Kafka as a Platform: the Ecosystem from the Ground Up



Robin Moffatt
@rmoff

#kafkasummit



EVENTS

@rmoff

EVENTS

EVENTS

- Something happened
- What happened

Human generated events

A Sale



A Stock movement

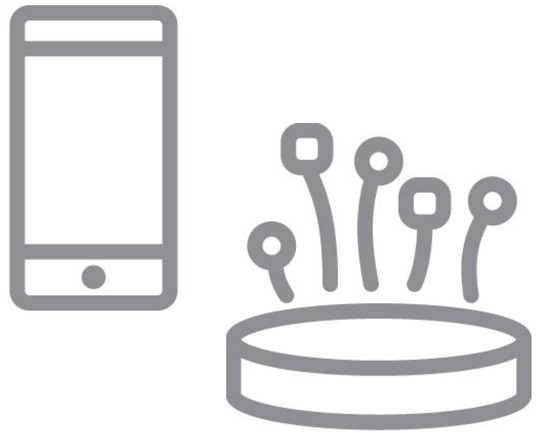


Machine generated events

Networking



IoT



Applications



EVENTS

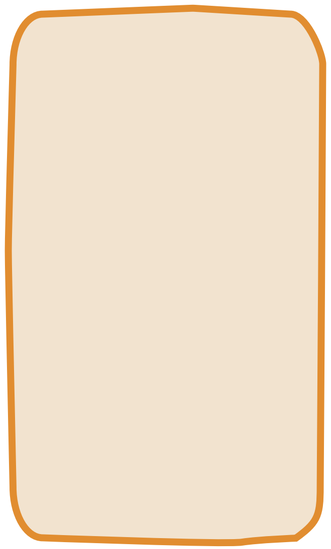
are

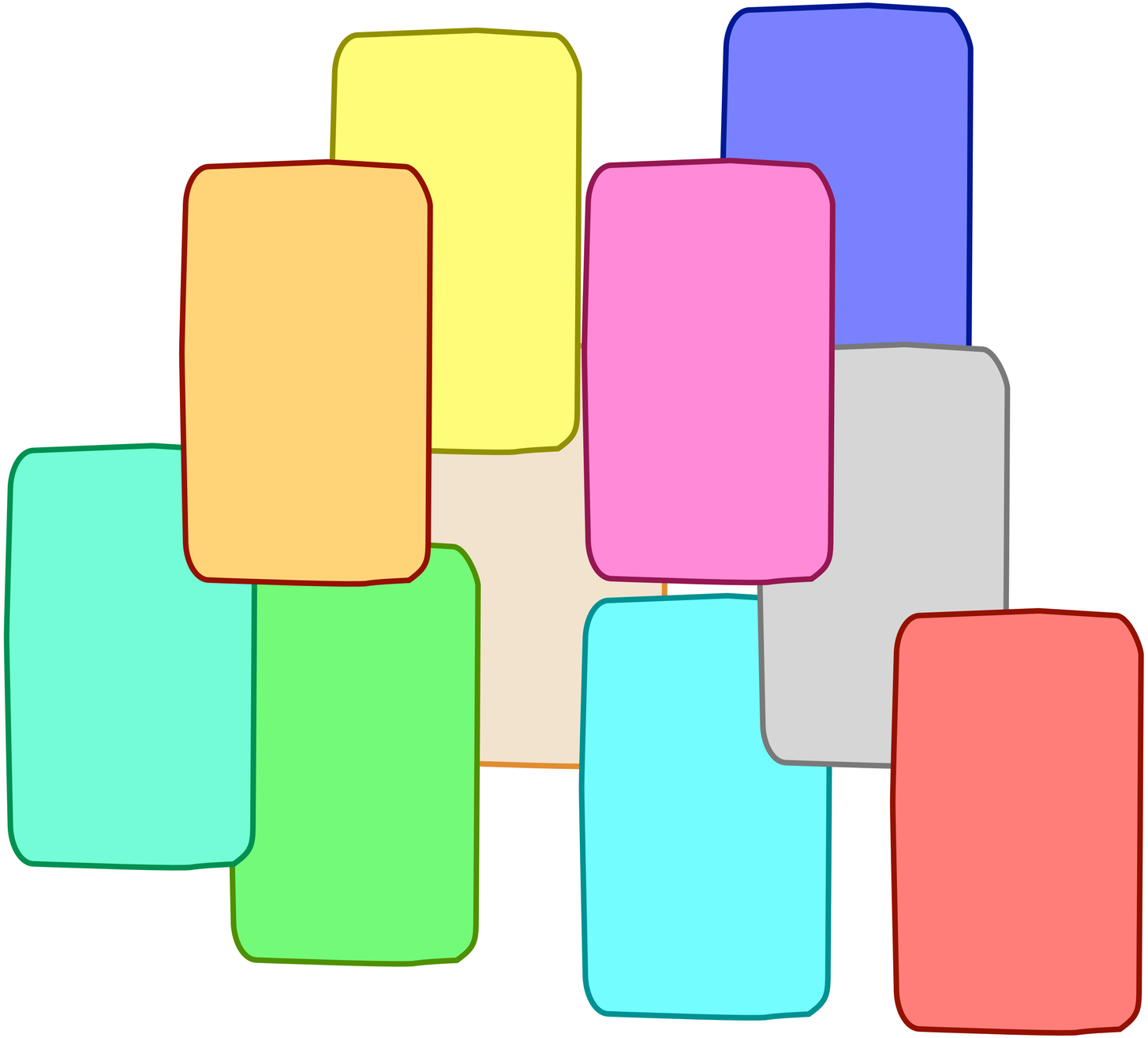
EVERYWHERE

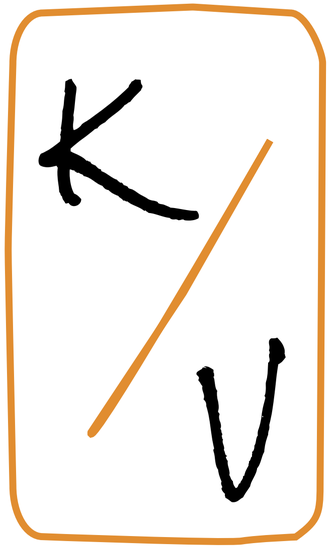
EVENTS

are

very POWERFUL






K
V



LOG

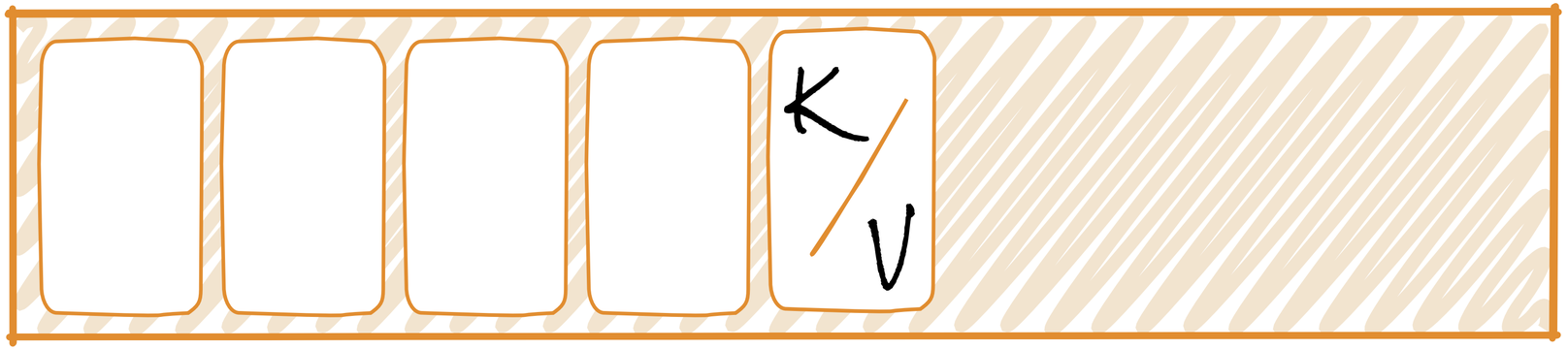
@rmoff

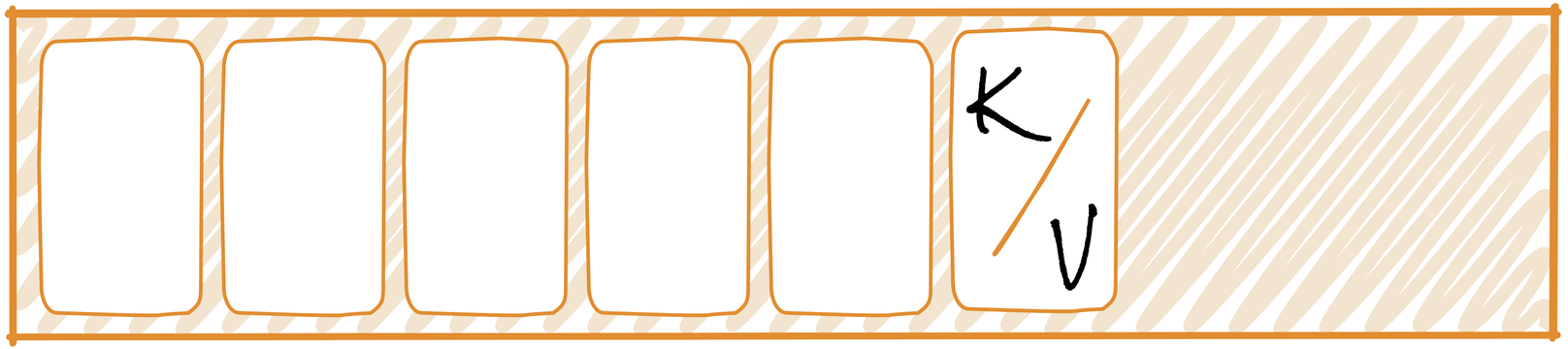

K
V

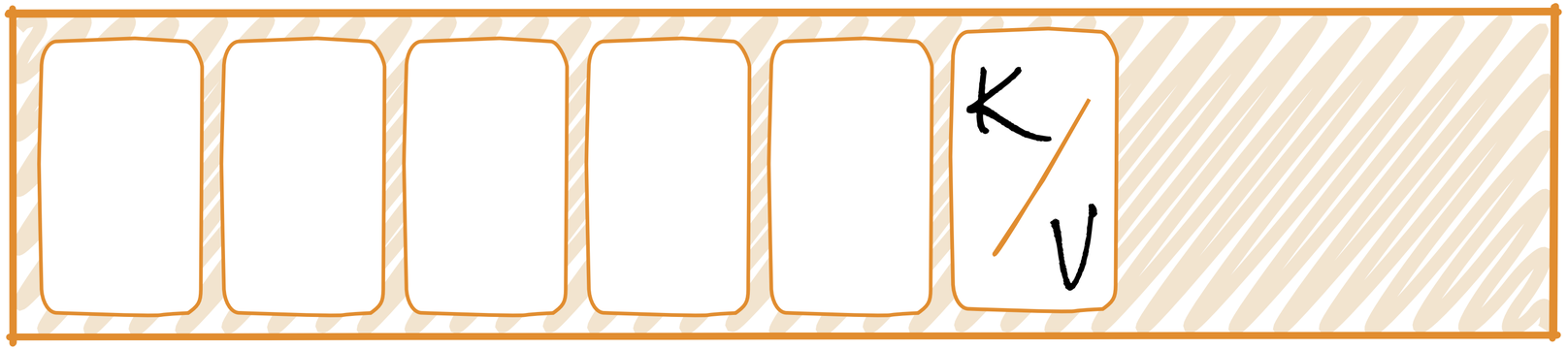

K
V


K
V

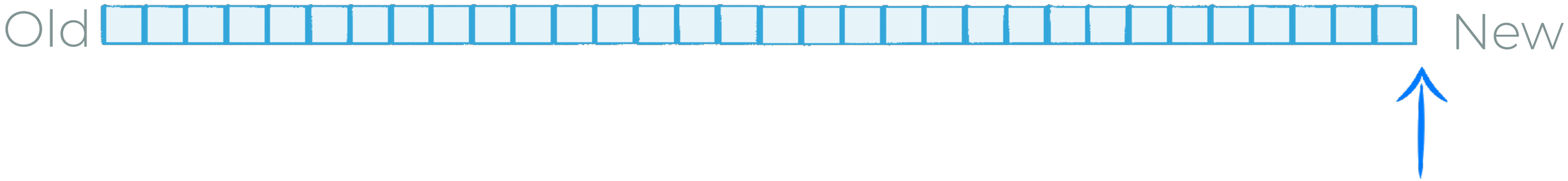

K
V


K
V


K
V


K
V

Immutable Event Log



Events are added at the end of the log



TOPICS

@rmoff

Topics

Clicks



Orders



Customers



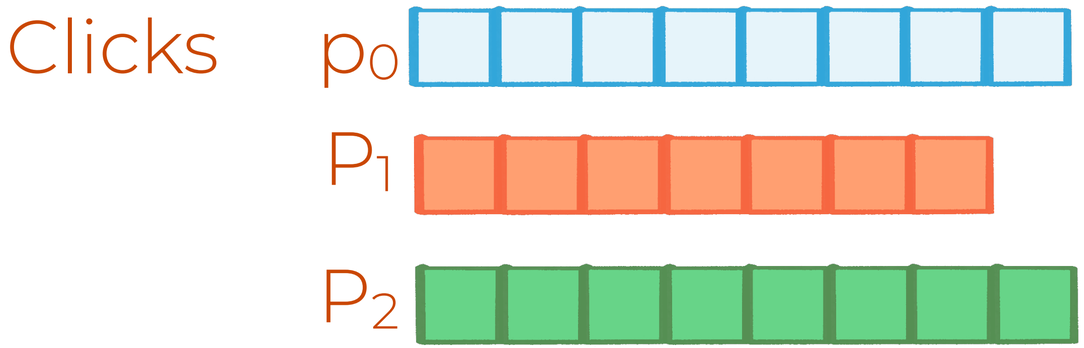
Topics are similar in concept to tables in a database



