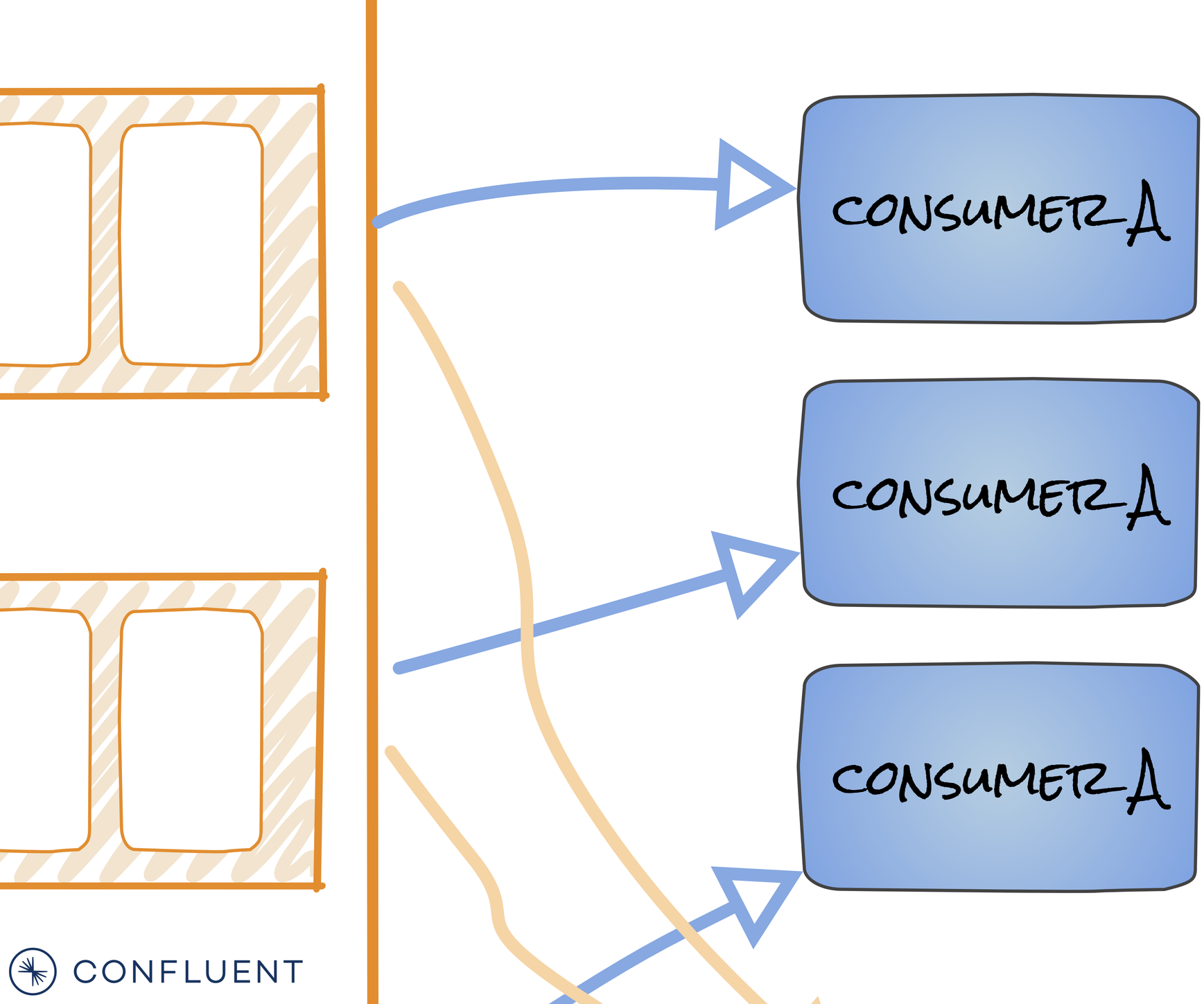
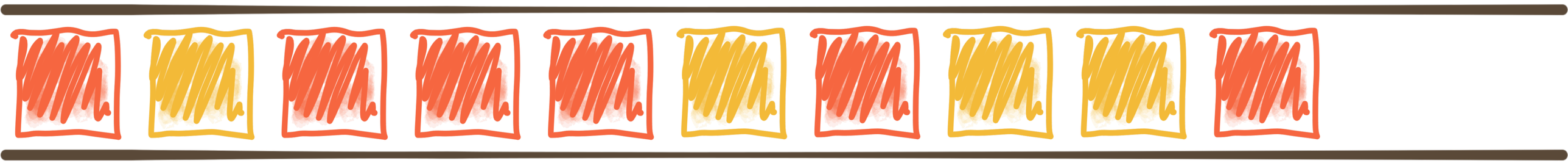

CONSUMER A
CONSUMER A
CONSUMER A



Photo by Franck V. on Unsplash

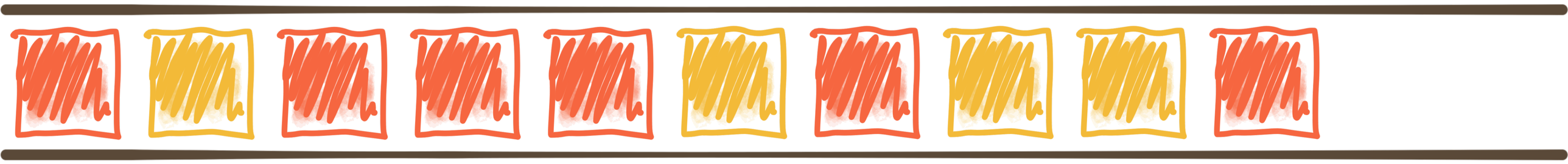
Streams of events



Time



Stream Processing



Stream: widgets



Stream: widgets_red

Stream Processing with Kafka Streams

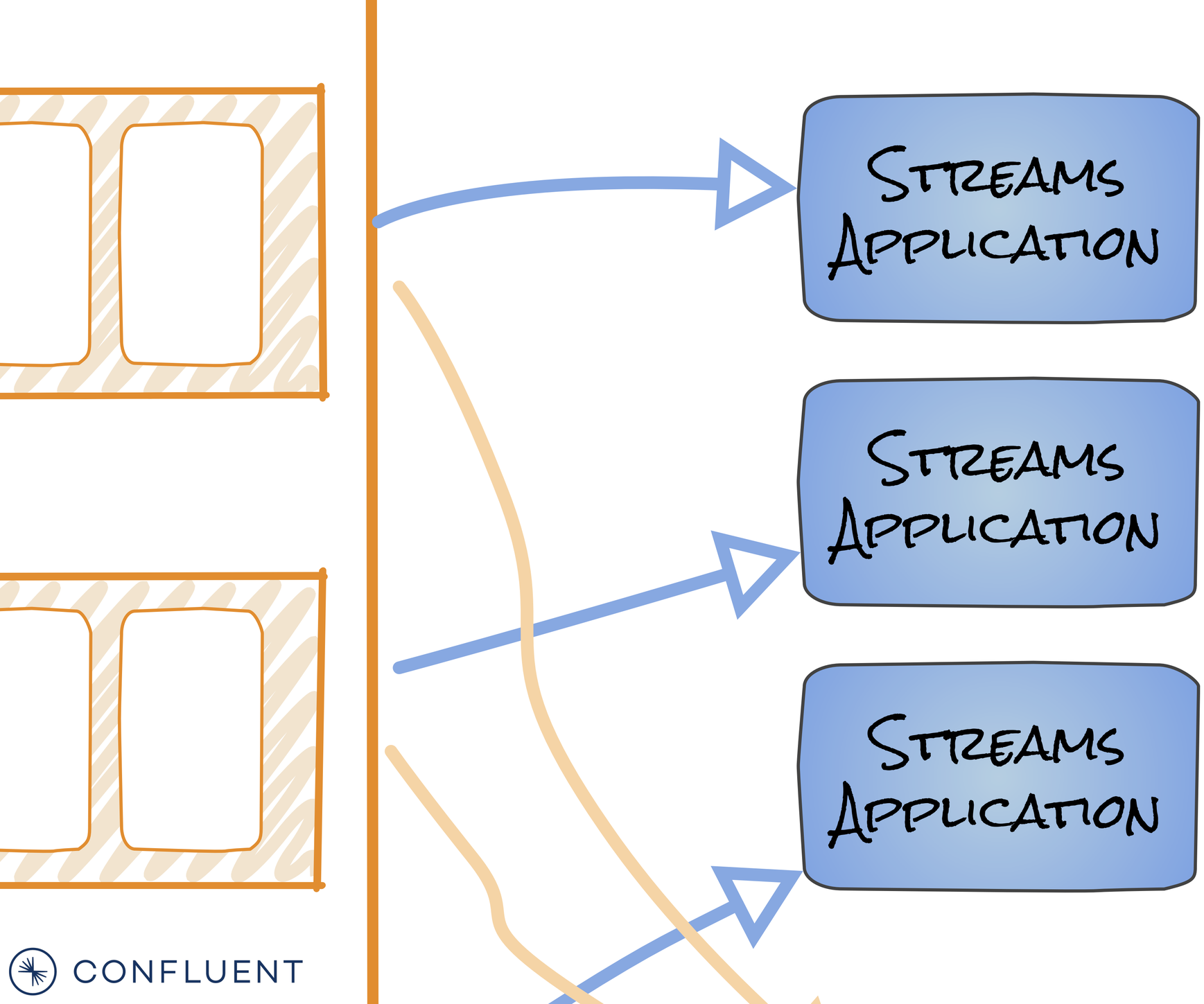


Stream:
widgets

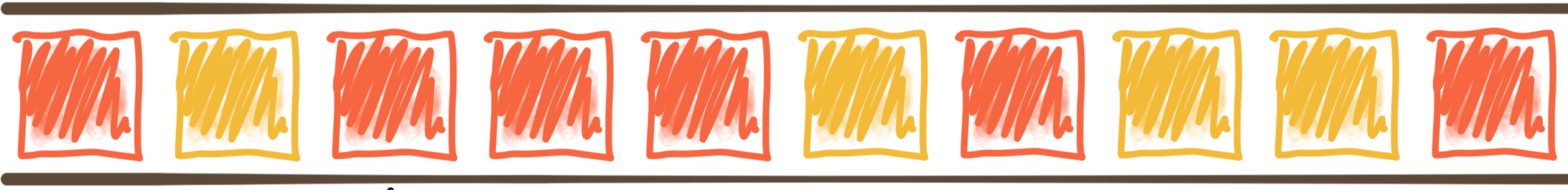
```
final StreamsBuilder builder = new StreamsBuilder()
    .stream("widgets", Consumed.with(stringSerde, widgetsSerde))
    .filter( (key, widget) -> widget.getColour().equals("RED") )
    .to("widgets_red", Produced.with(stringSerde, widgetsSerde));
```