PARTITIONS

@rmoff

Partitions



Messages are guaranteed to be strictly ordered within a partition



PUB / SUB

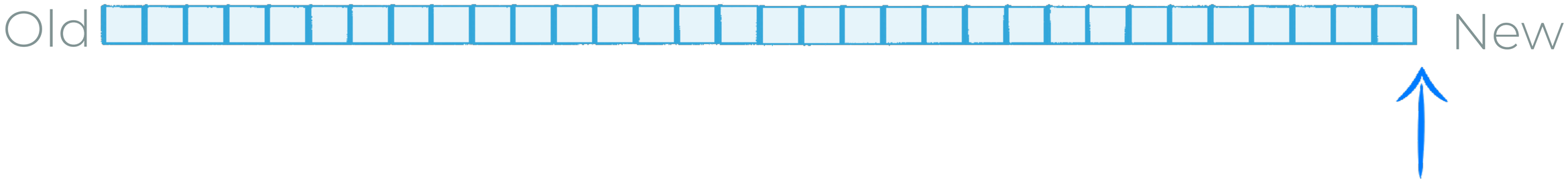
@rmoff



PUB / SUB

@rmoff

Producing data



Messages are added at the end of the log

```go
package main

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

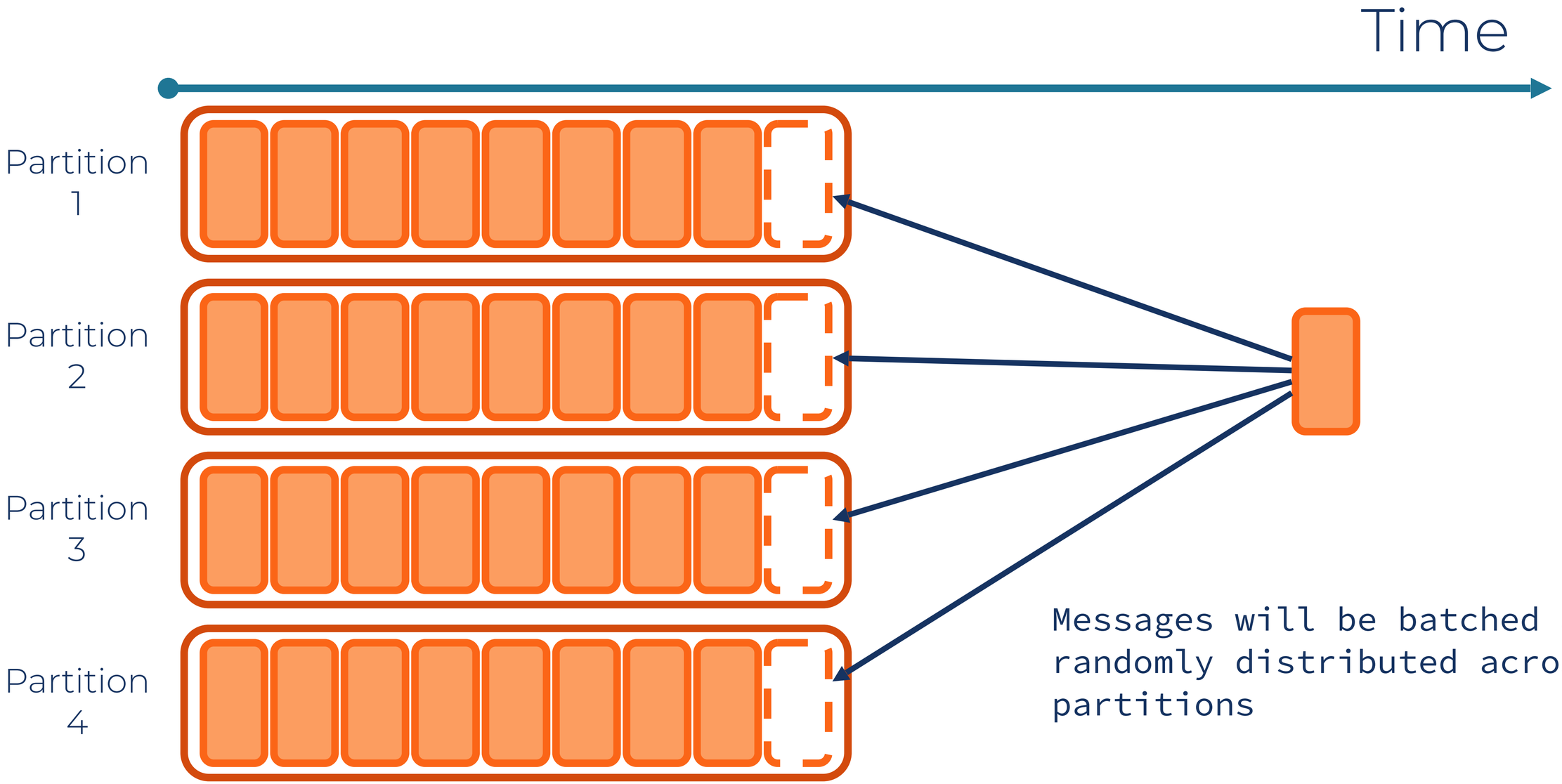
func main() {

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

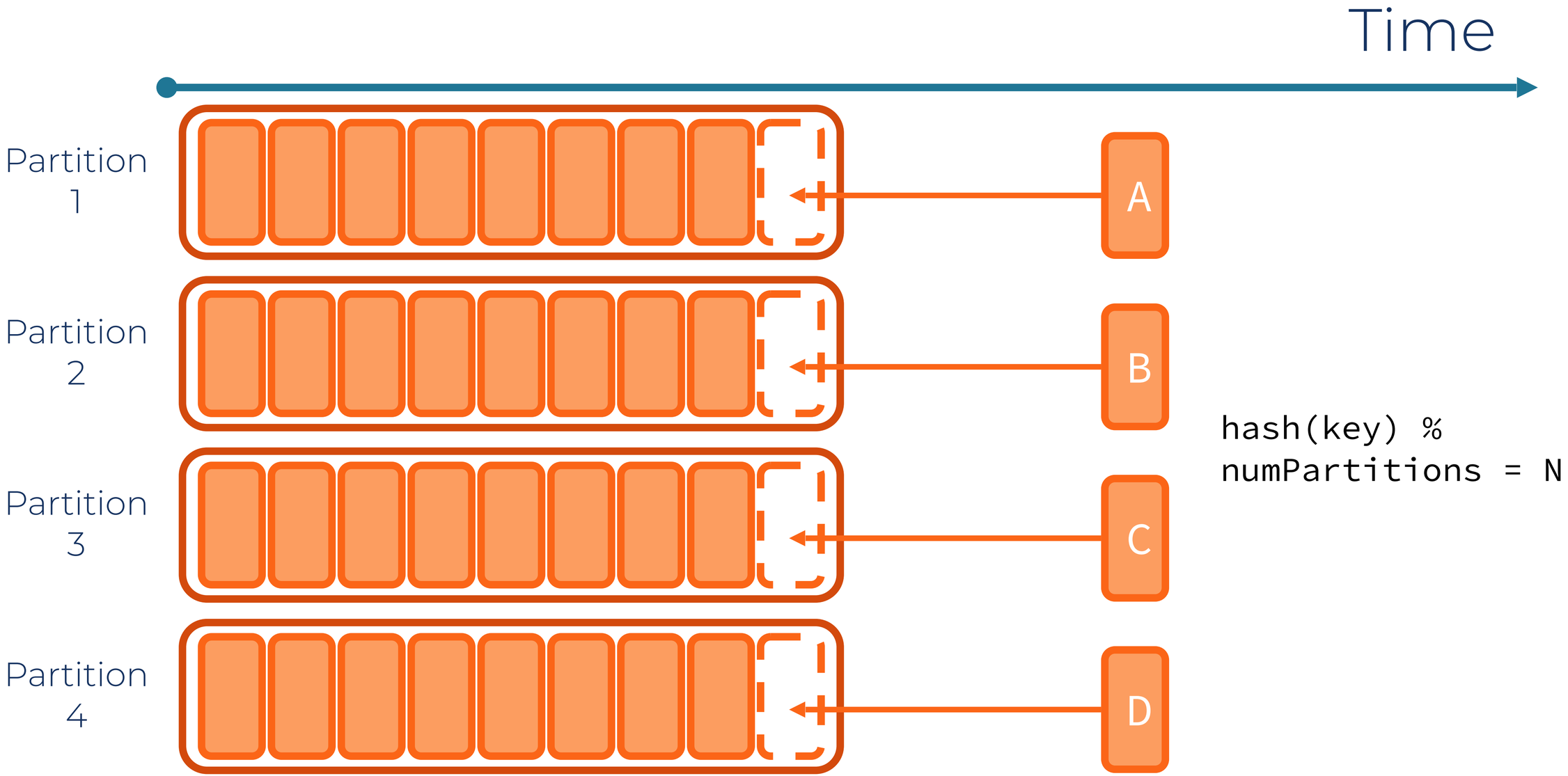
    p.Produce(&kafka.Message{
        TopicPartition: kafka.TopicPartition{Topic: &topic,
            Partition: 0},
        Value: []byte("Hello world")}, nil)

}
```

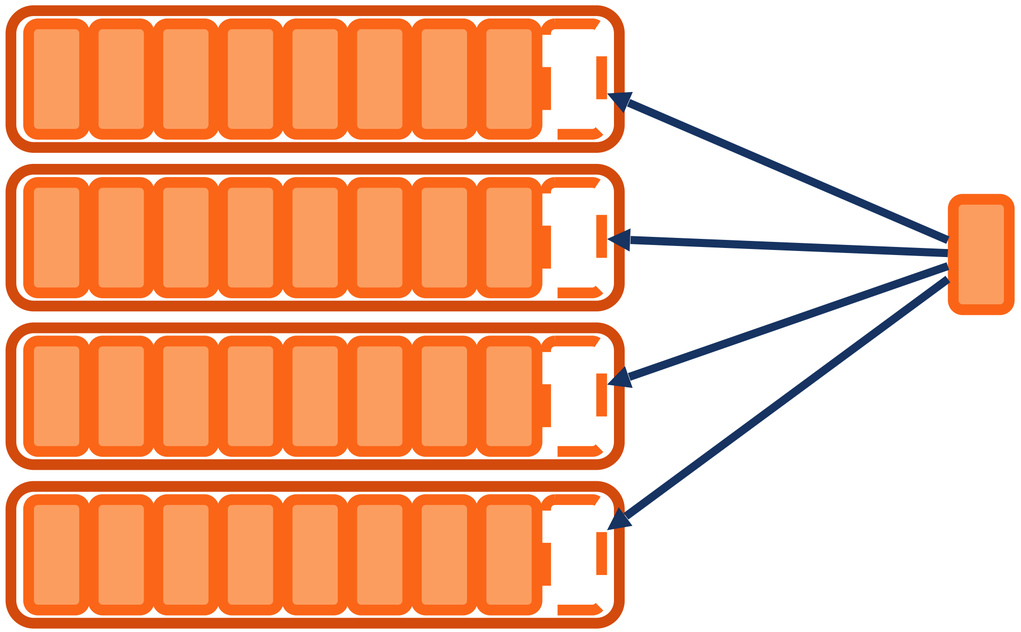
Producing to Kafka - No Key



Producing to Kafka - With Key



Producers



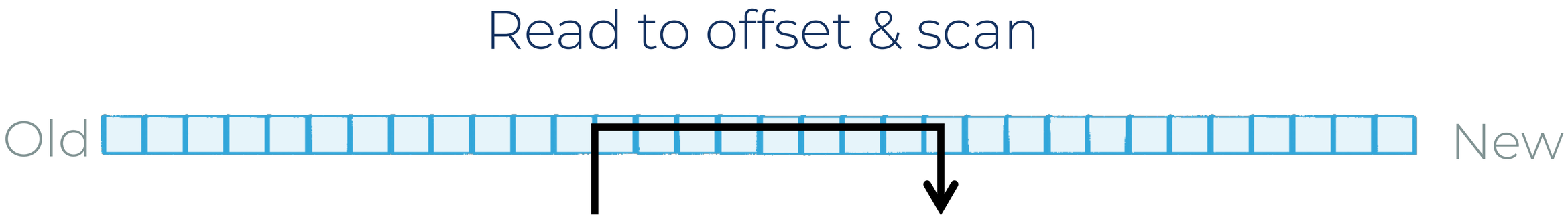
- **A client application**
- **Puts messages into topics**
- **Handles partitioning, network protocol**
- **Java, Go, .NET, C/C++, Python**
- **Also every other language**
  Plus REST proxy if not



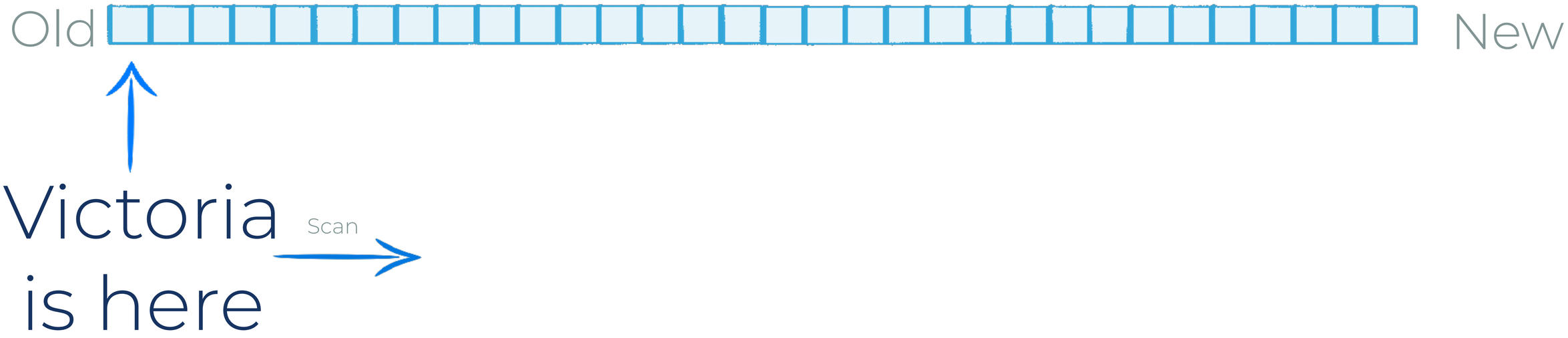
PUB / SUB

@rmoff

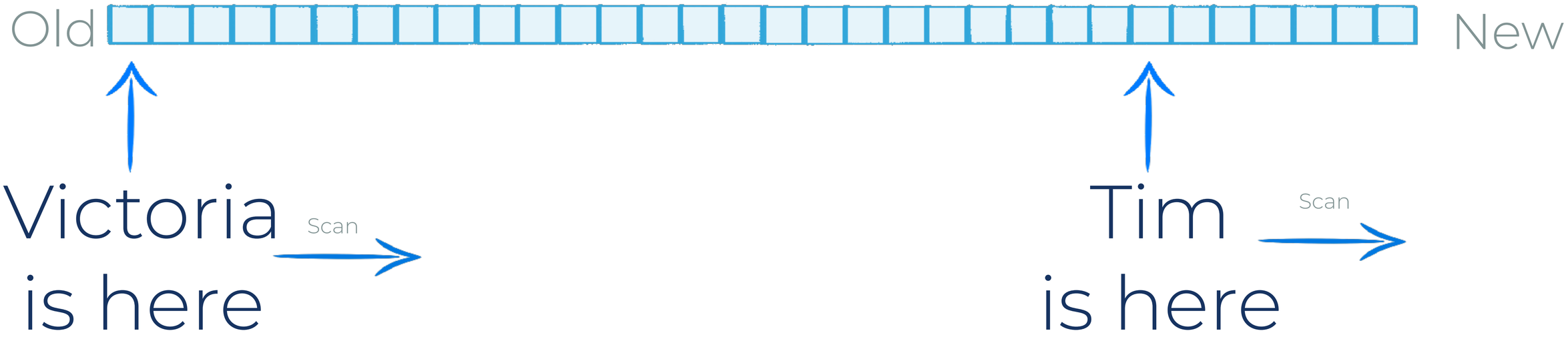
Consuming data - access is only sequential