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
```



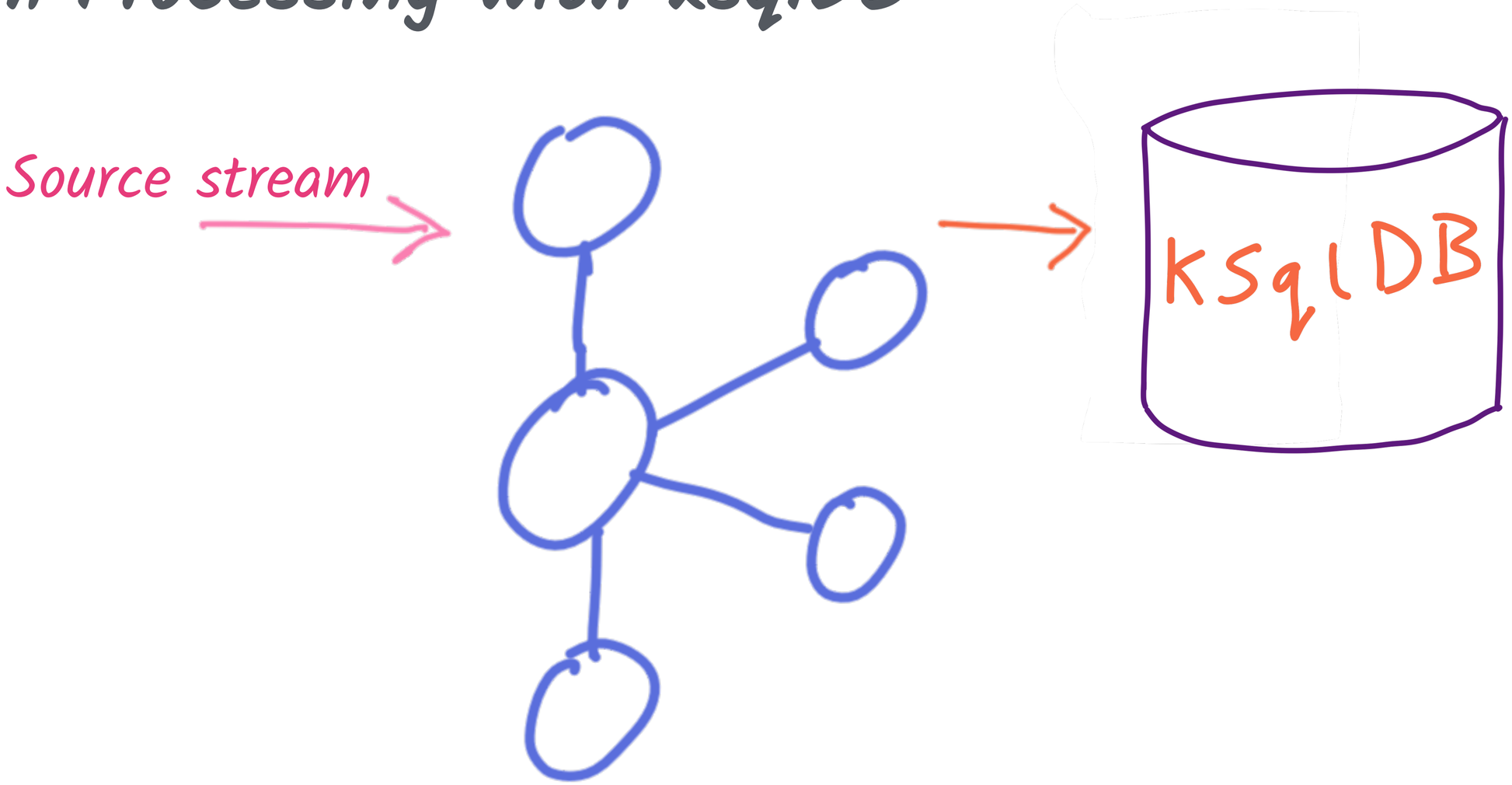
Object store, data warehouse, RDBMS

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

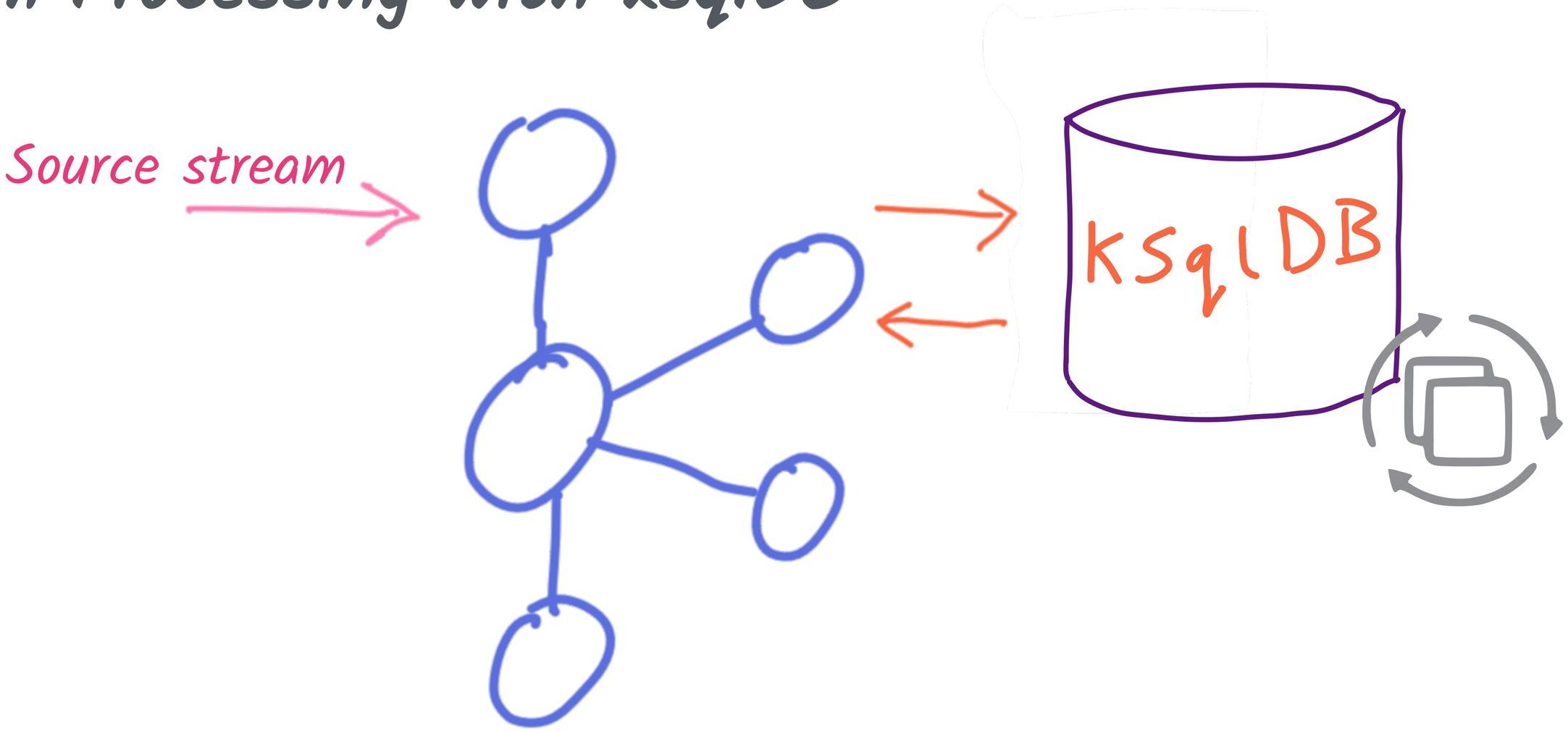
```
{
    "reading_ts": "2020-02-14T12:19:27Z",
    "sensor_id": "aa-101",
    "production_line": "w01",
    "widget_type": "acme94",
    "temp_celcius": 23,
    "widget_weight_g": 100
}
```

Photo by Franck V. on Unsplash

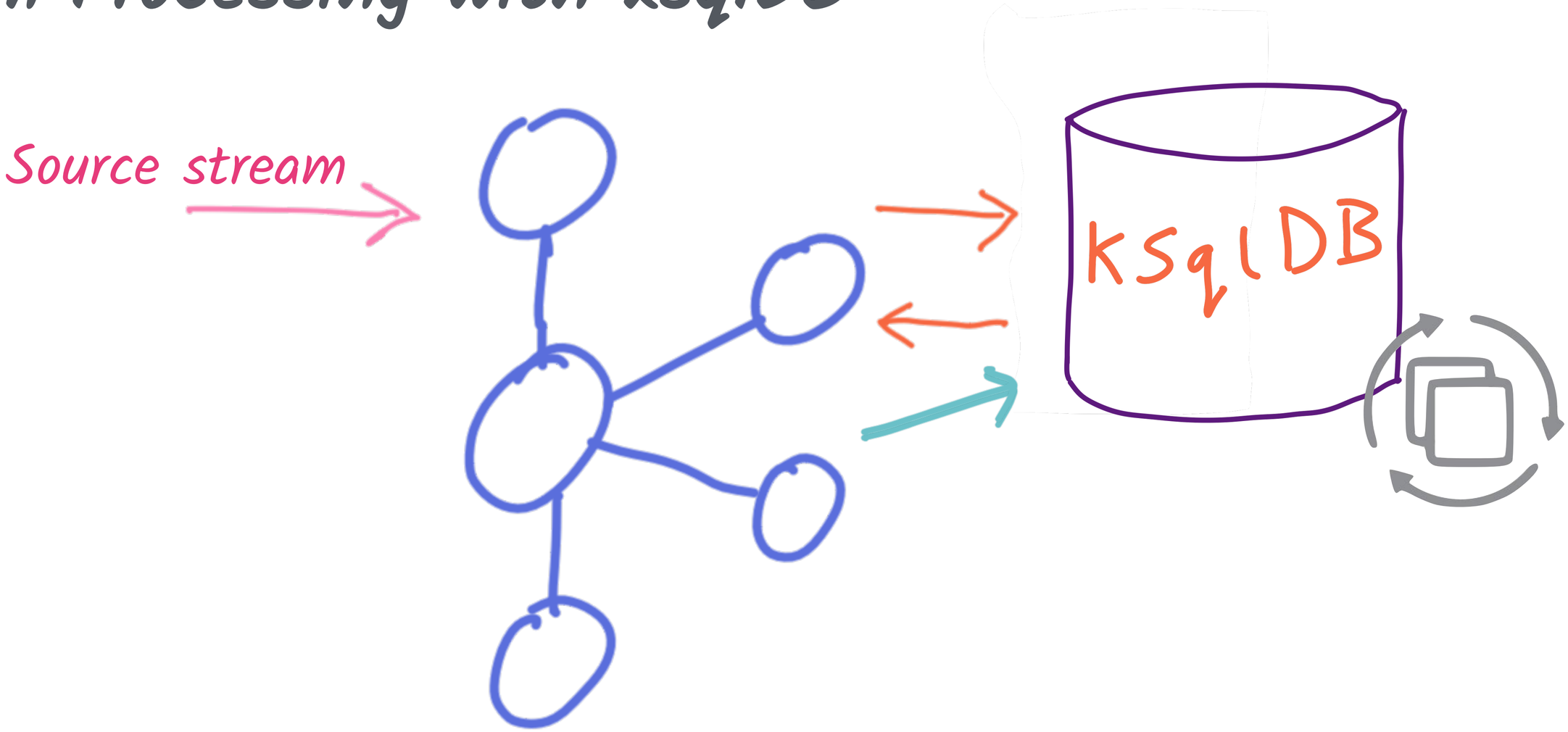
Stream Processing with ksqlDB



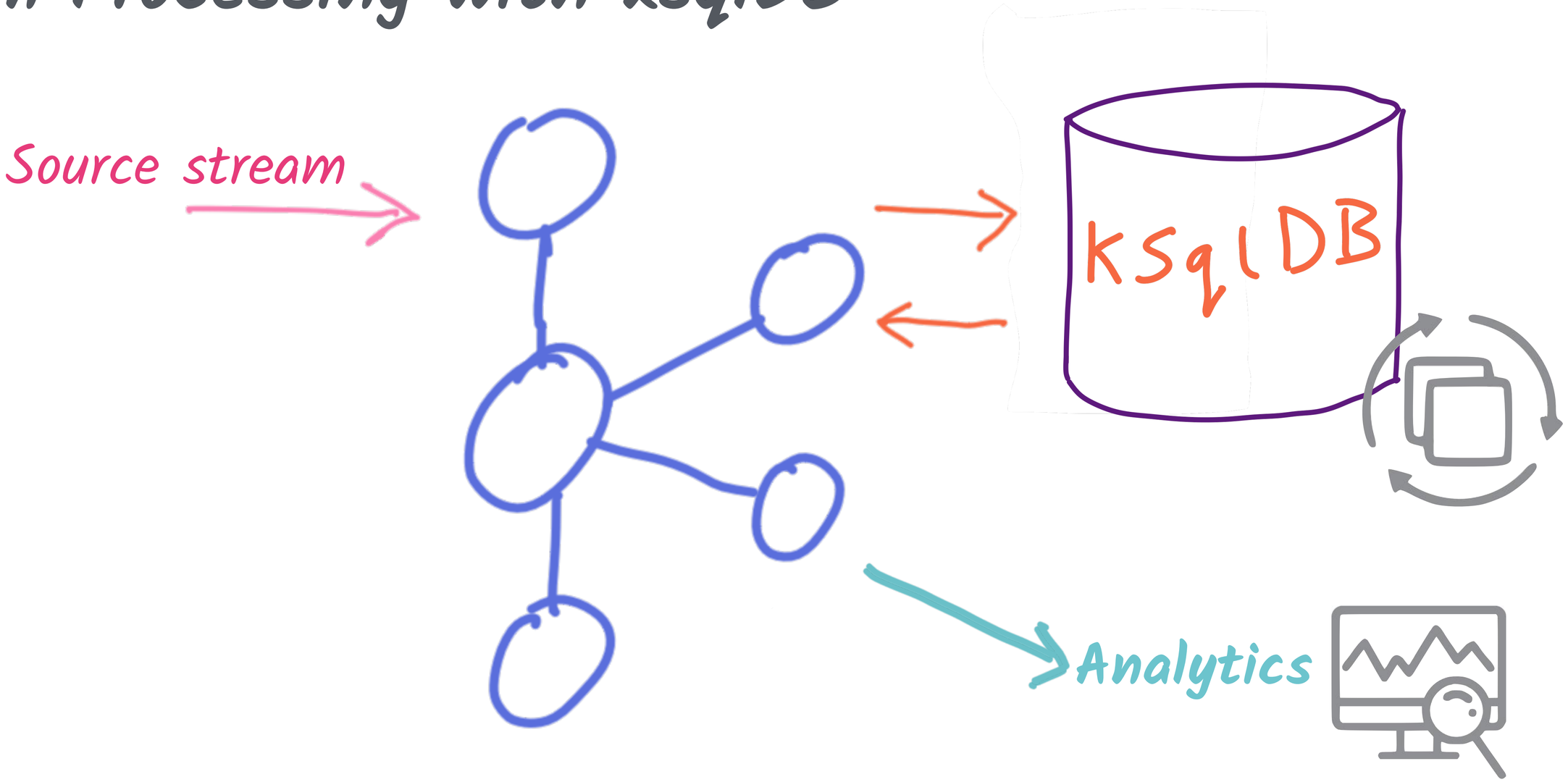
Stream Processing with ksqlDB



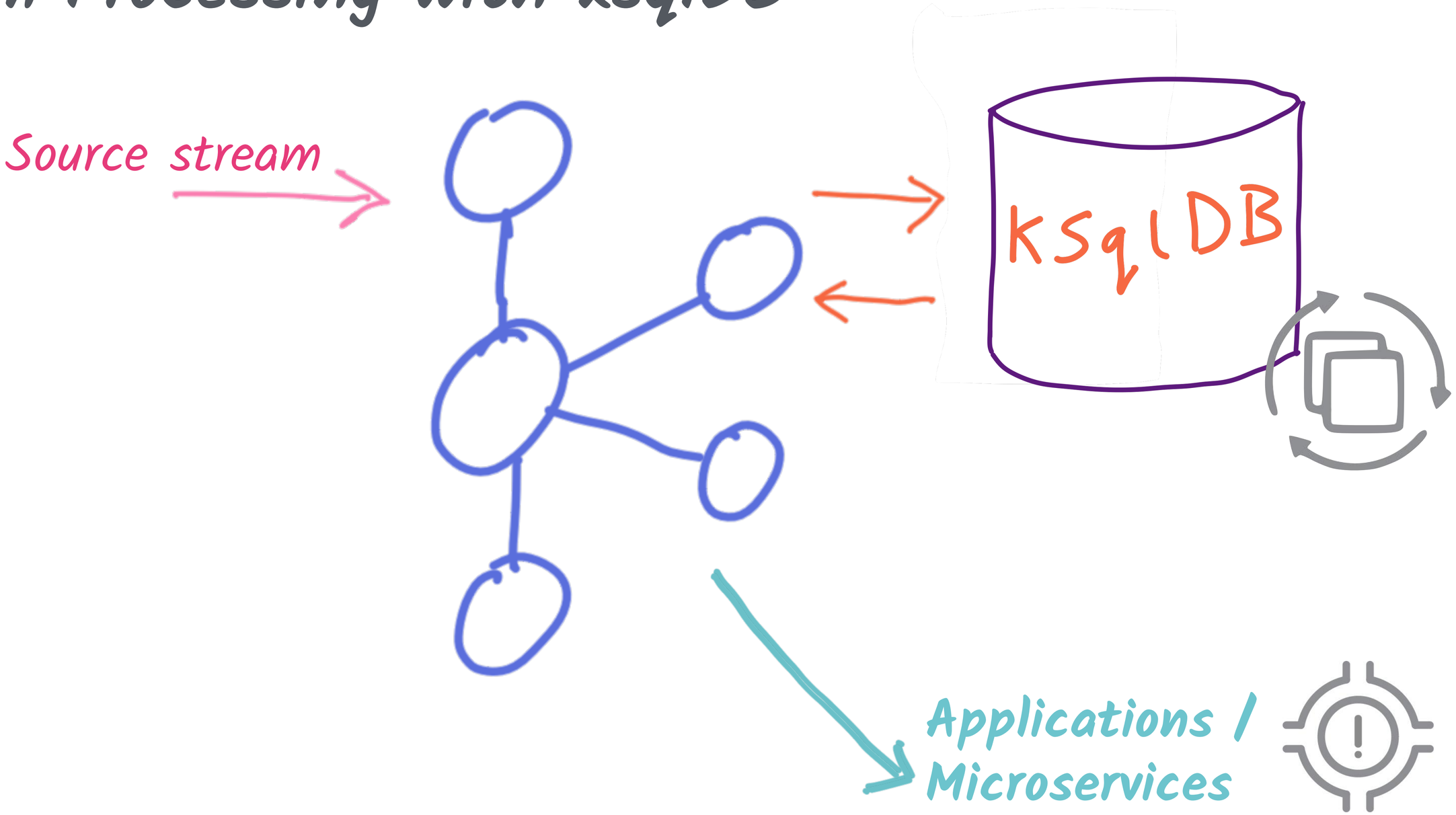
Stream Processing with ksqlDB