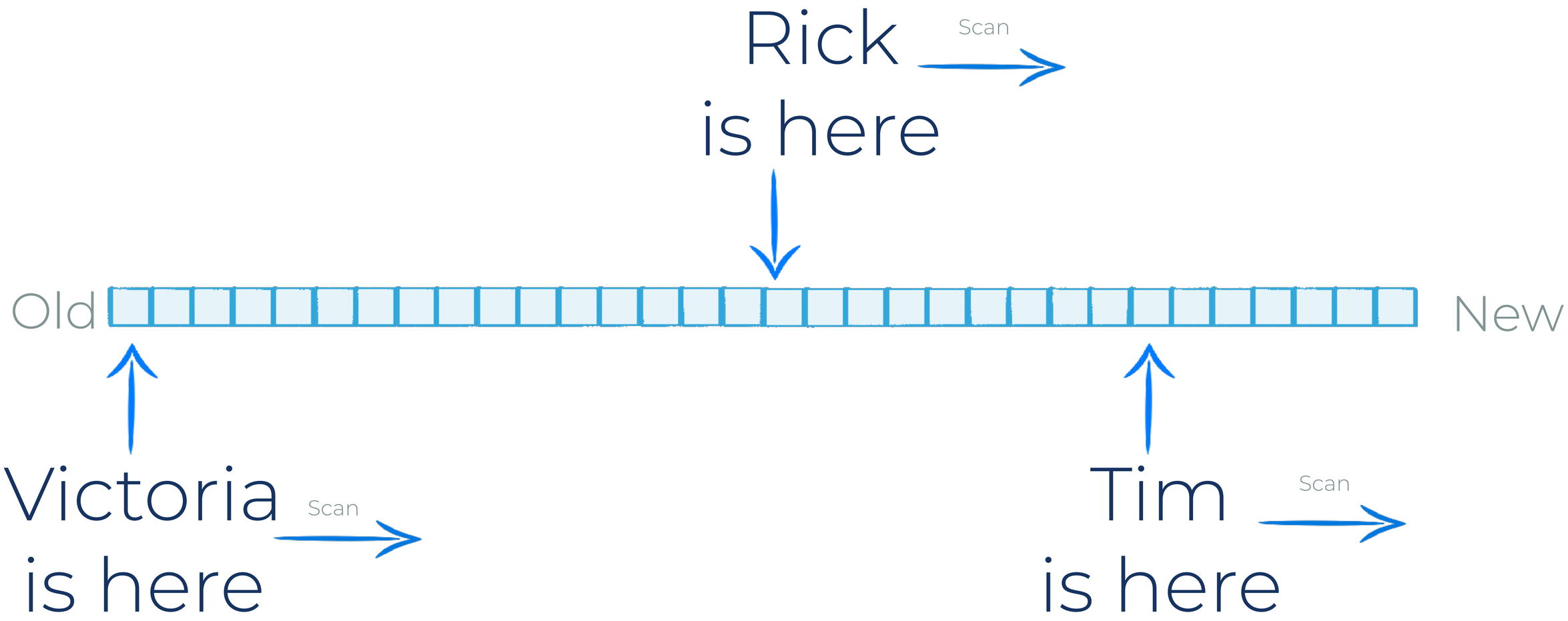
Consumers have a position of their own



Consumers have a position of their own



Consumers have a position of their own

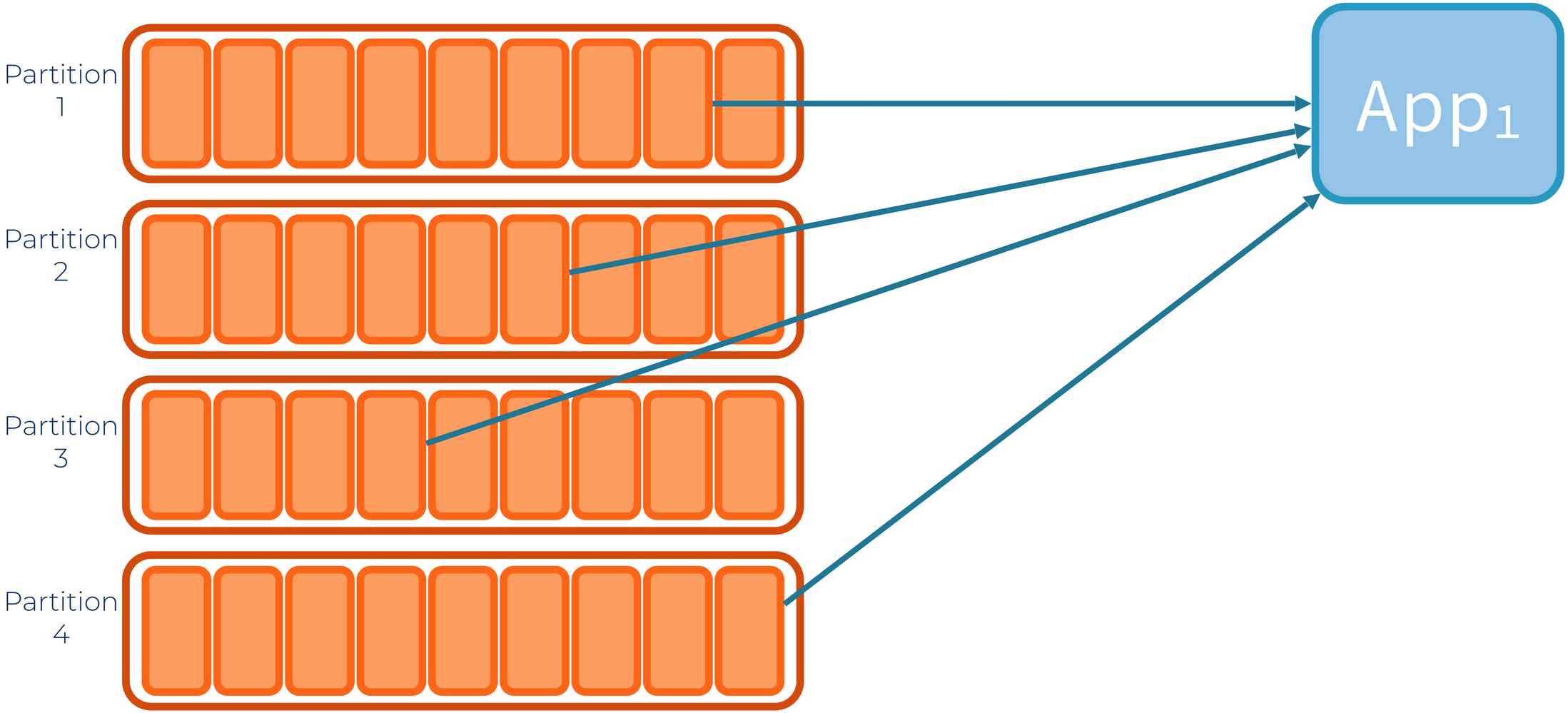


```
c, _ := kafka.NewConsumer(&cm)
defer c.Close()
c.Subscribe(topic, nil)

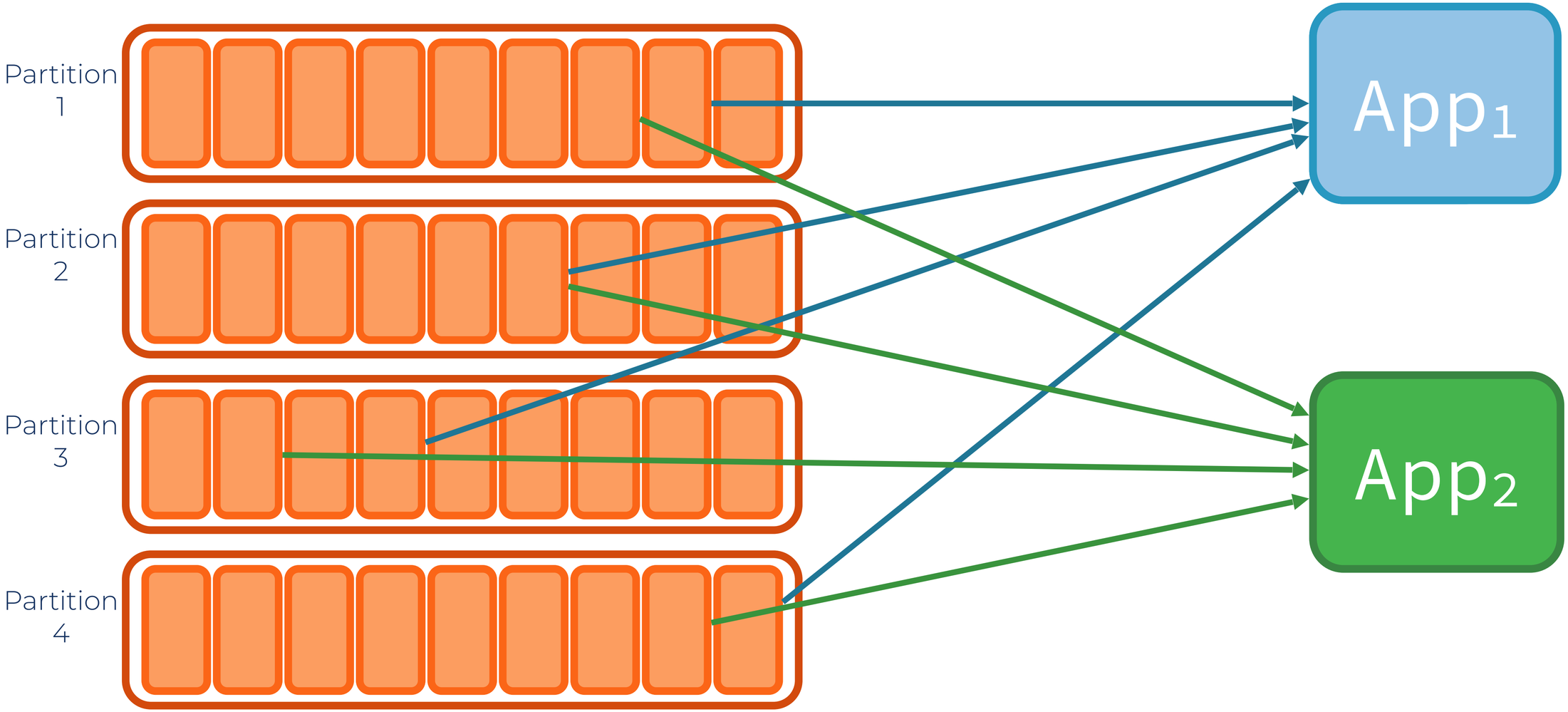
for {
    select {
    case ev := <-c.Events():
        switch ev.(type) {

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

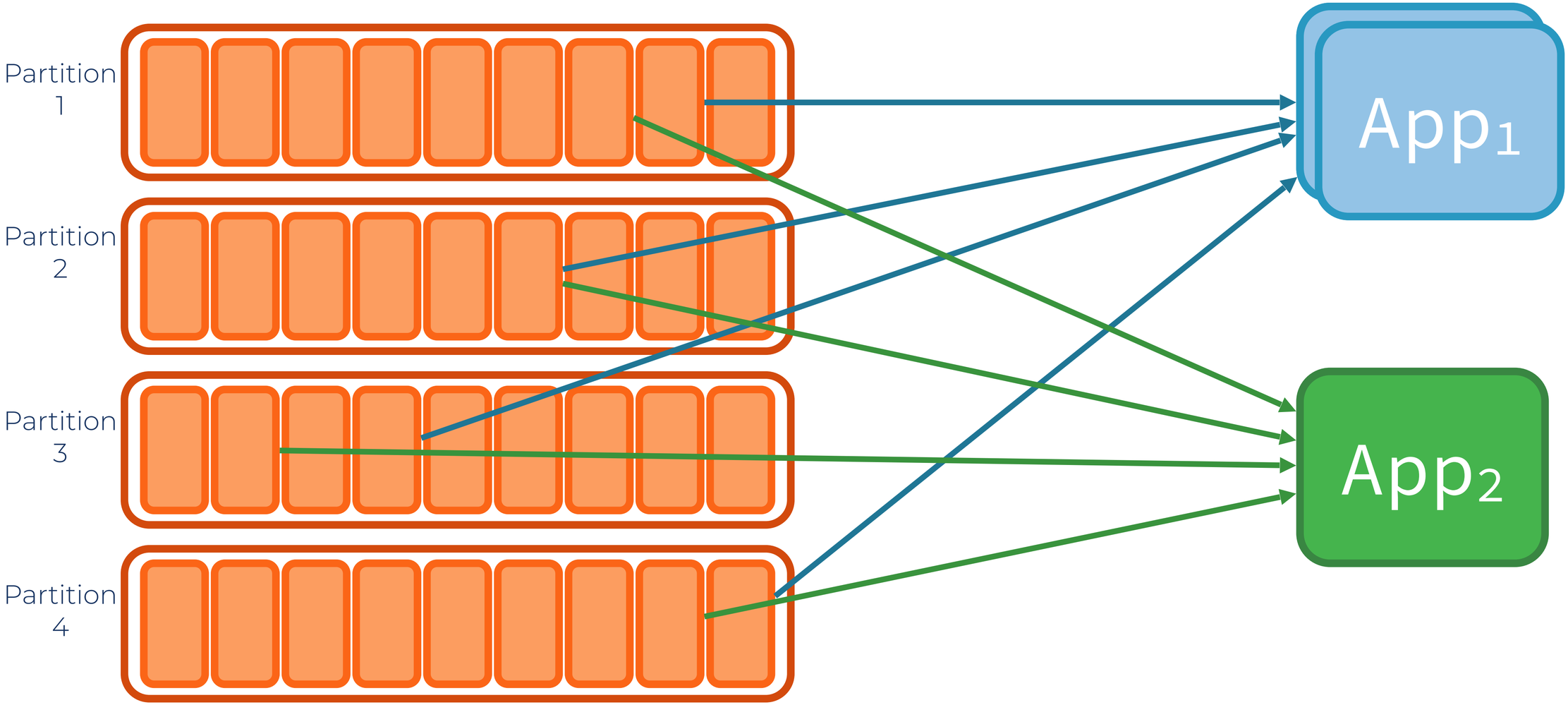
Consuming From Kafka - Single Consumer



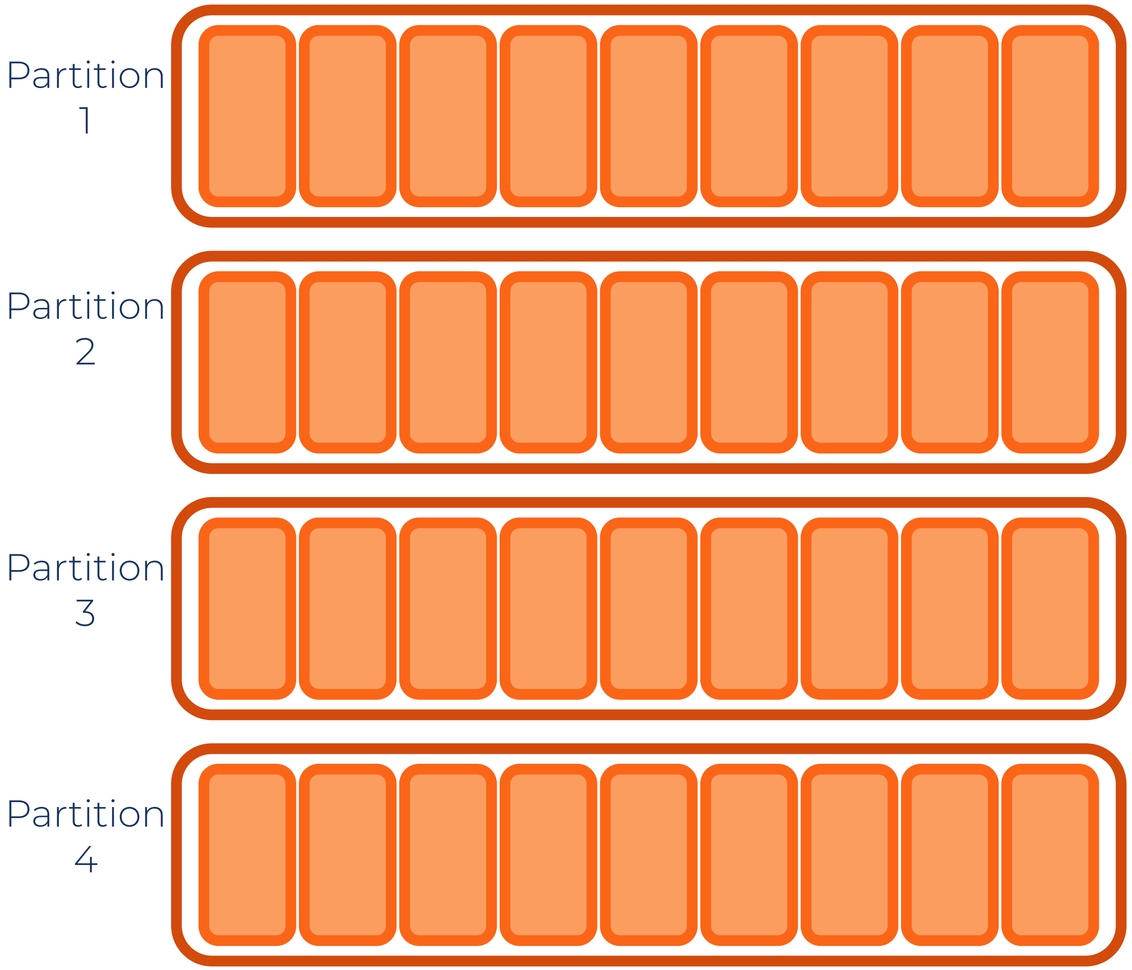
Consuming From Kafka - Multiple Consumers

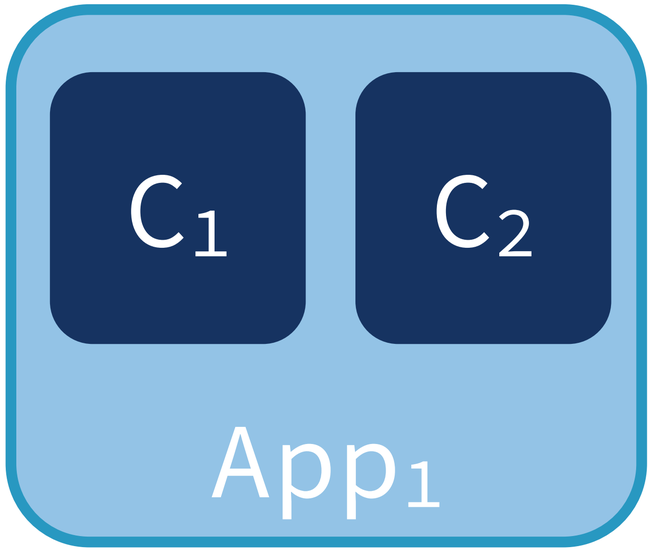


Consuming From Kafka - Grouped Consumers