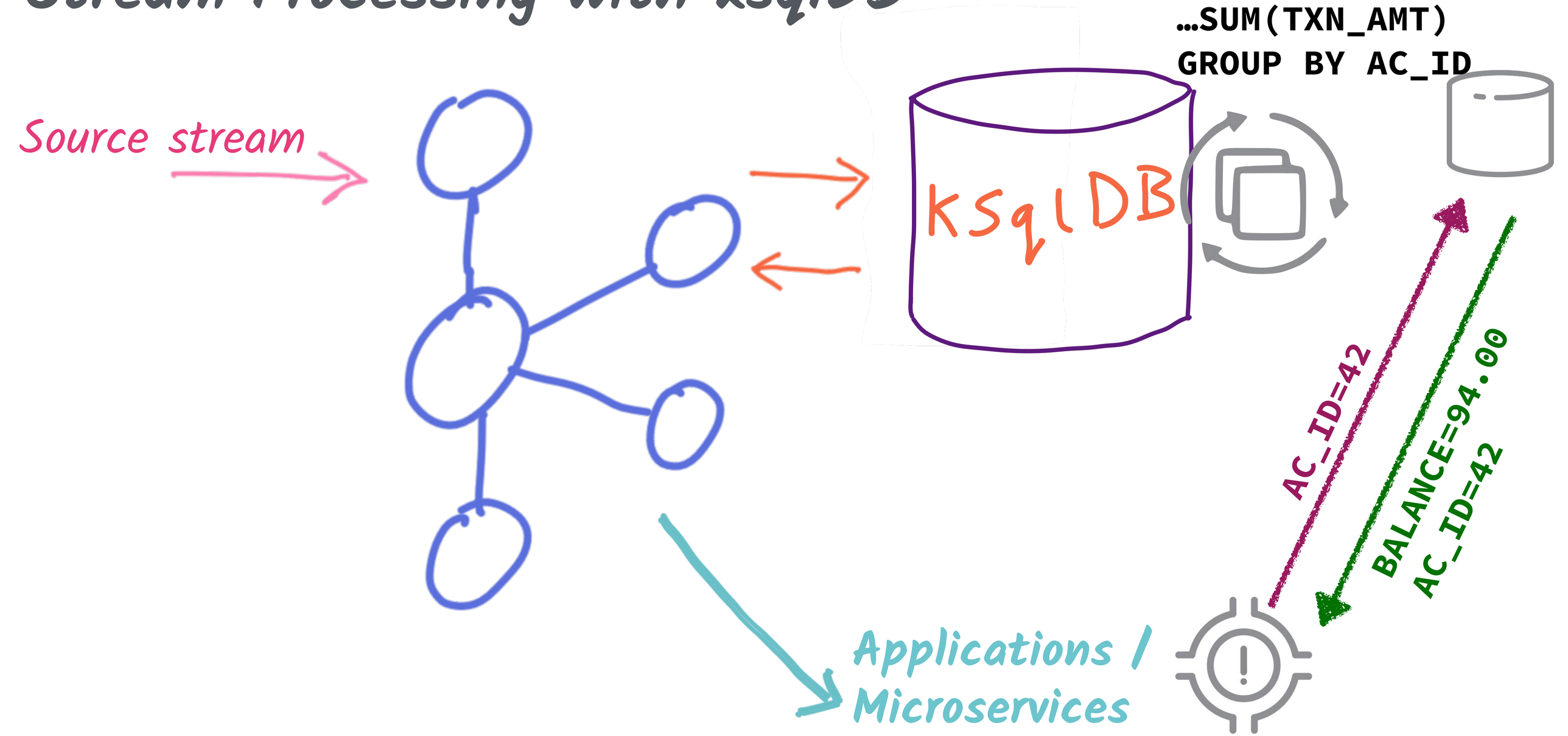
Stream Processing with ksqlDB



Stream Processing with ksqlDB



Stream Processing with ksqlDB



CONFLUENT

ksqlDB

or

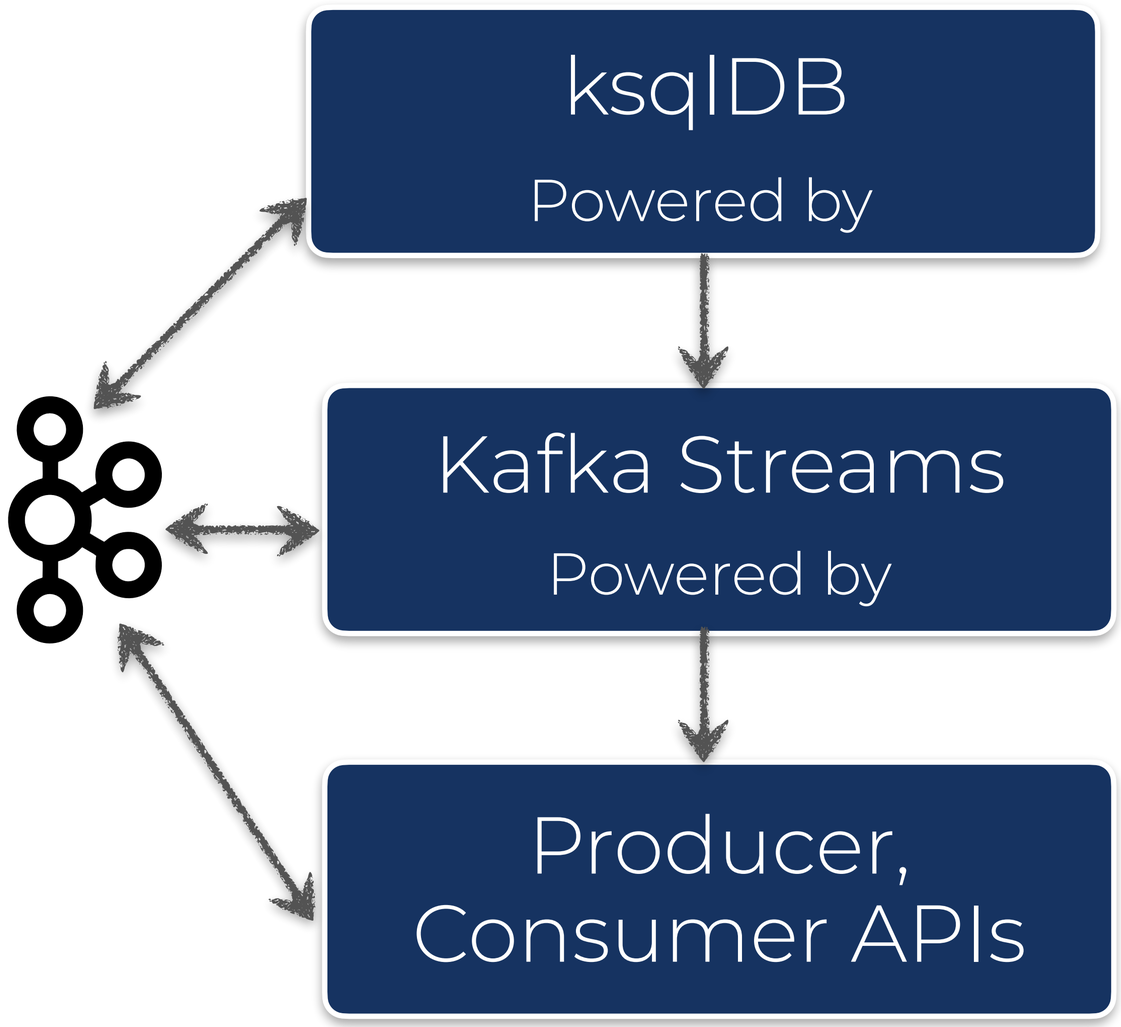
Kafka Streams?



Photo by Ramiz Dedaković on Unsplash

@rmoff

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(), …
```