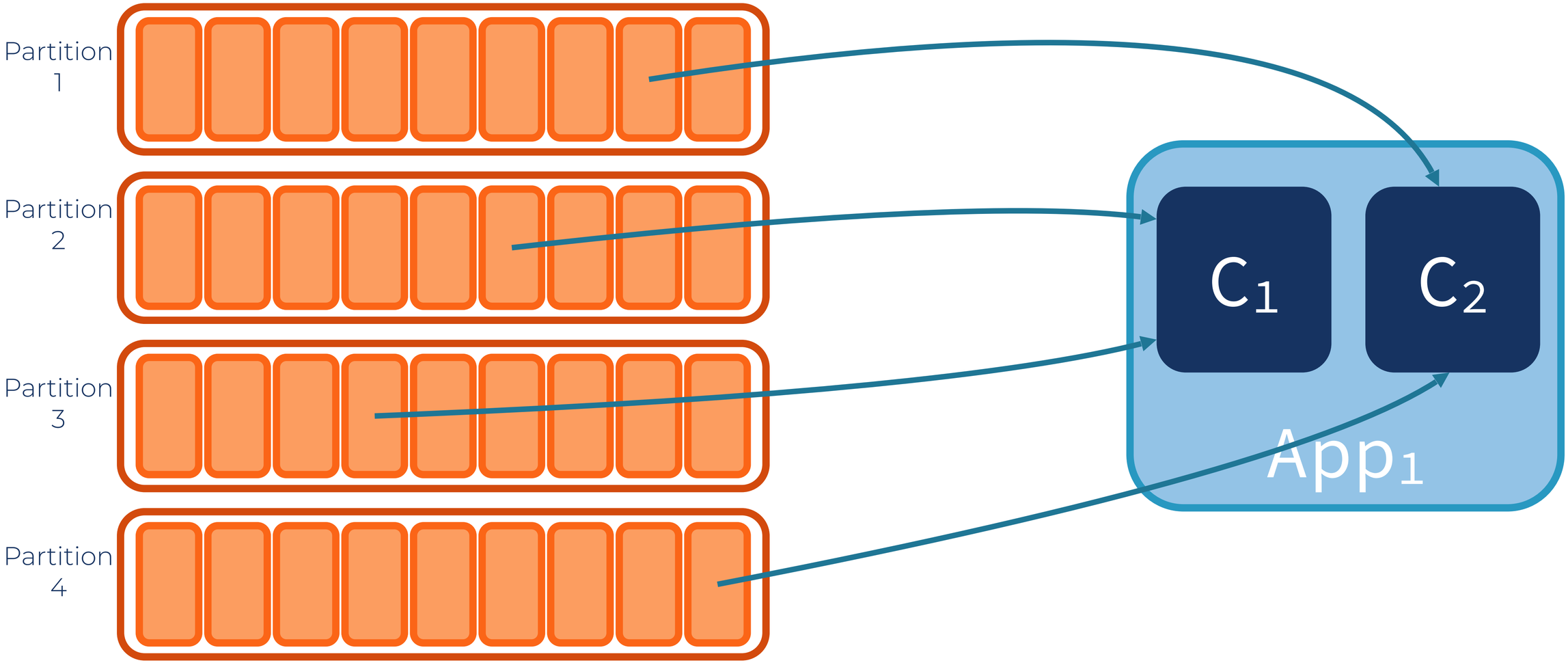


Consuming From Kafka - Grouped Consumers

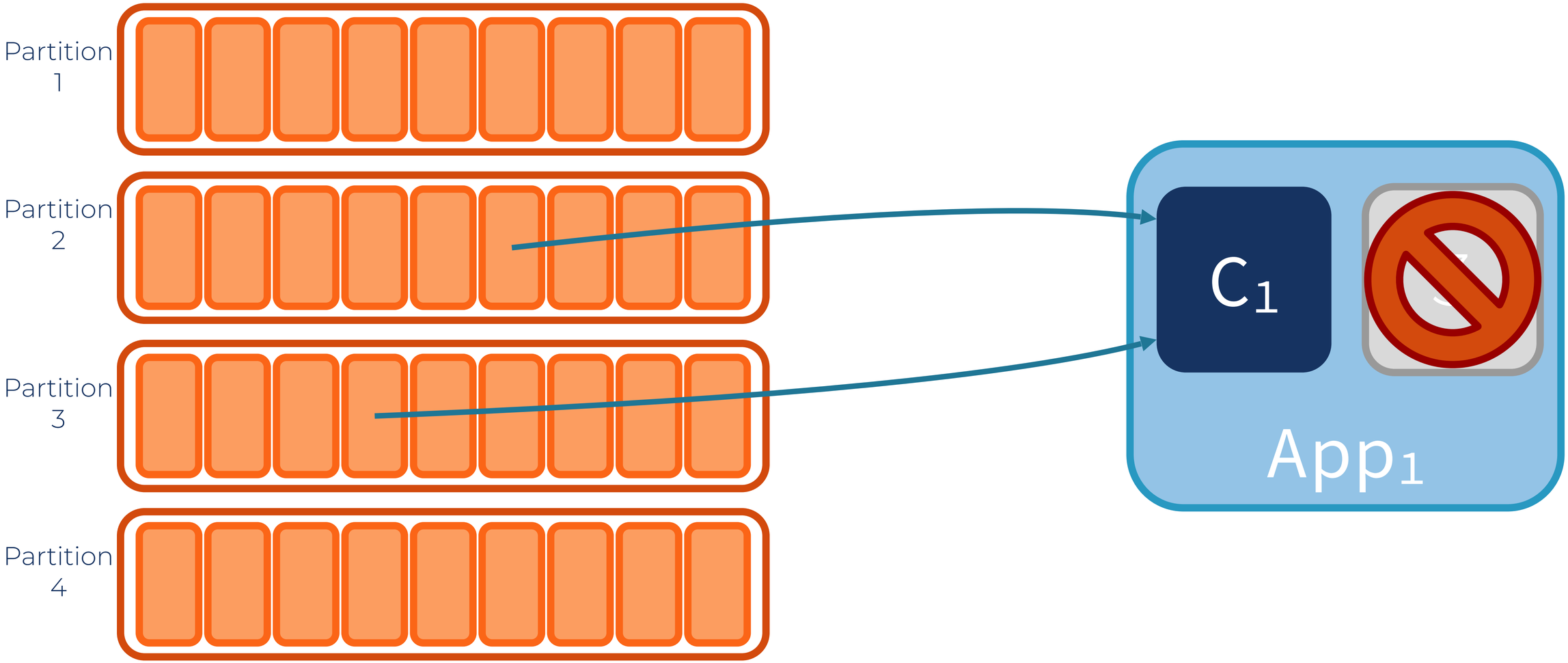




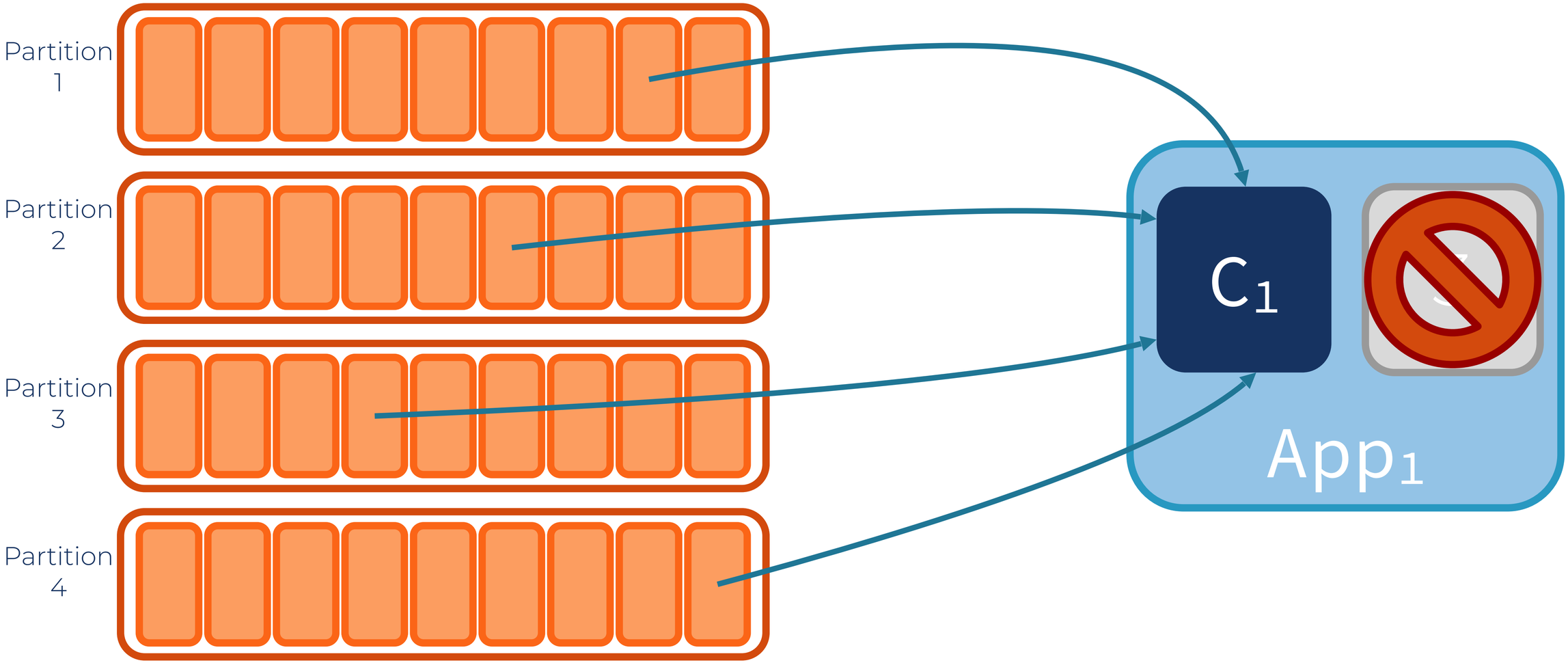
Consuming From Kafka - Grouped Consumers



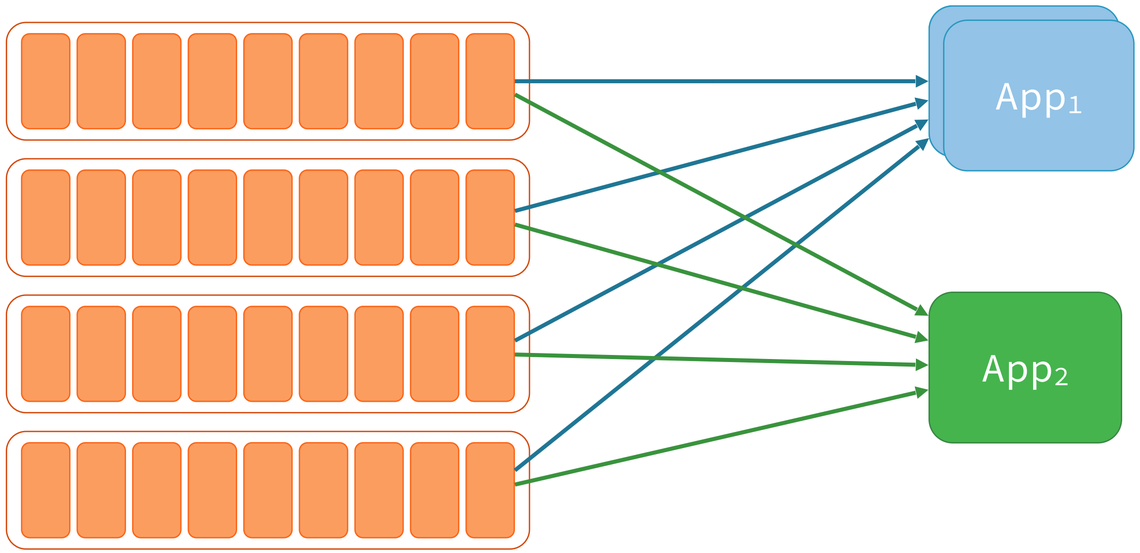
Consuming From Kafka - Grouped Consumers



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**
  Plus REST proxy if not

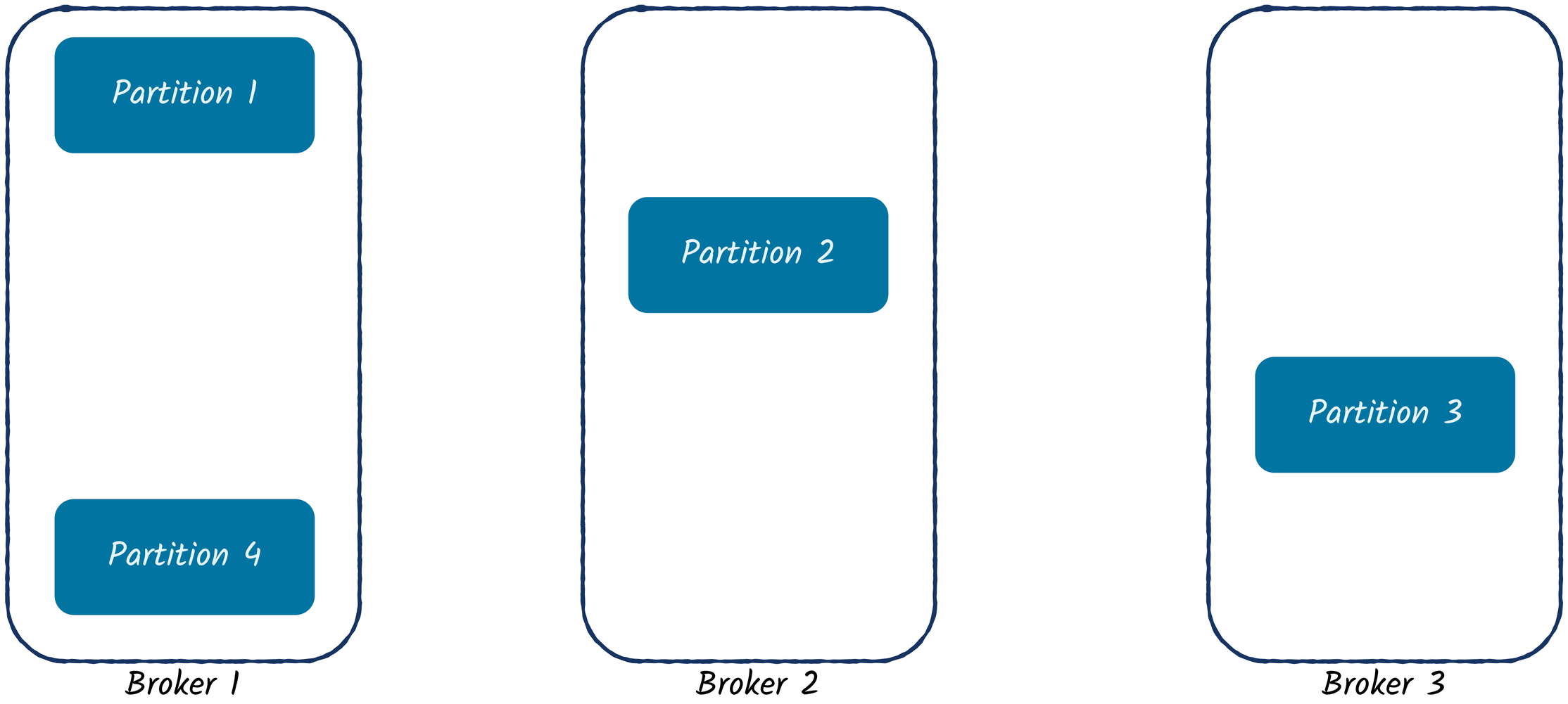


BROKERS

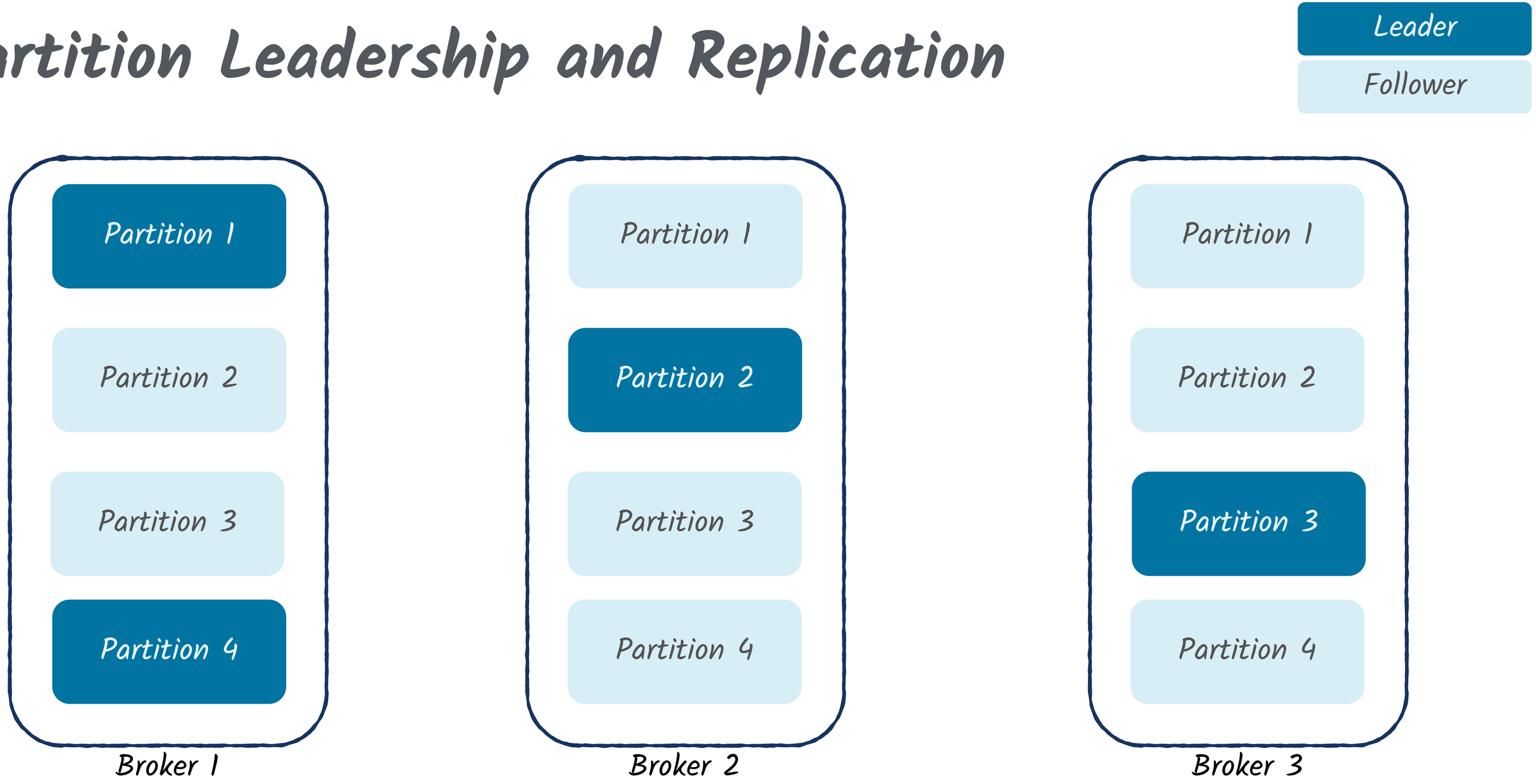
and REPLICATION

@rmoff

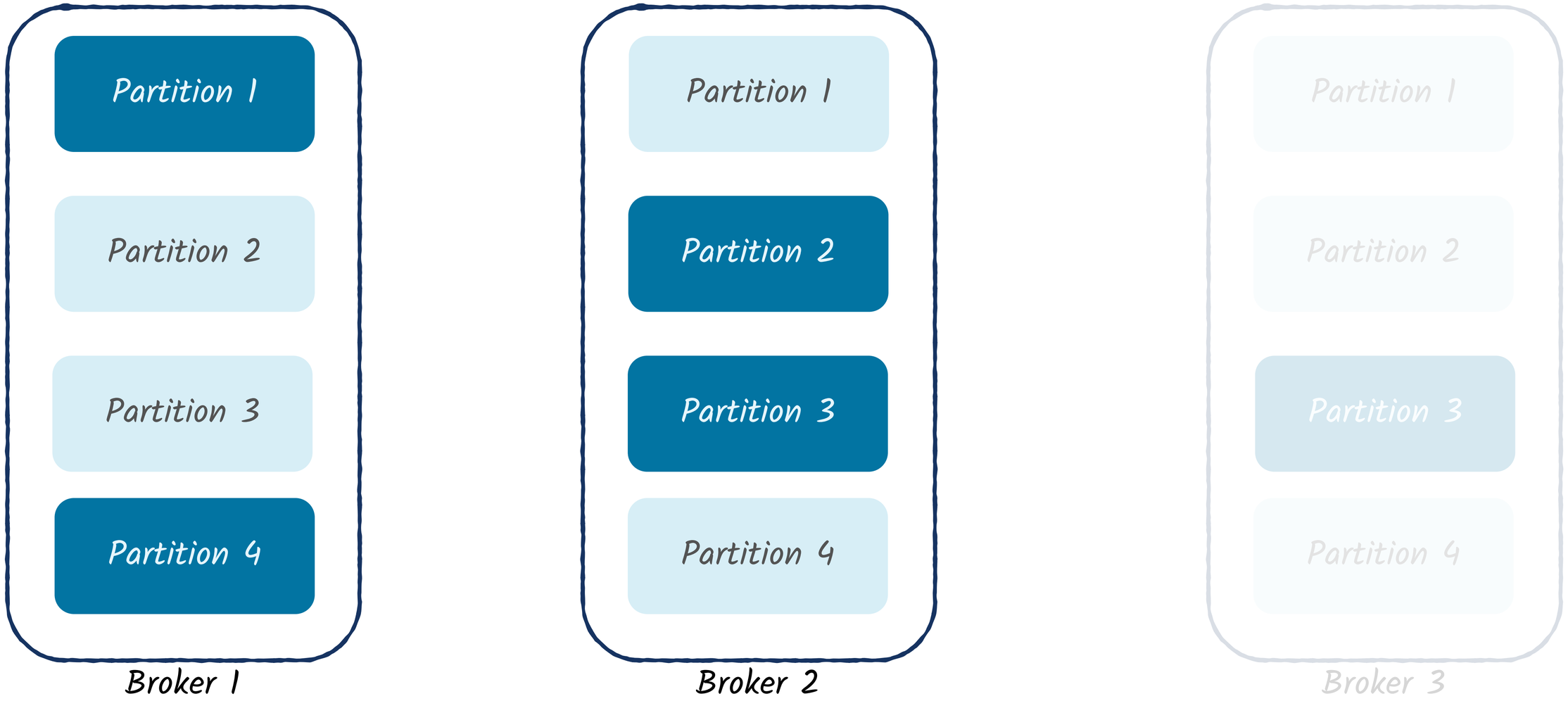
Partition Leadership and Replication



Partition Leadership and Replication



Partition Leadership and Replication





So far, this is

Pretty good

@rmoff



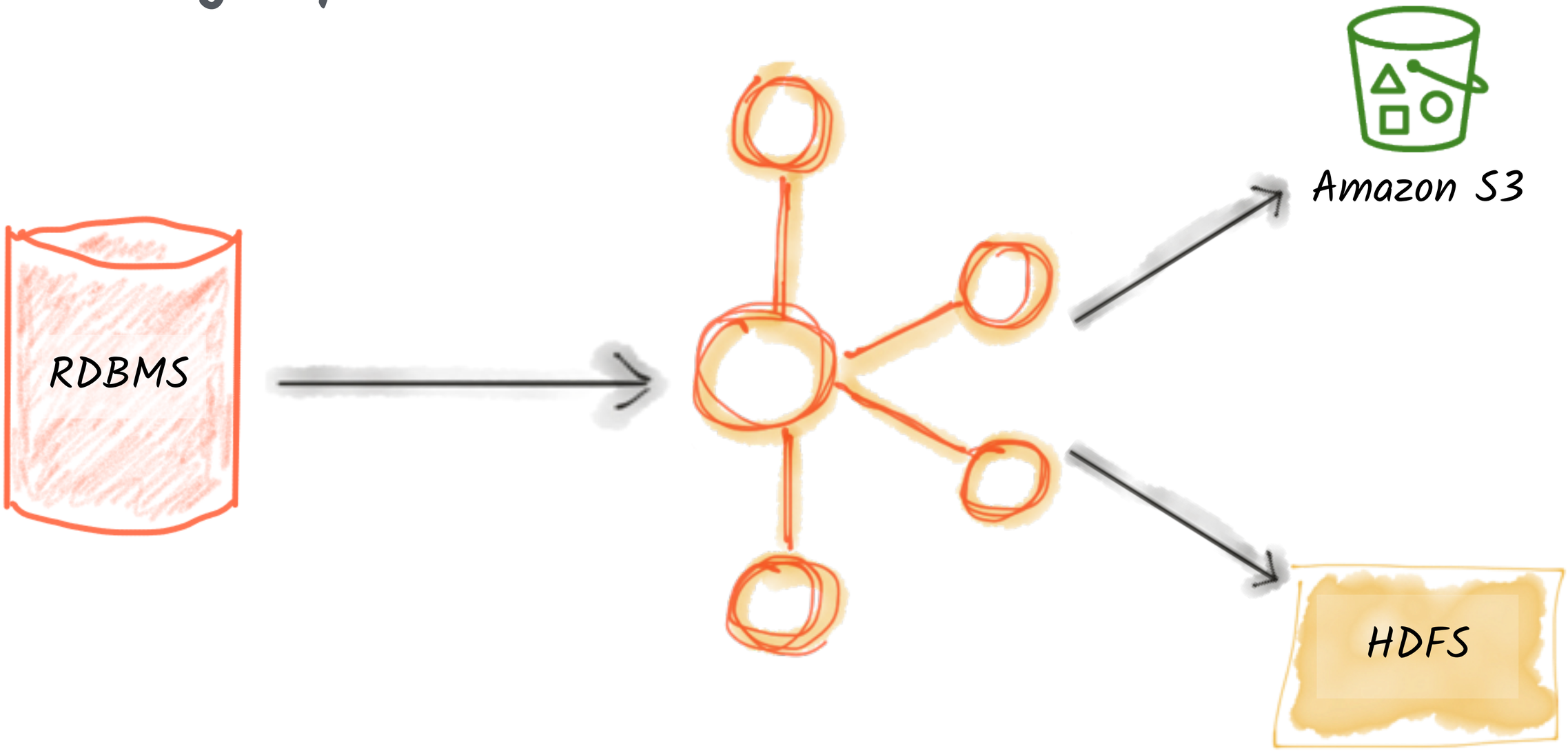
So far, this is

Pretty good

but I've not finished yet...

@rmoff

Streaming Pipelines

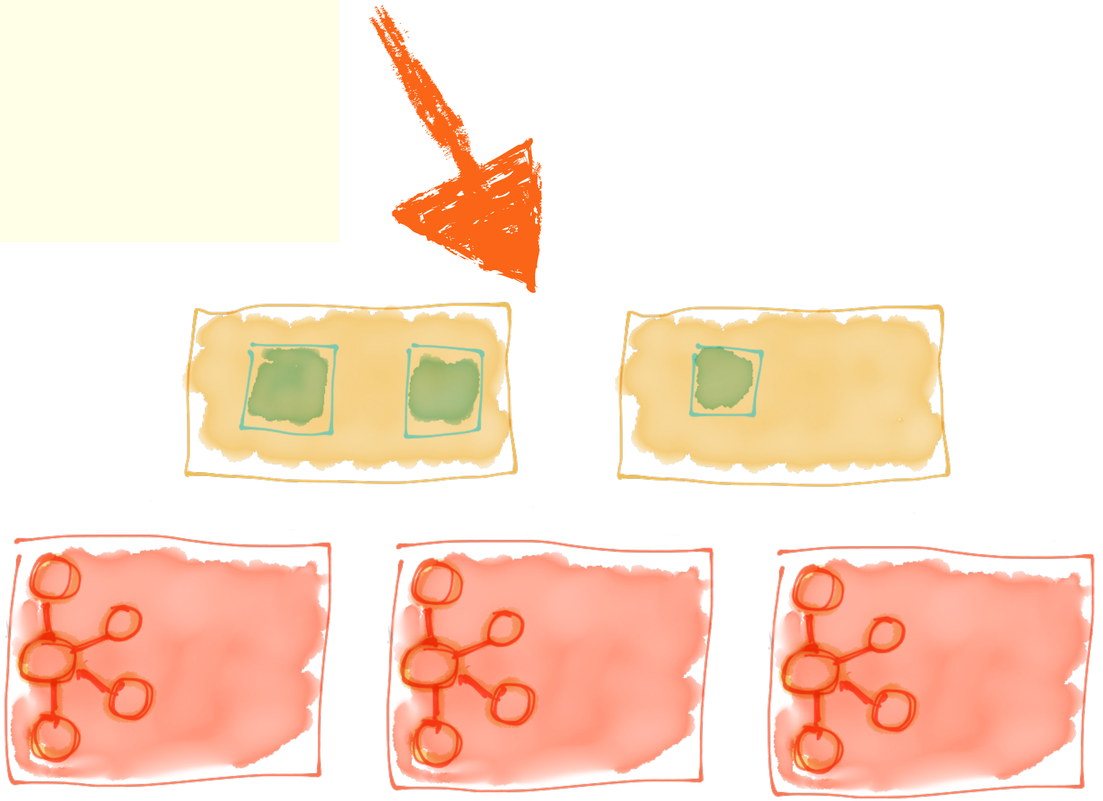




Streaming Integration with Kafka Connect



Sources



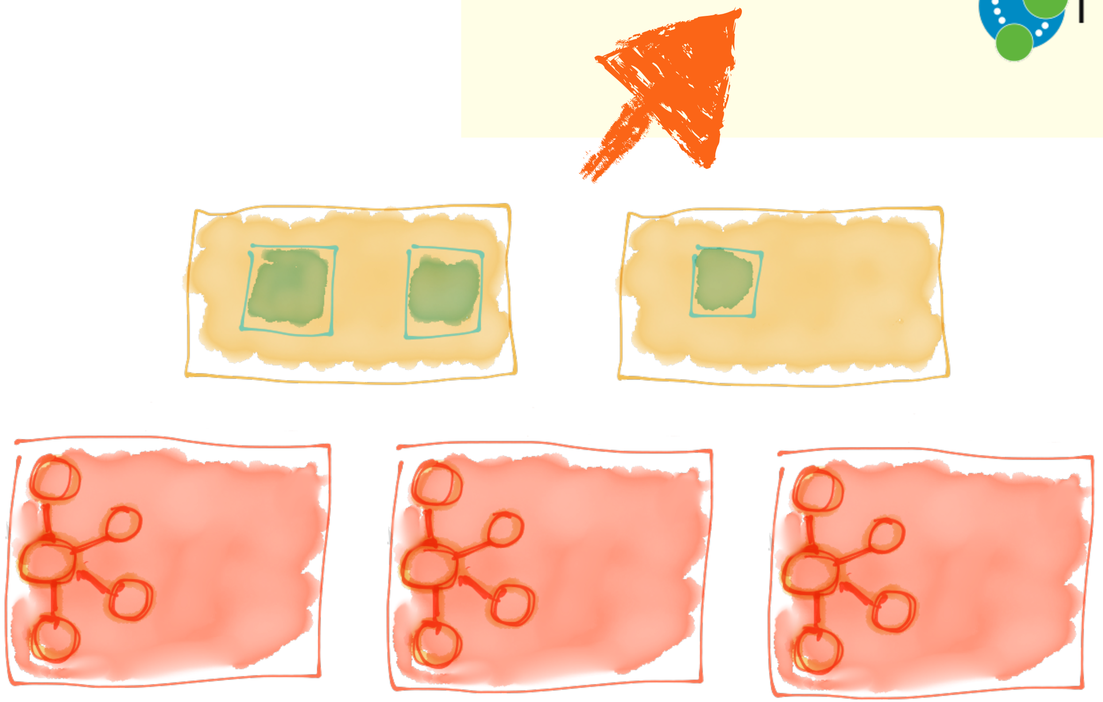
Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

Sinks





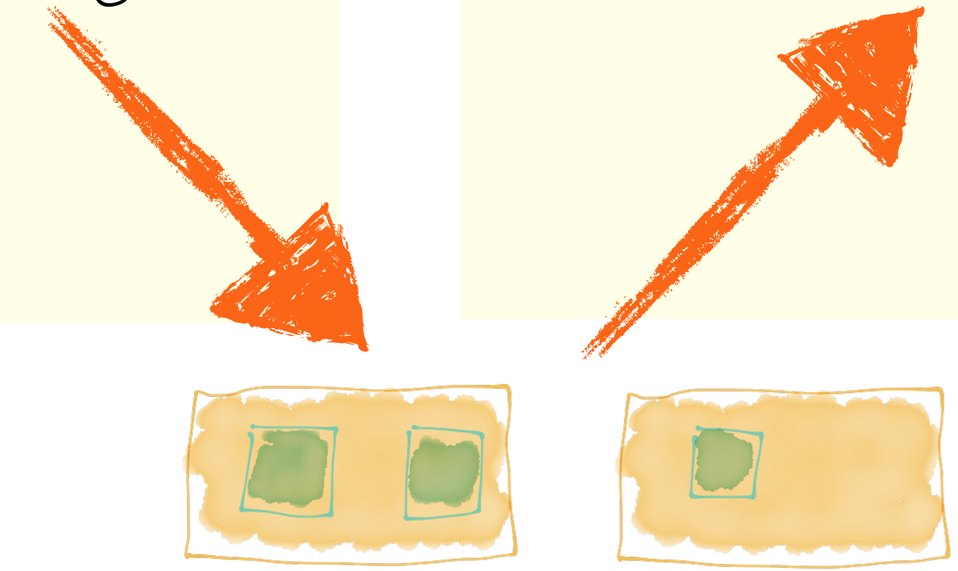
Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect







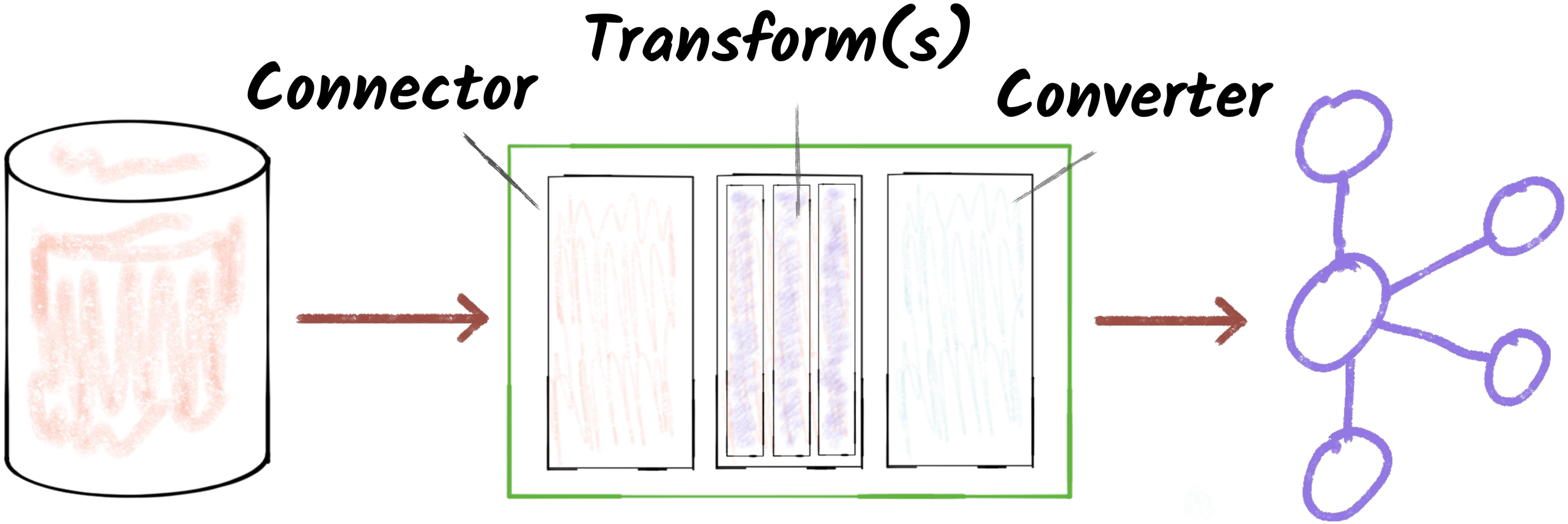
Kafka Connect

Kafka Brokers

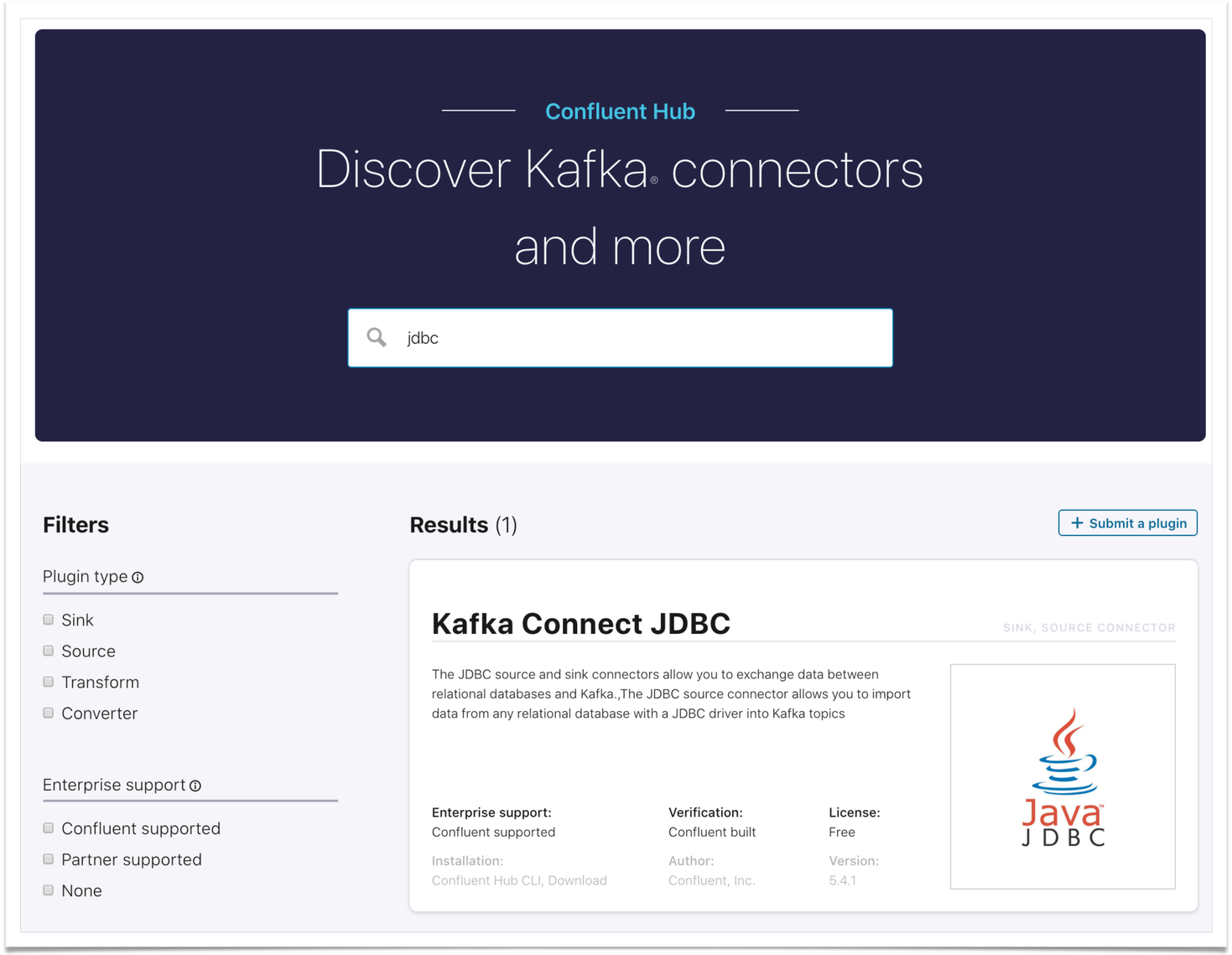
Look Ma, No Code!

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

Extensible



hub.confluent.io

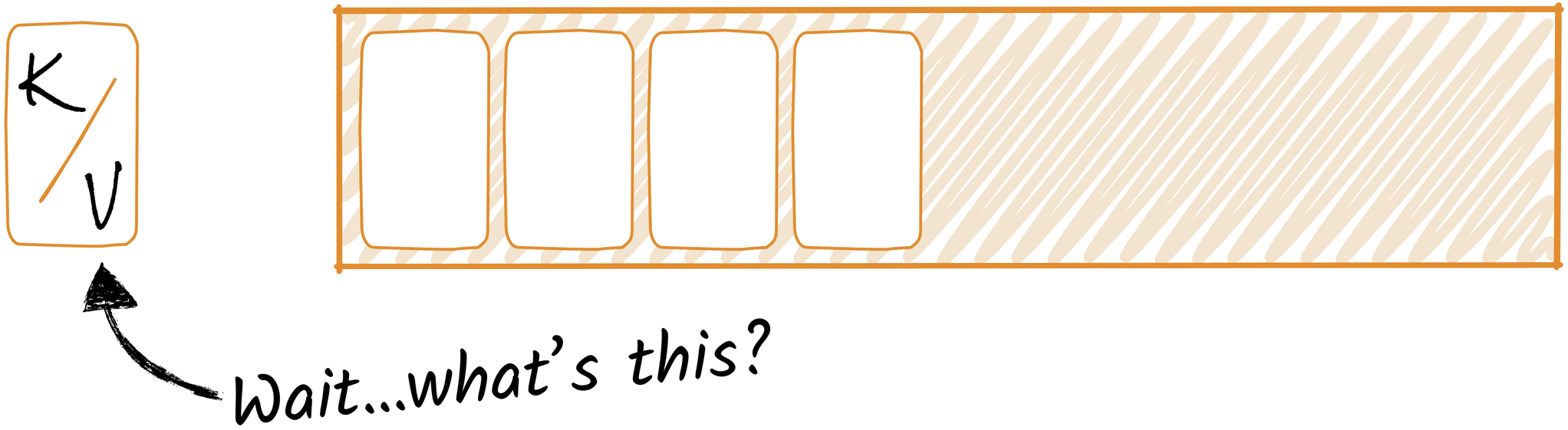



K
V


K
V


K
V


K
V


K
V
Wait...what's this?