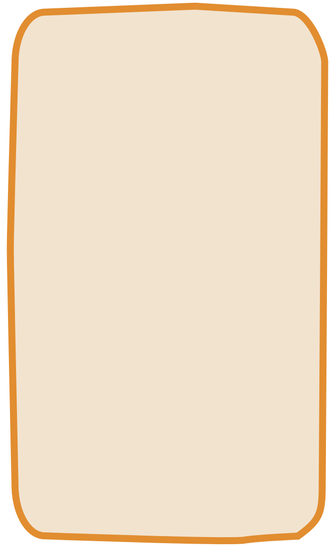


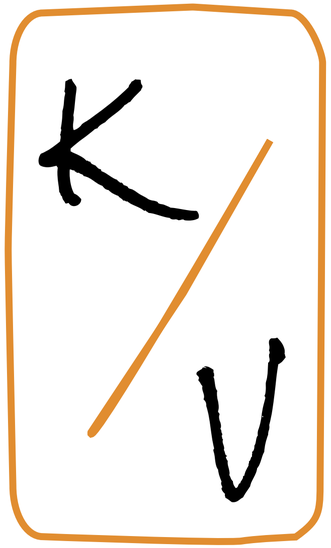
CONFLUENT

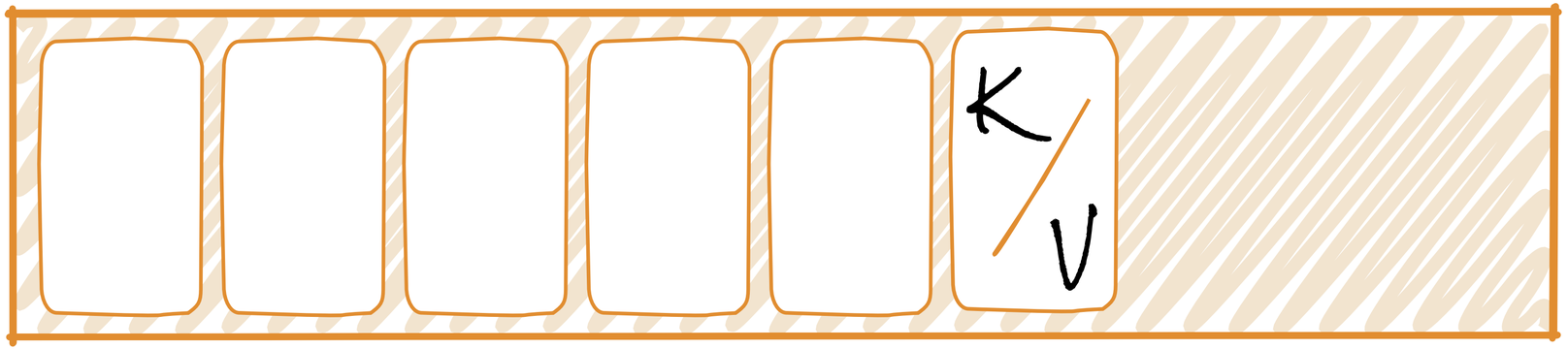
DEMO

@rmoff

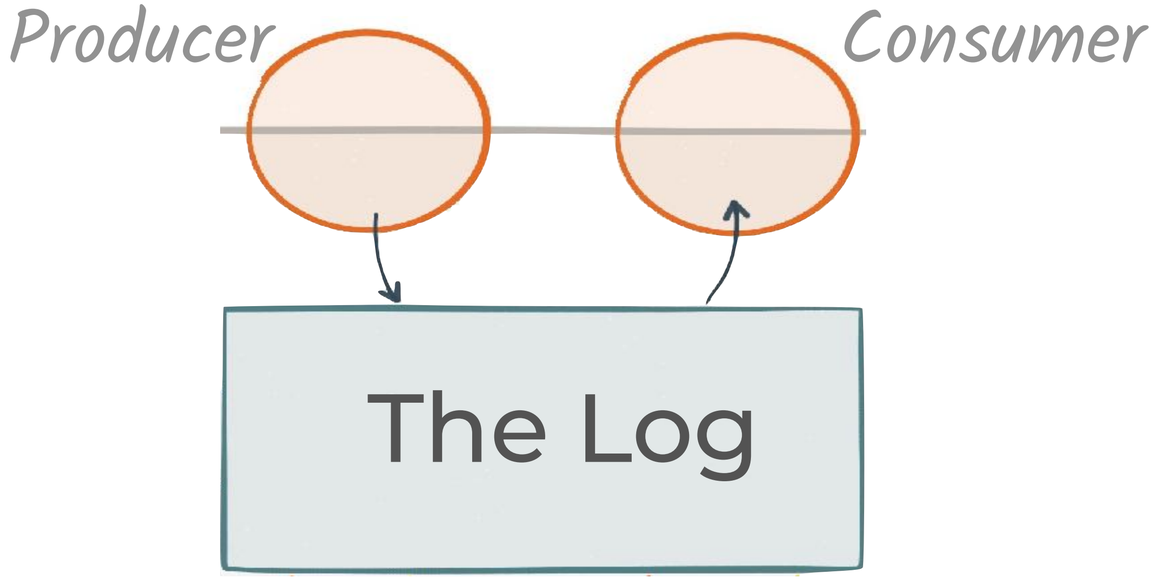
Summary

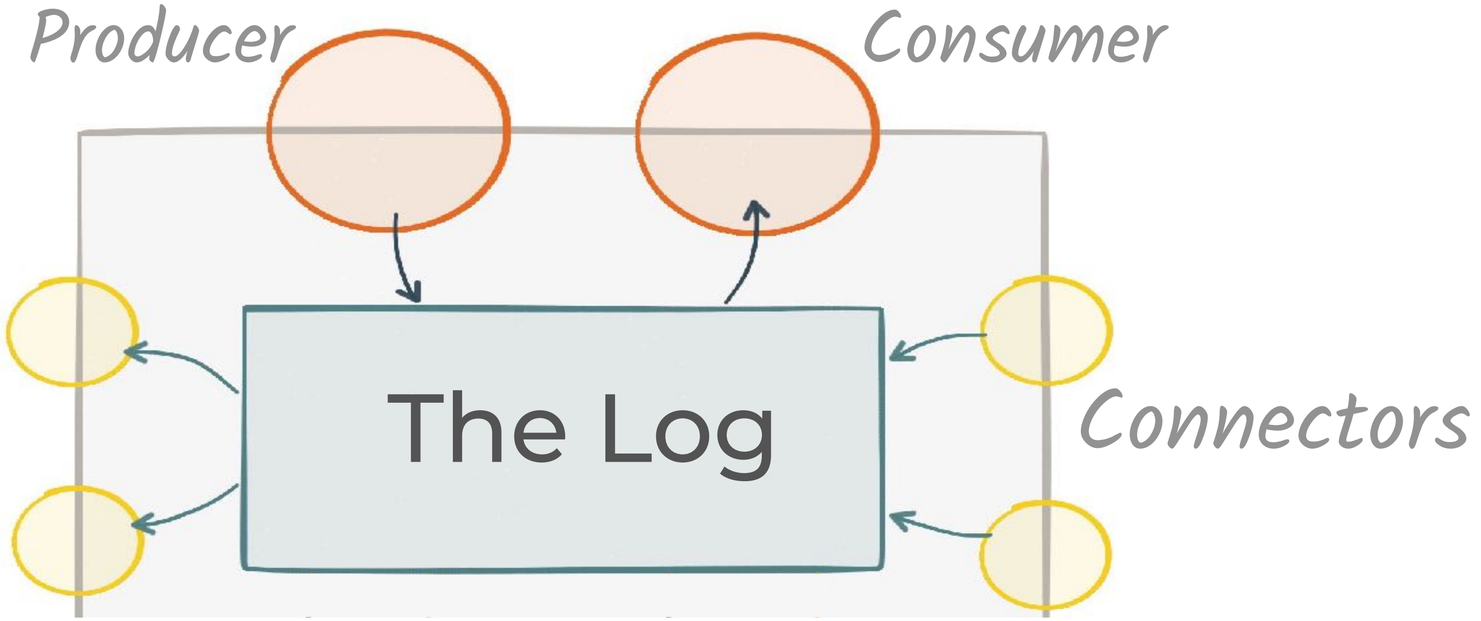


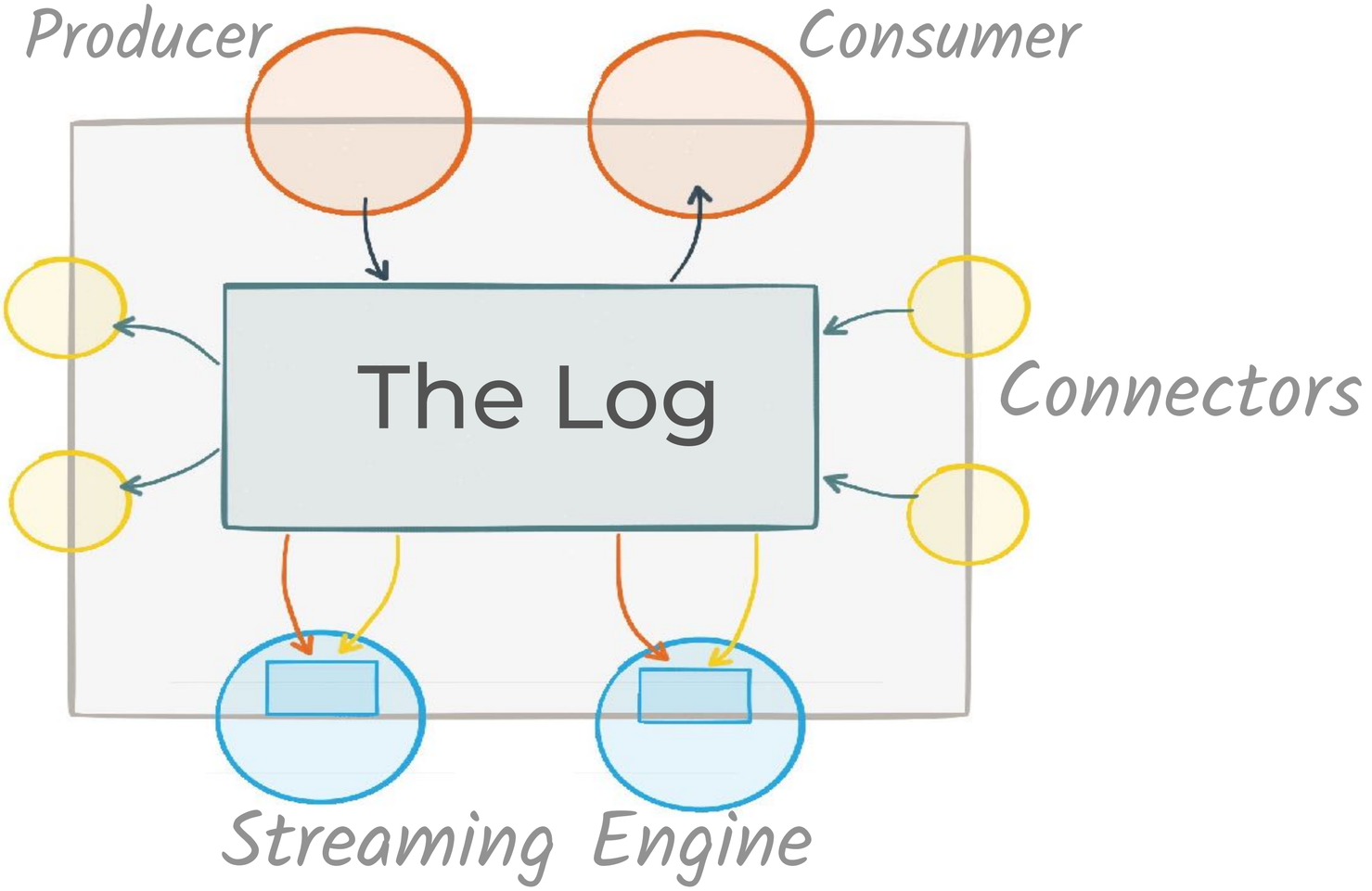

K
V


K
V