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

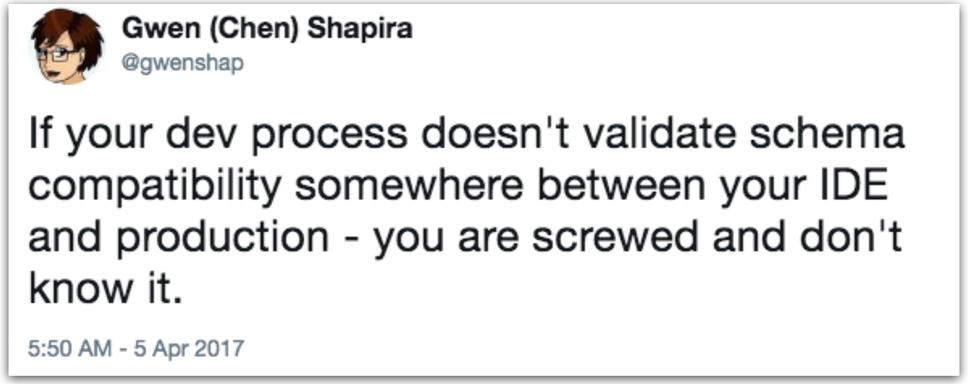
Avro 👍

Protobuf 👍

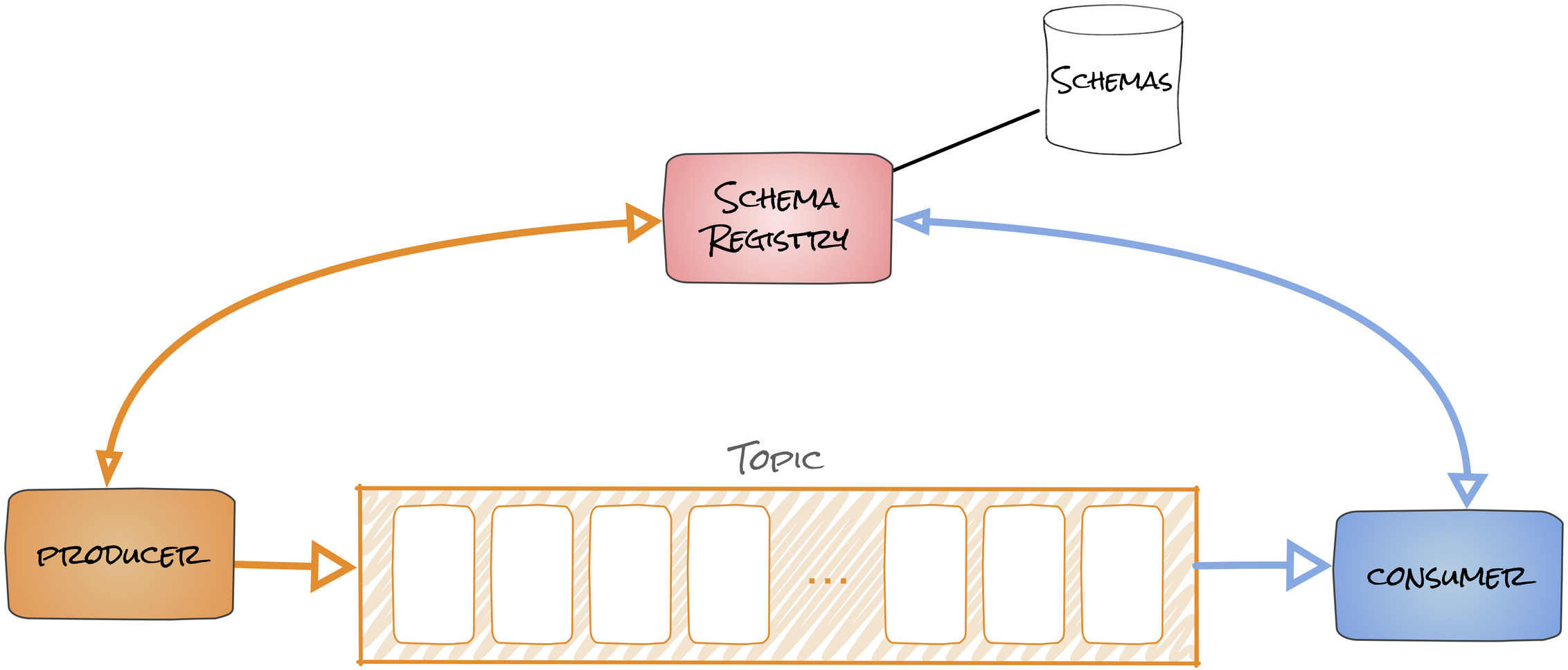
JSON Schema 👍

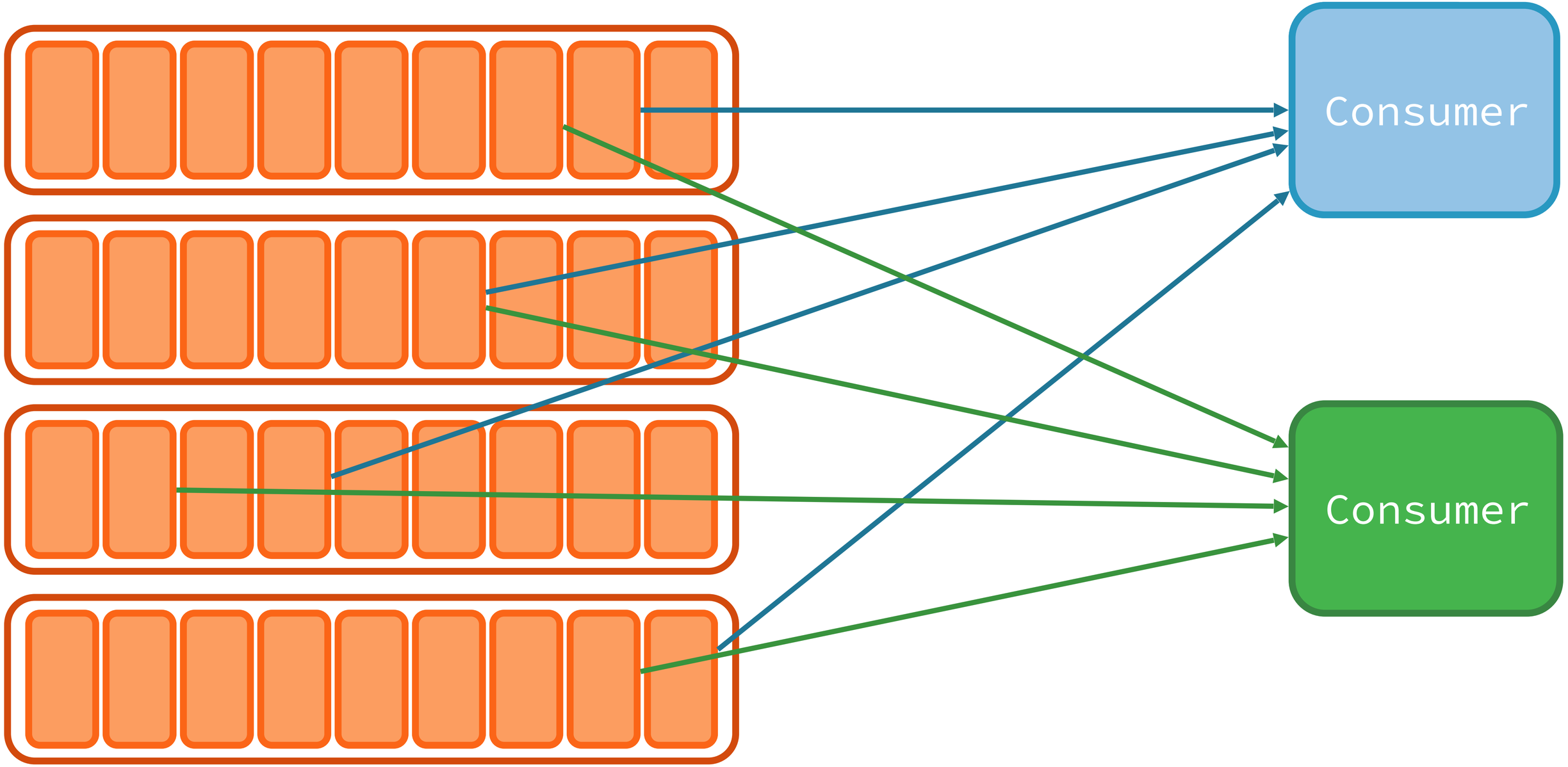
JSON 😬

CSV 



https://rmoff.dev/qcon-schemas


SCHEMAS
SCHEMA REGISTRY
TOPIC
PRODUCER
CONSUMER


Consumer
Consumer

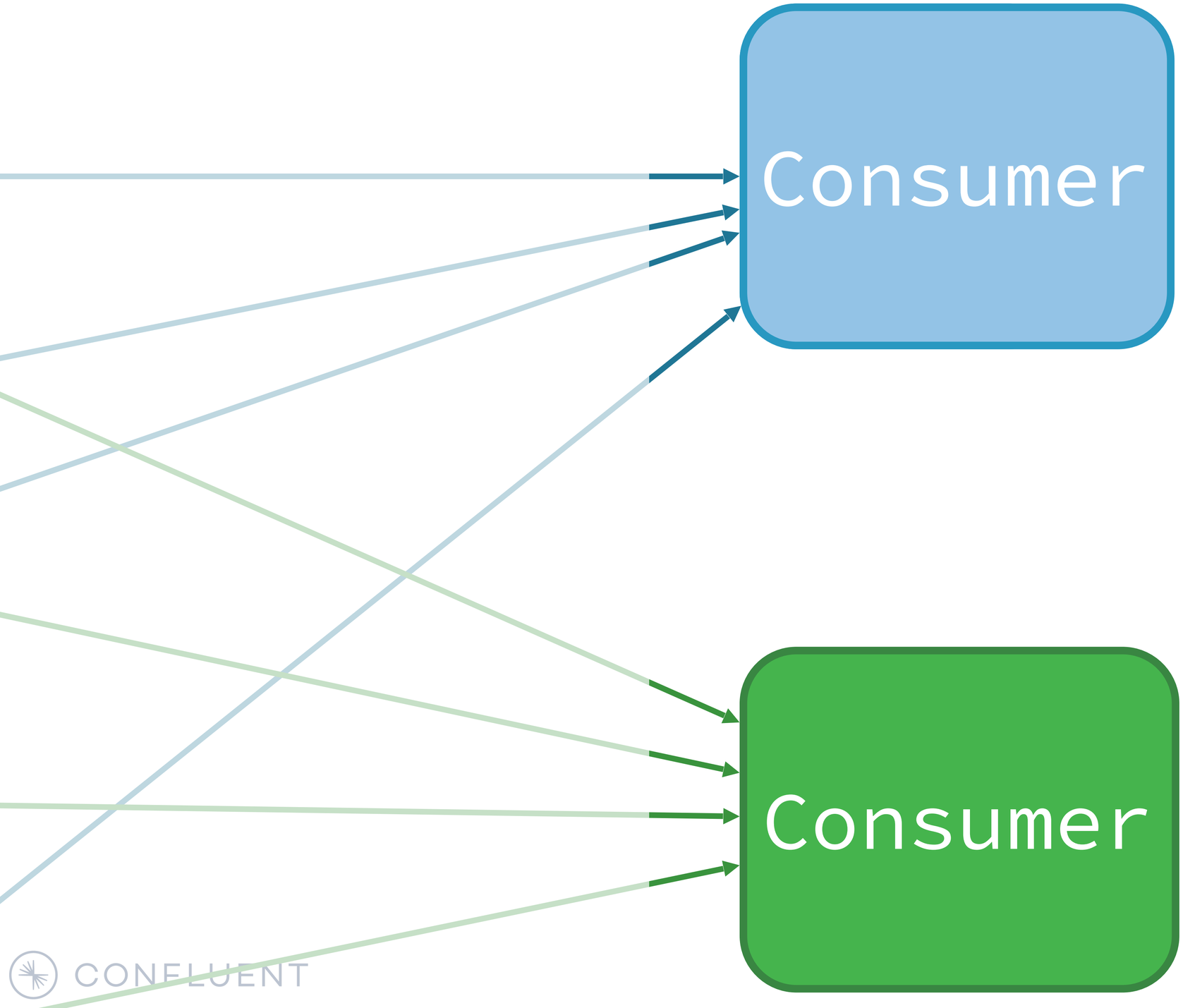
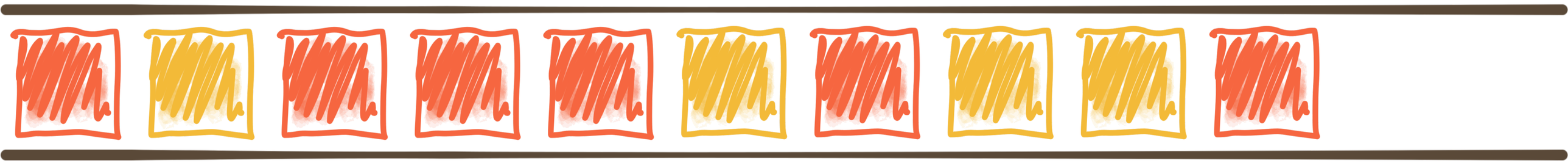

Consumer
Consumer



Photo by Franck V. on Unsplash

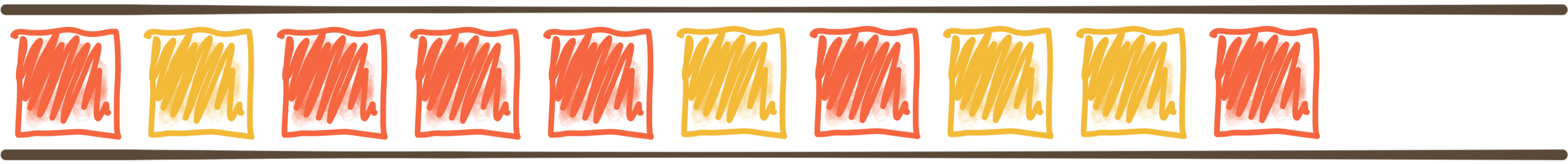
Streams of events



Time



Stream Processing



Stream: widgets



Stream: widgets_red

Stream Processing with Kafka Streams

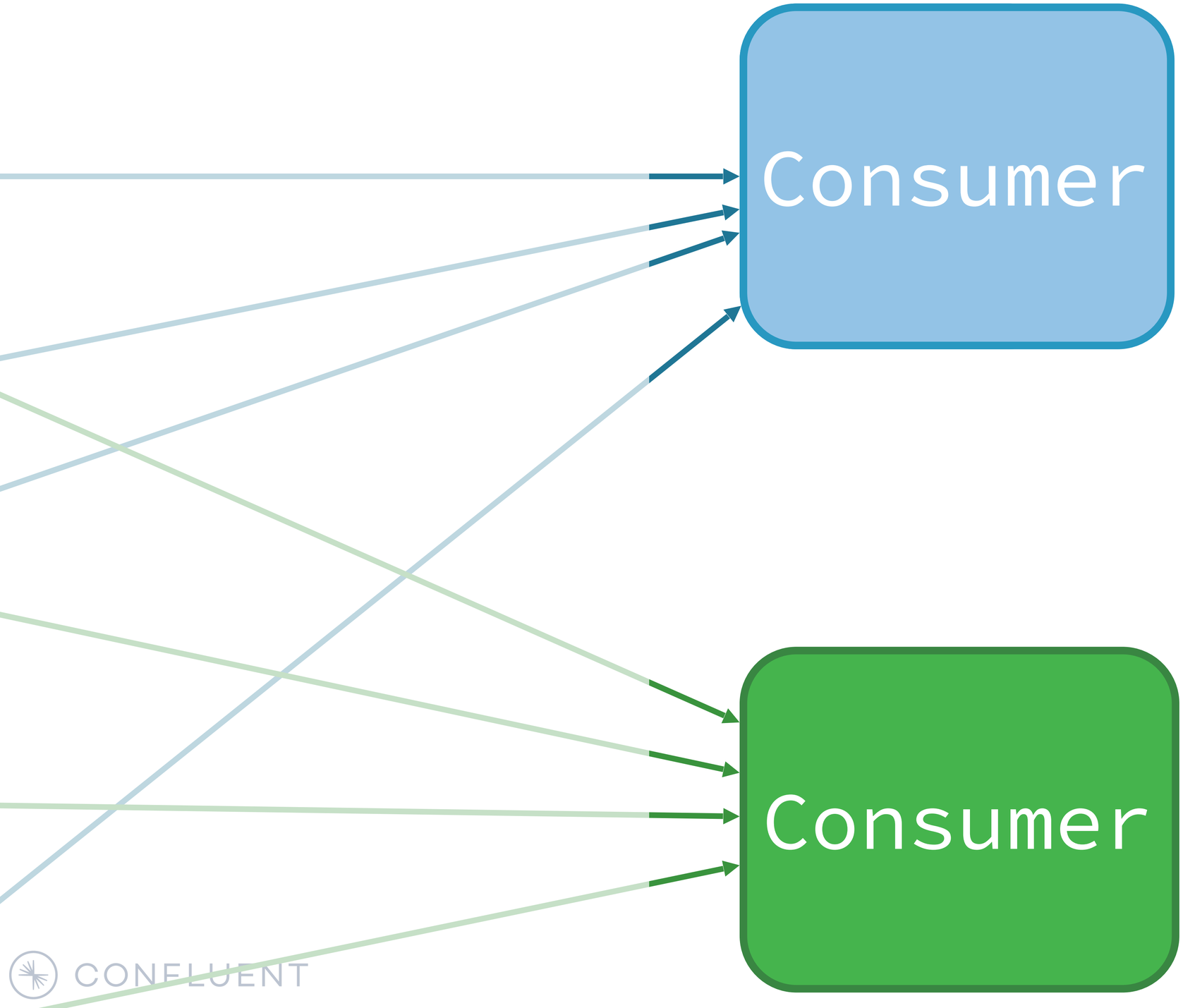


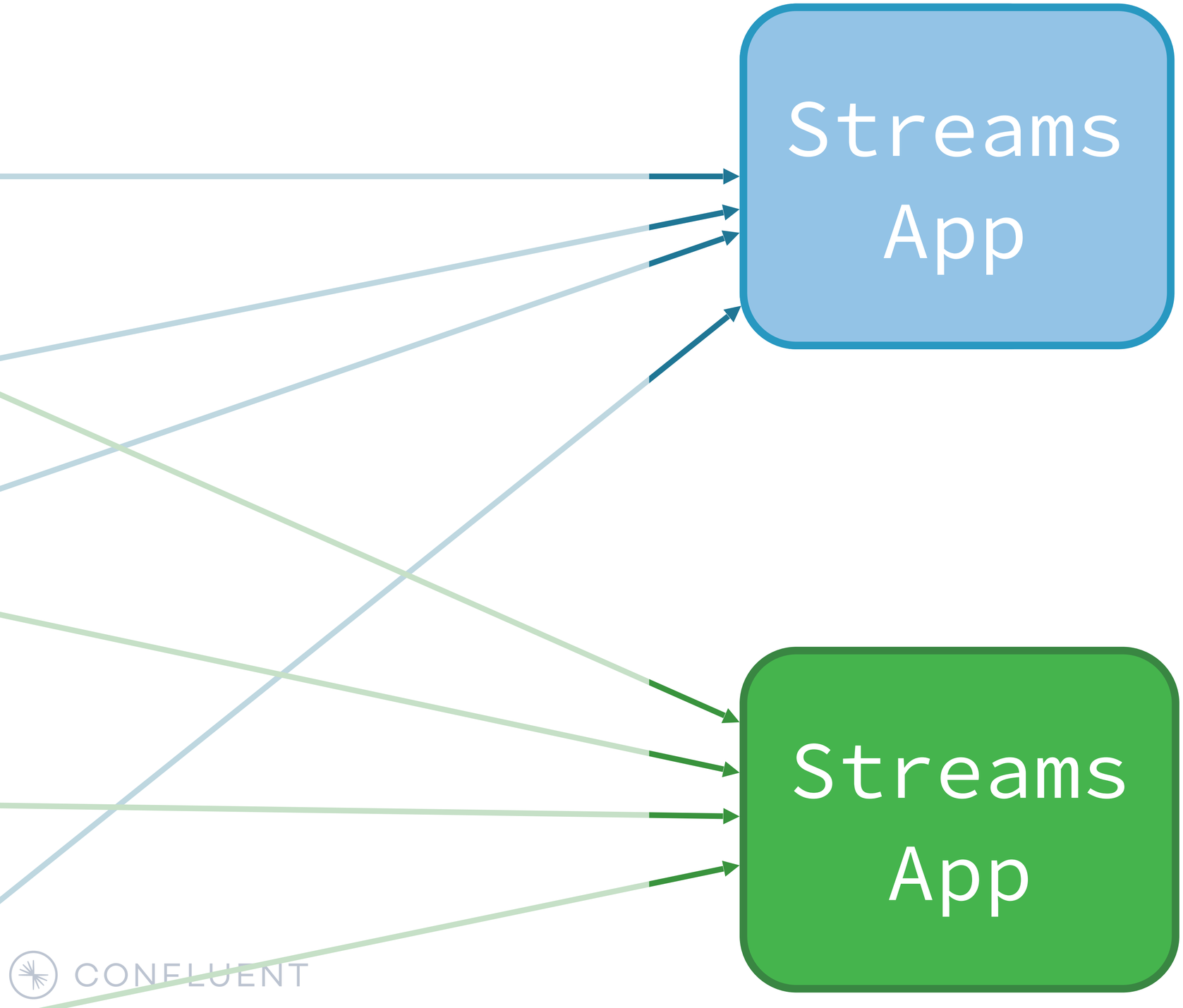
Stream:
widgets

```
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));
```

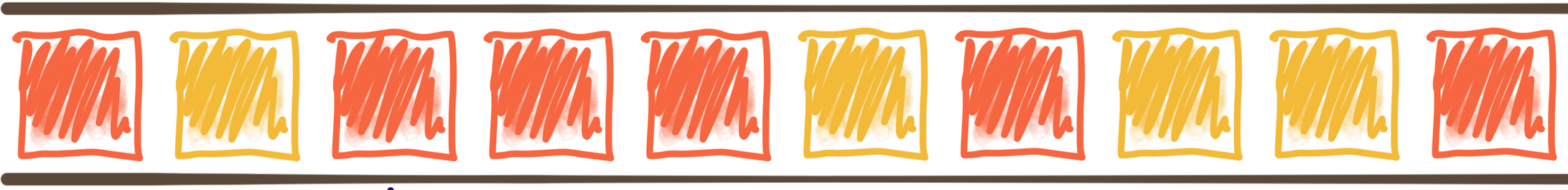


Stream:
widgets_red


Consumer
Consumer


Streams App
Streams App

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

CONFLUENT

ksqlDB

or

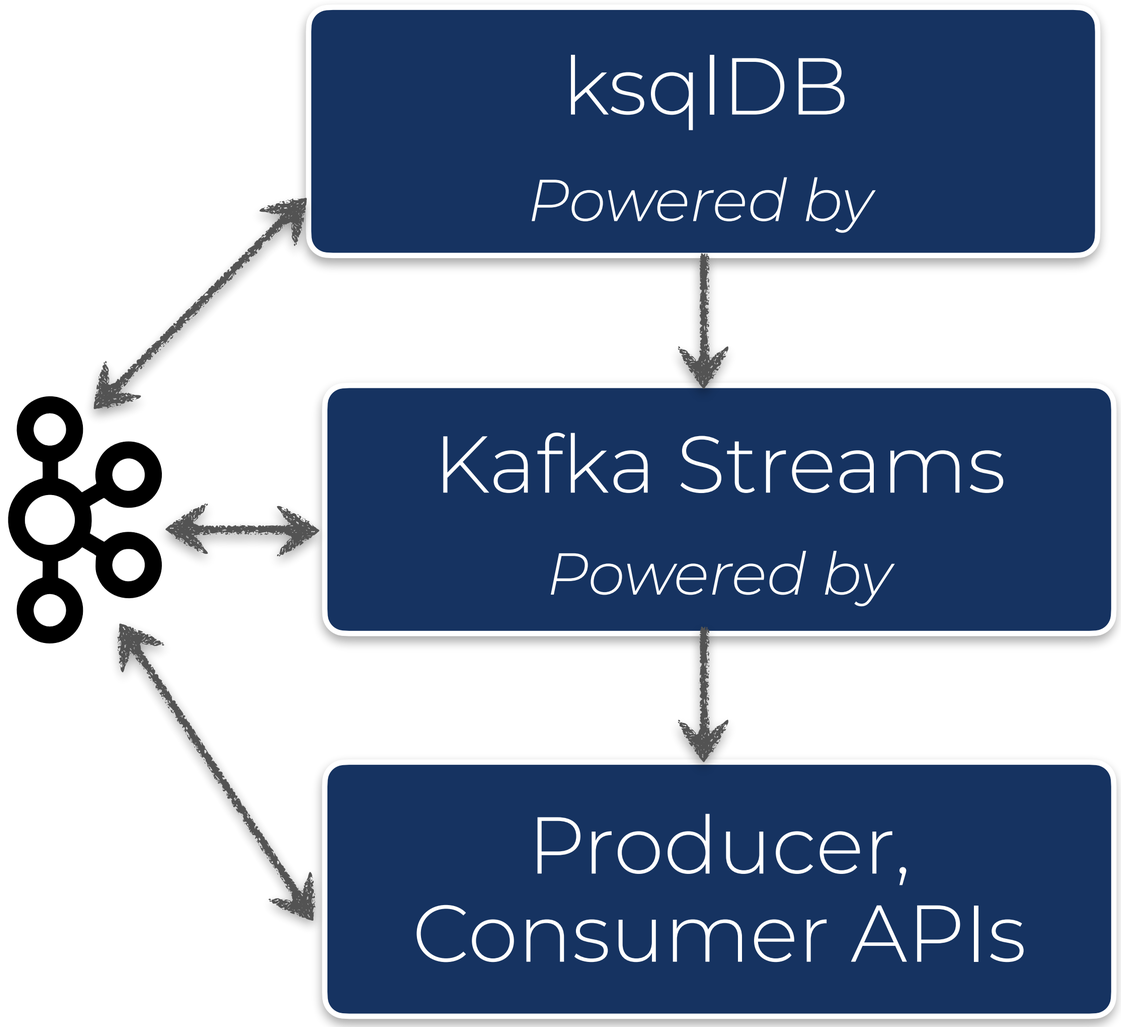
Kafka Streams?

@rmoff



Photo by Ramiz Dedaković on Unsplas

Standing on the Shoulders of Streaming Giants



Ease of use

Flexibility

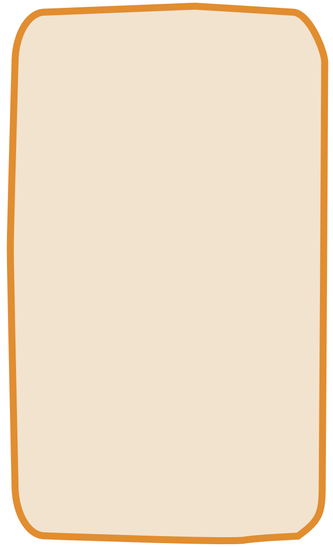
```
CREATE STREAM,
CREATE TABLE, SELECT, JOIN,
GROUP BY, SUM, …
```

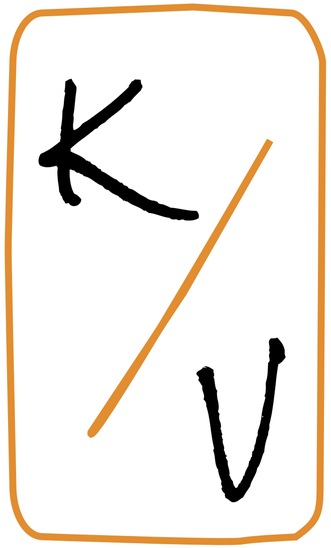
ksqlDB UDFs

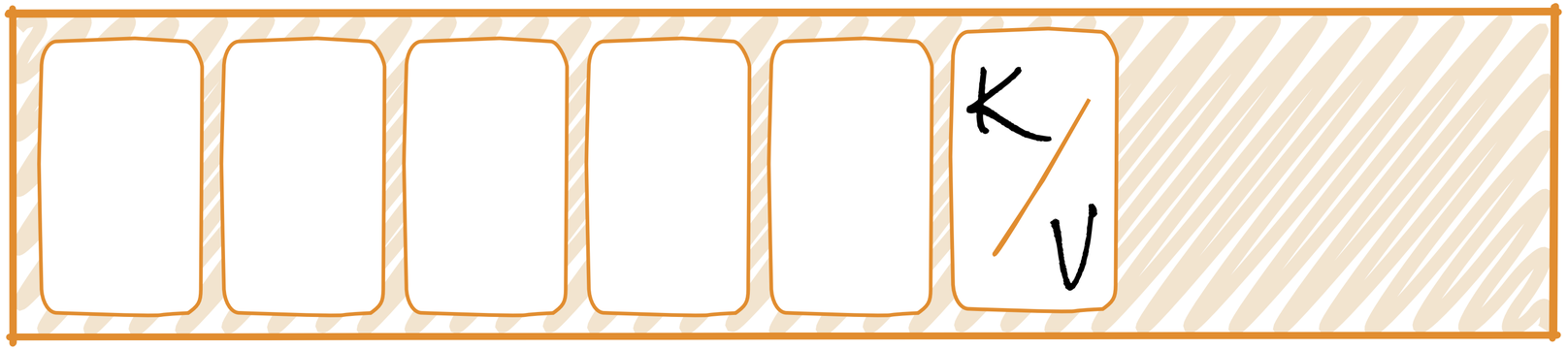
```
KStream<>, KTable<>, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …
```

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

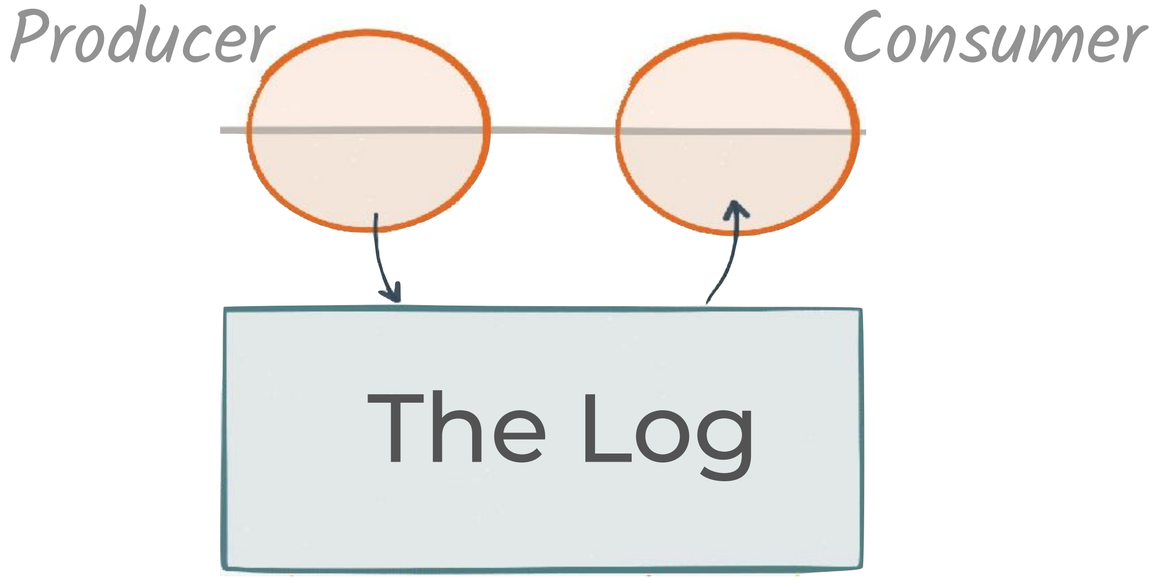
Summary

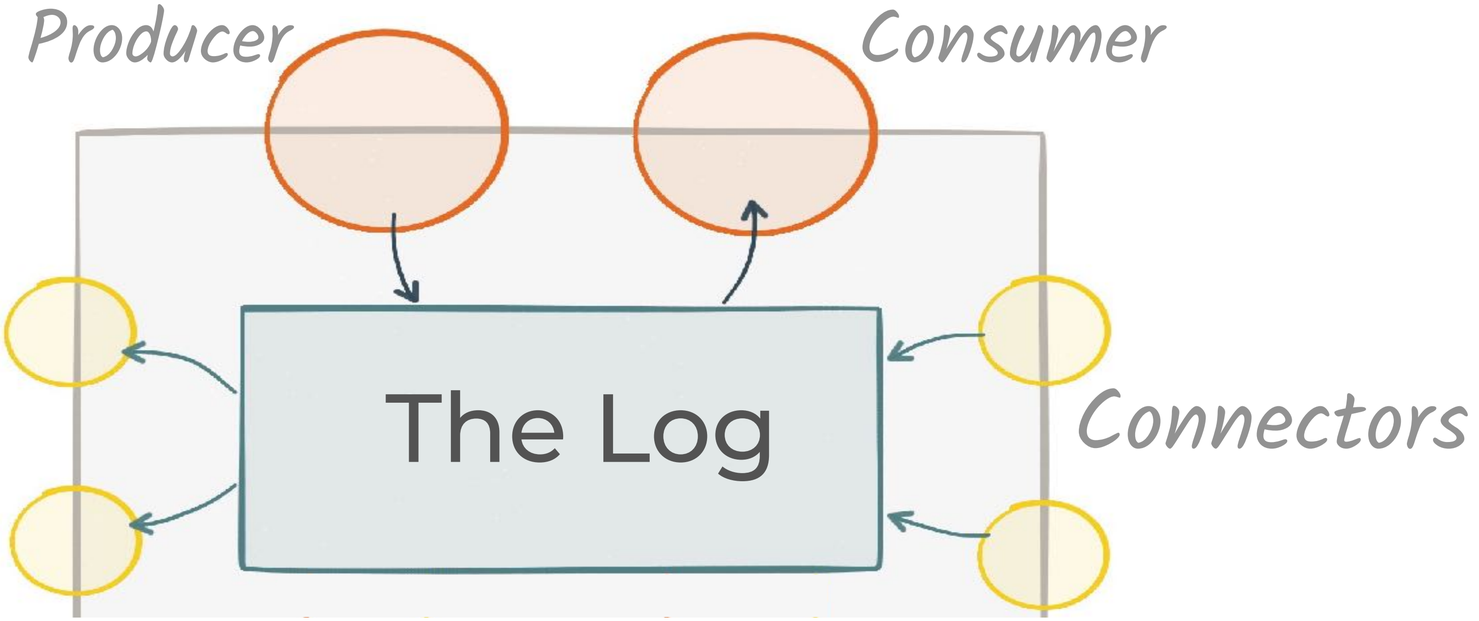


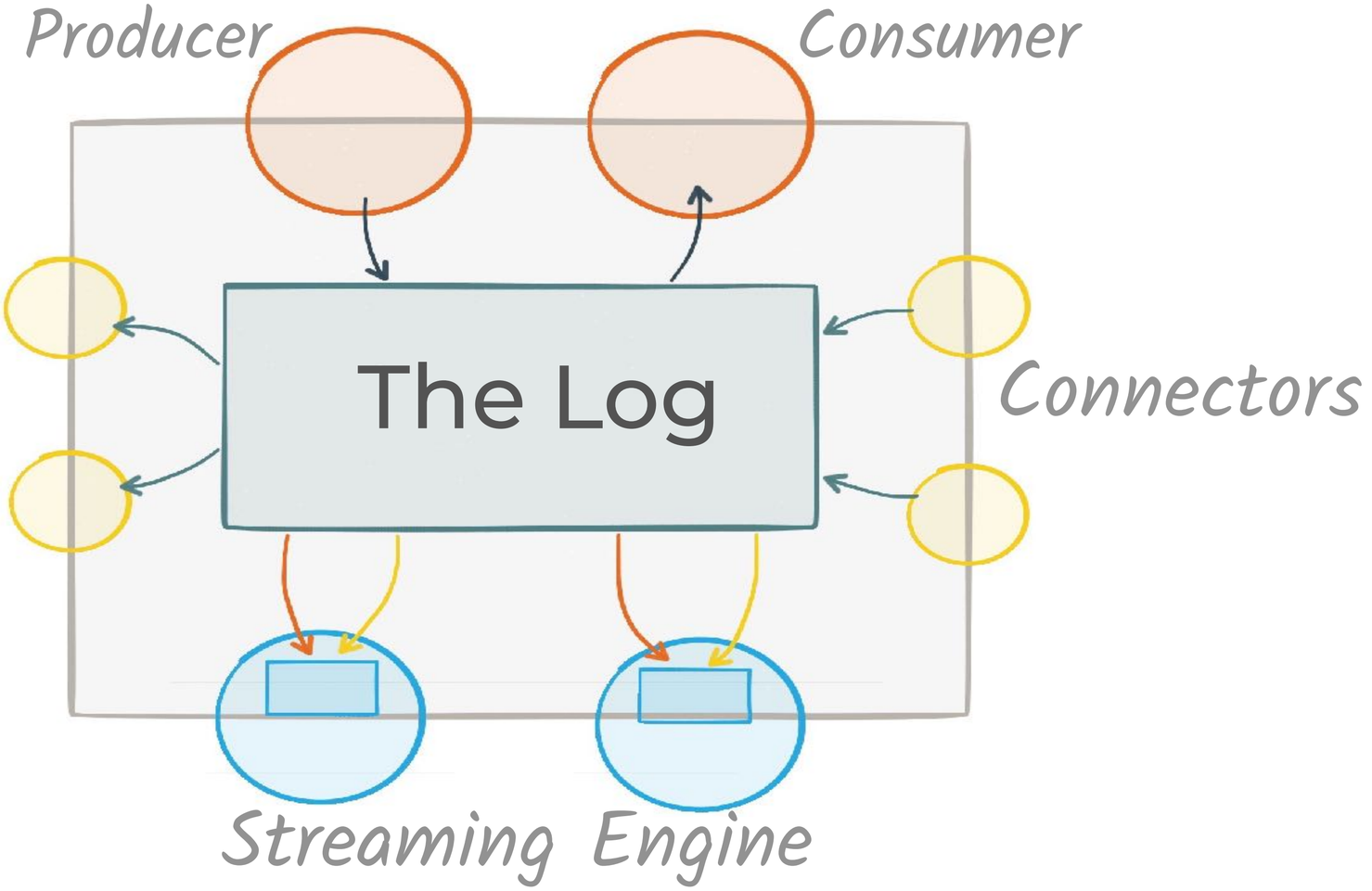

K
V


K
V

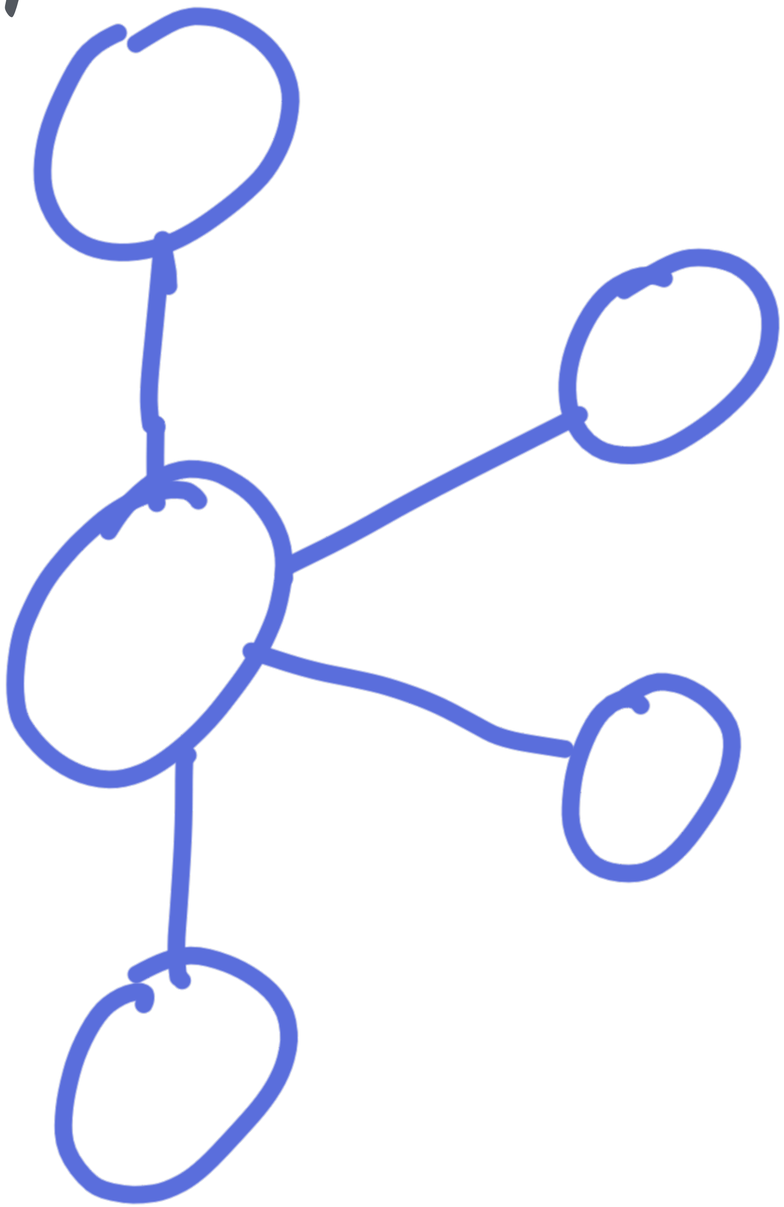
The Log

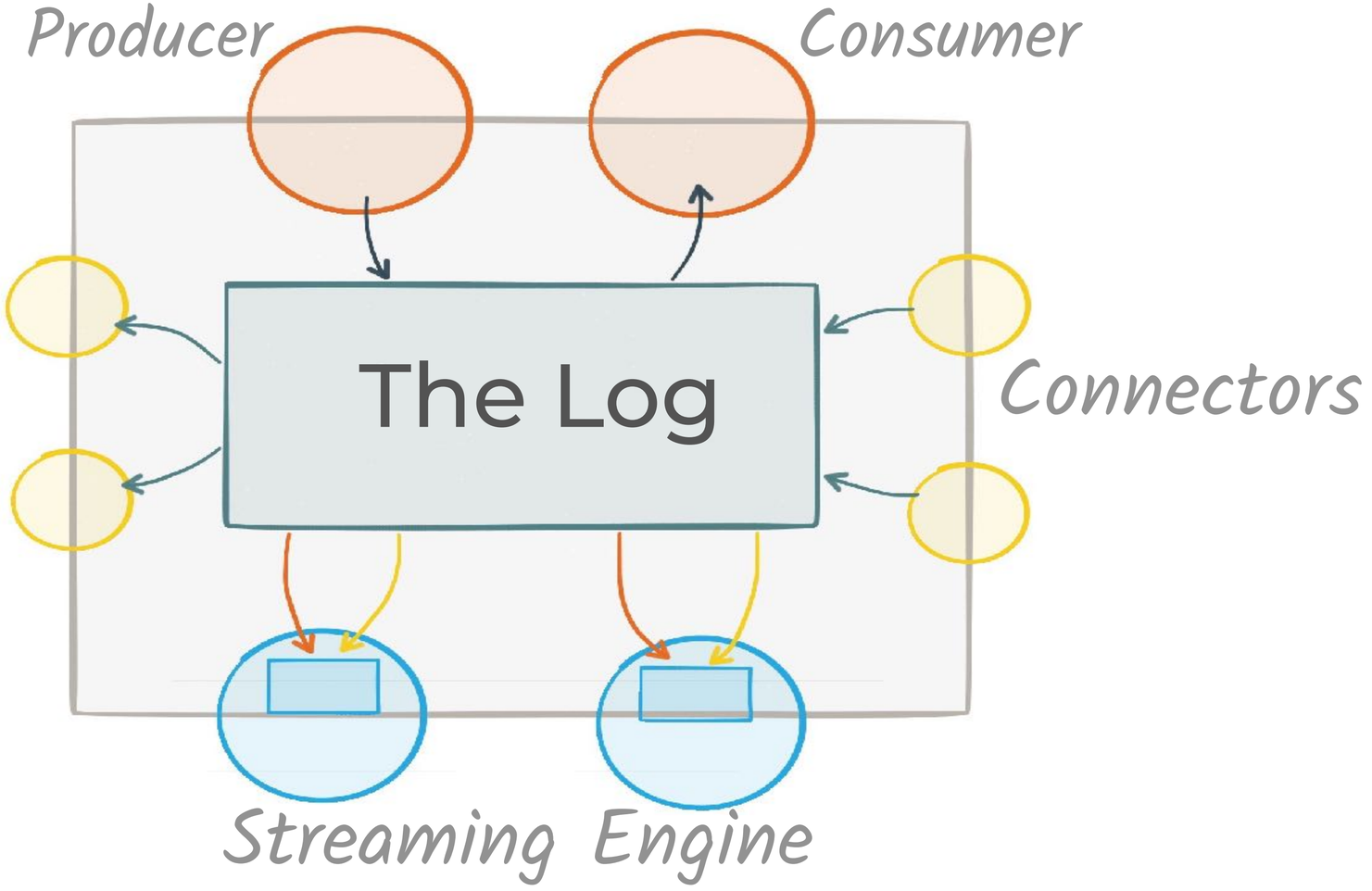

Producer
Consumer
The Log


Producer
Consumer
The Log
Connectors

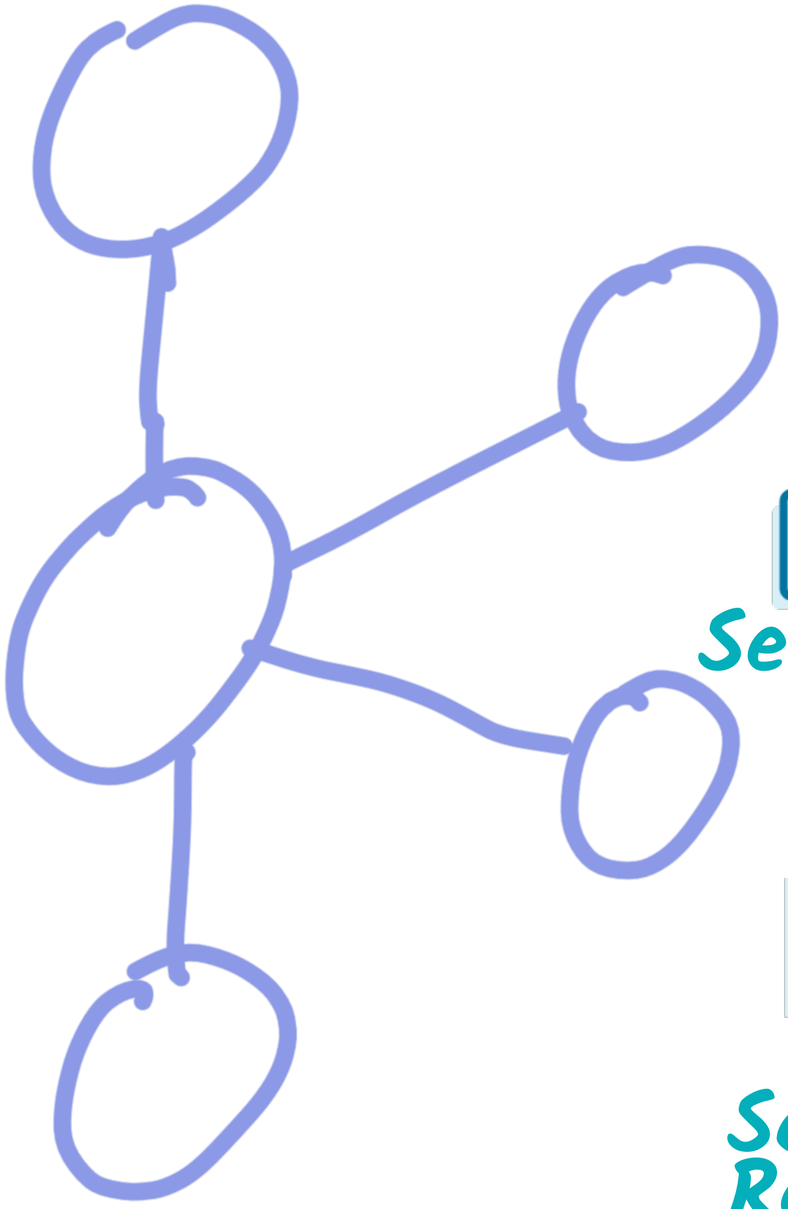

Producer
Consumer
The Log
Connectors