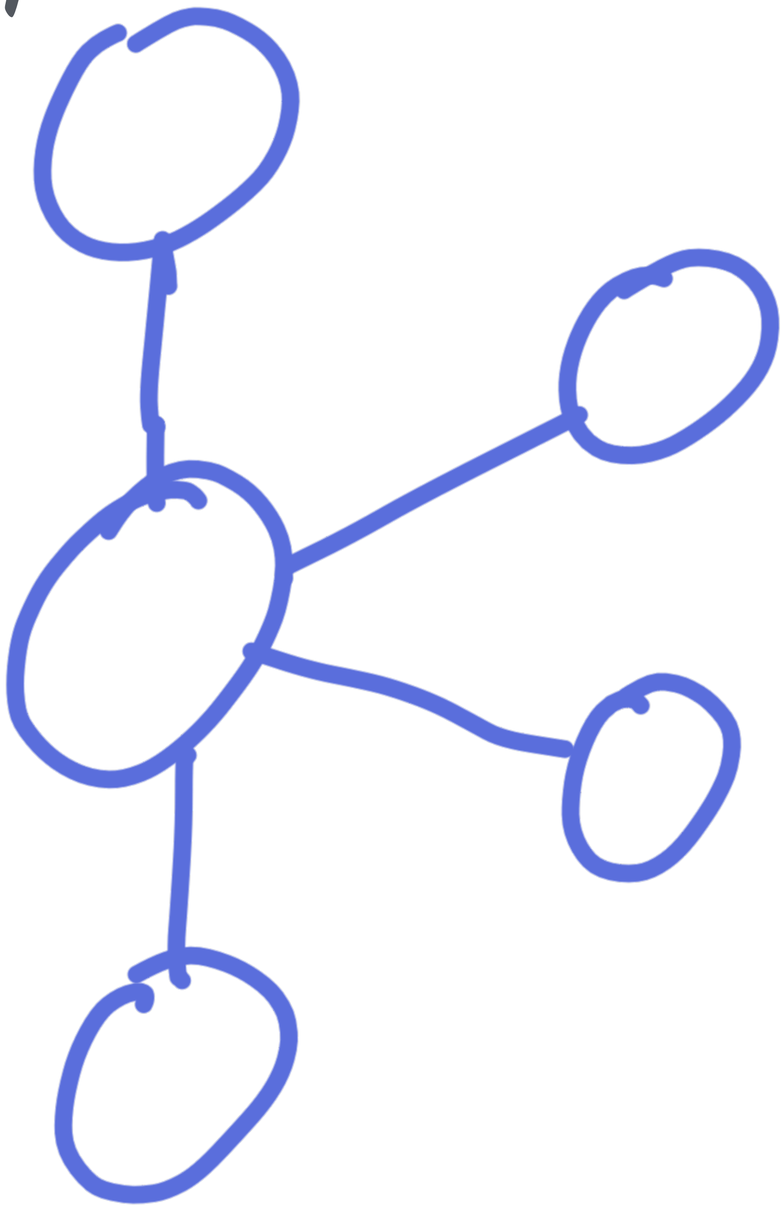
The Log

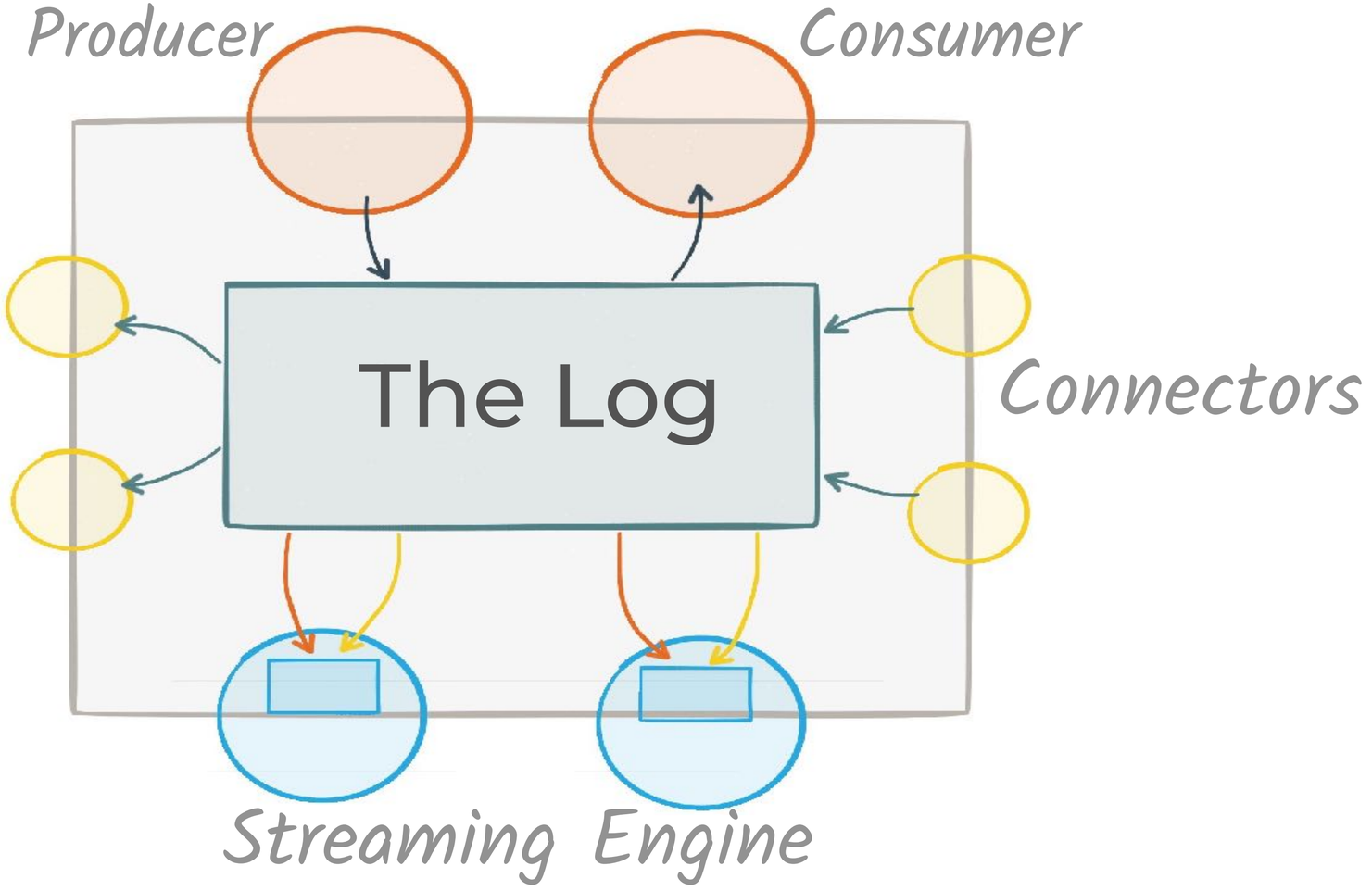

Producer
Consumer
The Log


Producer
Consumer
The Log
Connectors

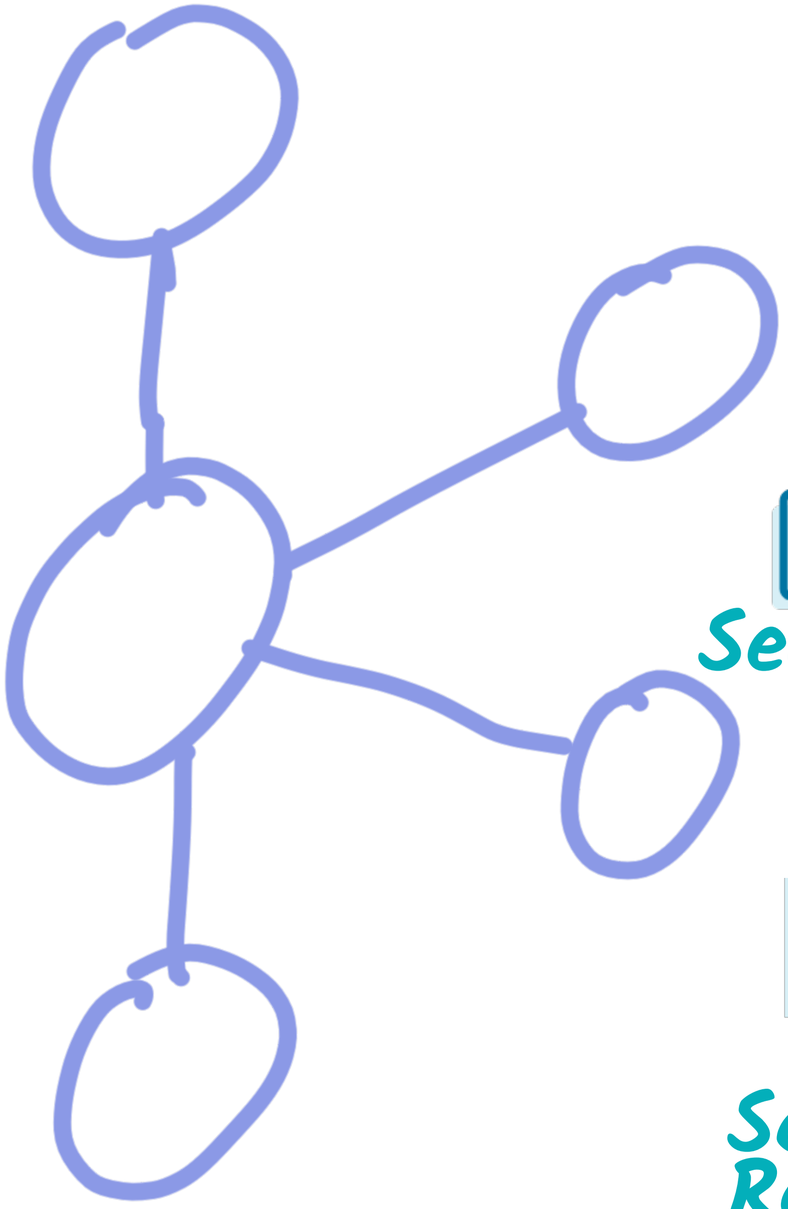

Producer
Consumer
The Log
Connectors
Streaming Engine

Apache Kafka



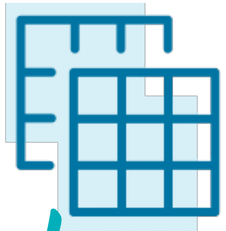


CONFLUENT