Streaming Engine

Apache Kafka





CONFLUENT





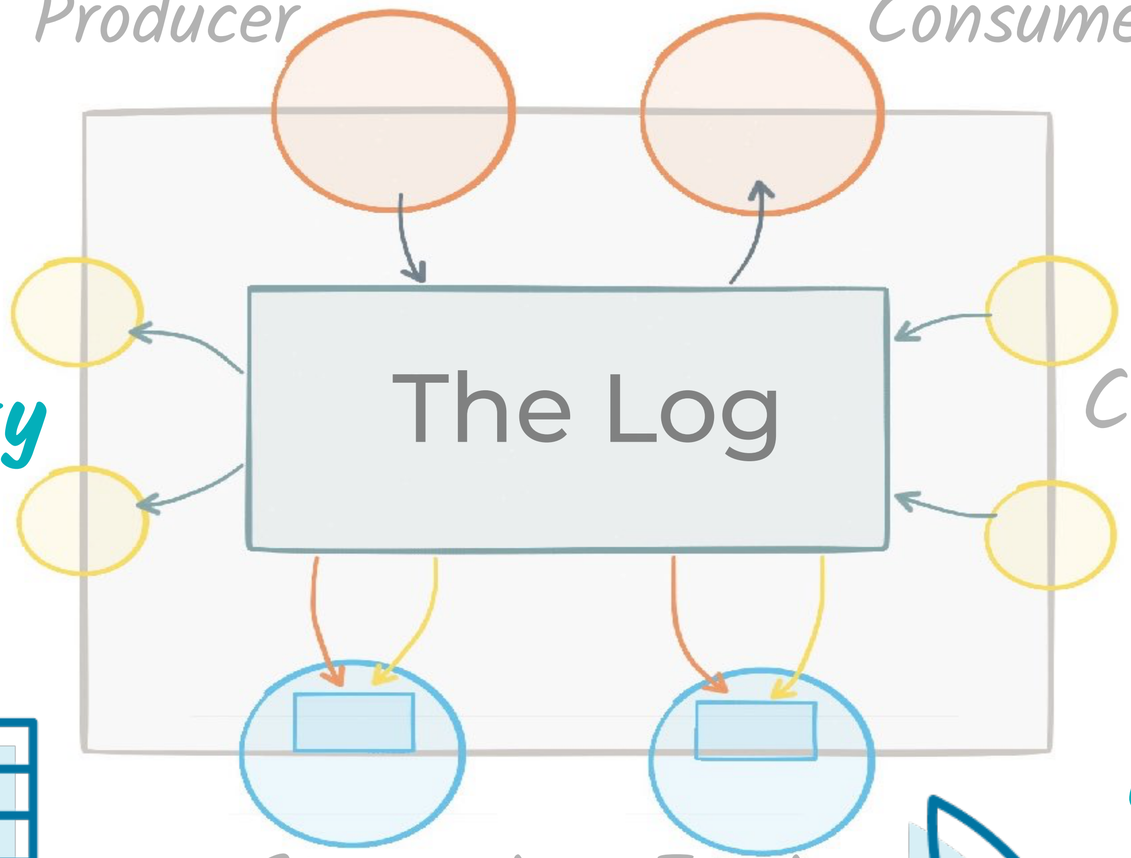
Security

Producer

Consumer



REST Proxy



Connectors



Schema Registry

Streaming Engine



Confluent Control Center



ksqlDB

RMOFF200

Free money!
(additional $200 towards your bill 😄)



confluent cloud

Fully Managed Kafka as a Service

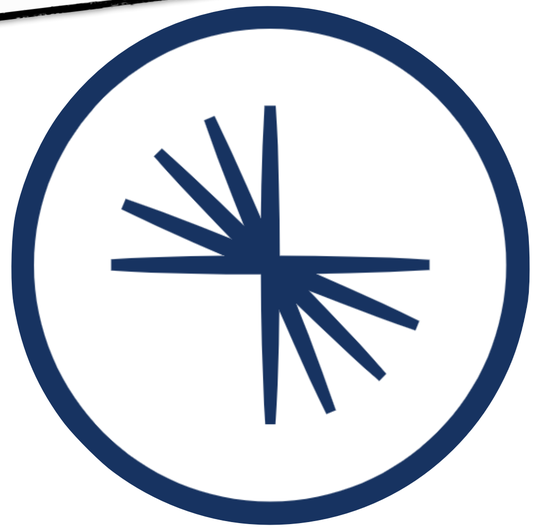
* T&C: https://www.confluent.io/confluent-cloud-promo-disclaimer

Deep-dive articles

- What is Apache Kafka?
- Event Streaming vs. Related Trends
- KRaft: Kafka without ZooKeeper
- Transactions & Guarantees in Kafka
- Processing & Storage Fundamentals
- Kafka Performance
- Cloud-native Kafka
- Streaming Database Systems
- Testing Apache Kafka
- Explore Kafka's Internals

Over 10 hours of free courses

- Apache Kafka 101
- Kafka Connect 101
- Kafka Streams 101
- ksqlDB 101
- Inside ksqlDB
- Spring Framework and Kafka
- Building Data Pipelines with Kafka
- Event Sourcing with Kafka
- Data Mesh 101



CONFLUENT

Developer

Plus: Hands-on Quick Starts and Client Language Guides + Event Streaming Patterns + More

developer.confluent.io

#EOF

@rmoff rmoff.dev/talks youtube.com/rmoff