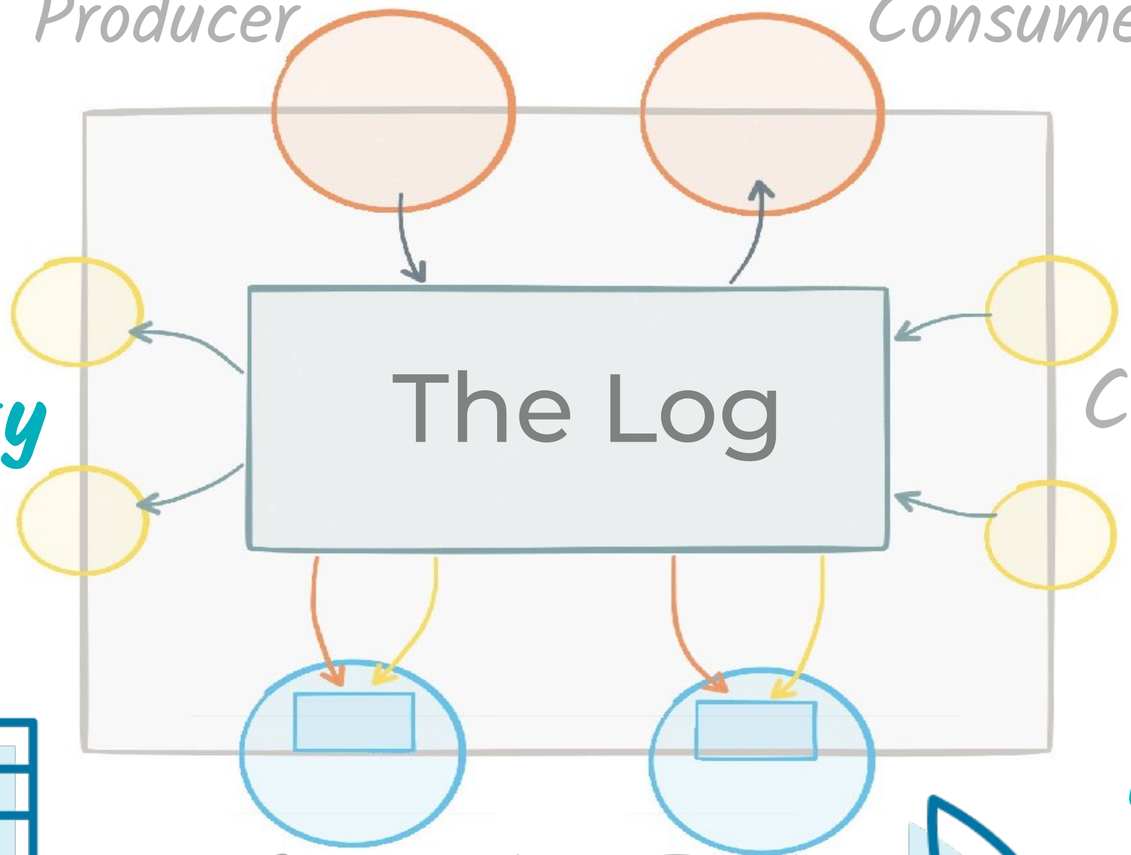




Security



Schema Registry





REST Proxy



Confluent Control Center



ksqlDB

EVENTS

are

EVERYWHERE

EVENTS

are

very POWERFUL


Learn Kafka.
Start building with
Apache Kafka at
Confluent Developer.
developer.confluent.io

@rmoff

#EOF

rmoff.dev/talks