

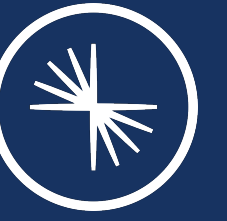
# *Kafka as a Platform: the Ecosystem from the Ground Up*



*Robin Moffatt  
@rmoff*

*London Microservices Meetup*





# 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

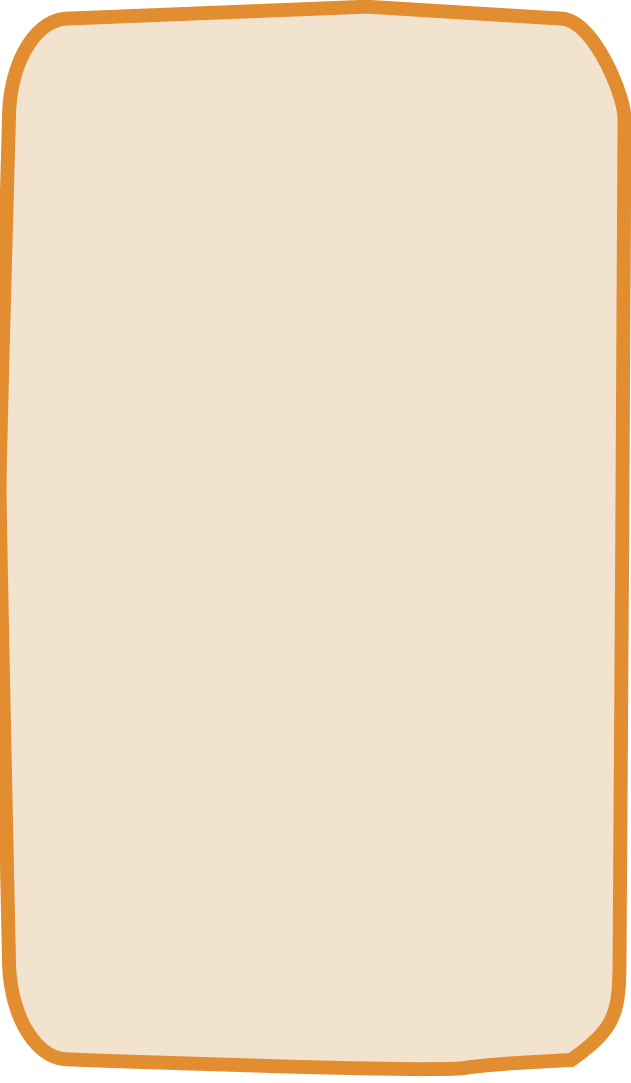
EVENTS

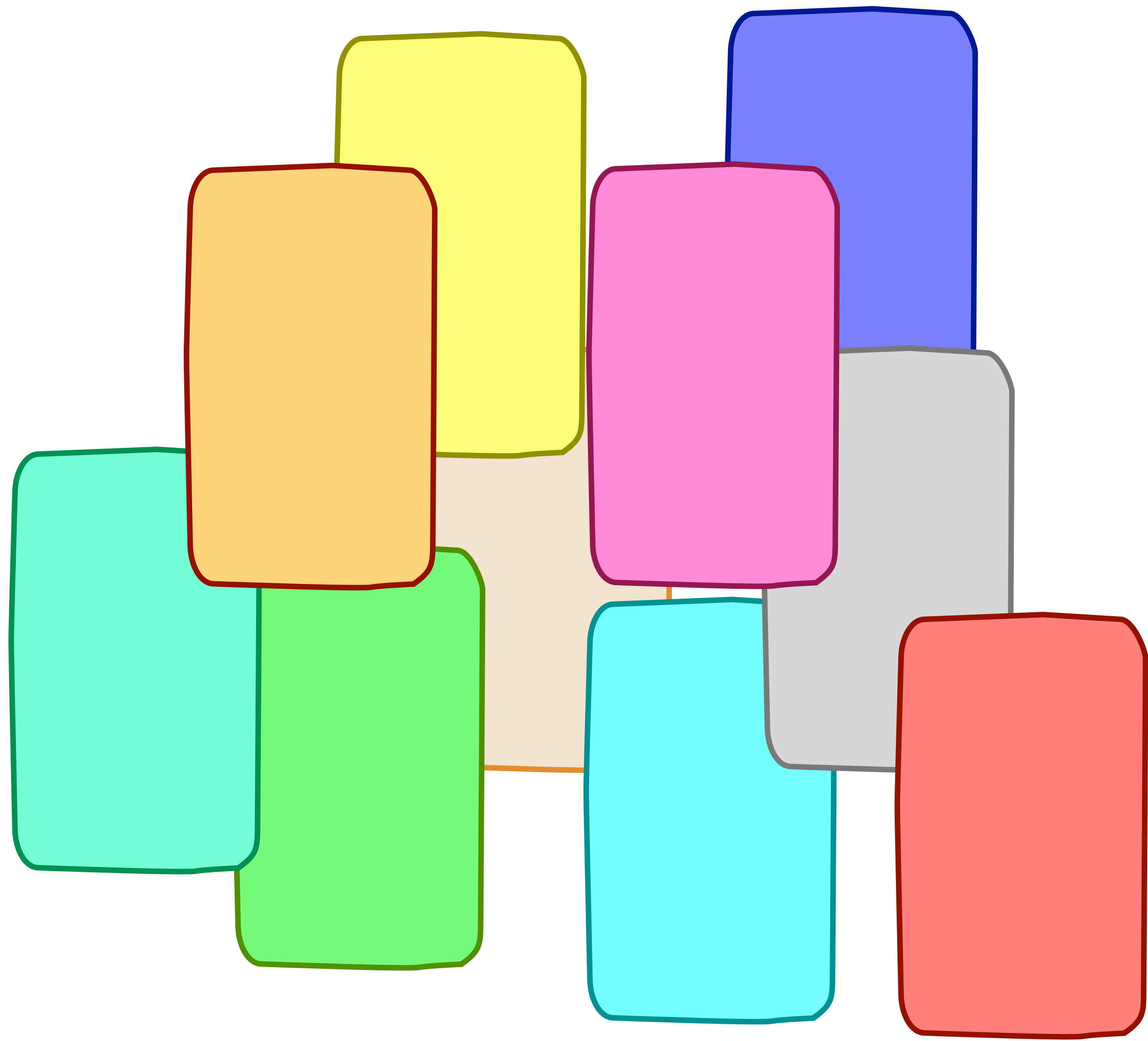
are

very  
^

POWERFUL

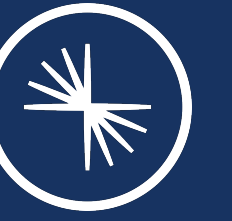








$K/v$



LOGG

@rmoff



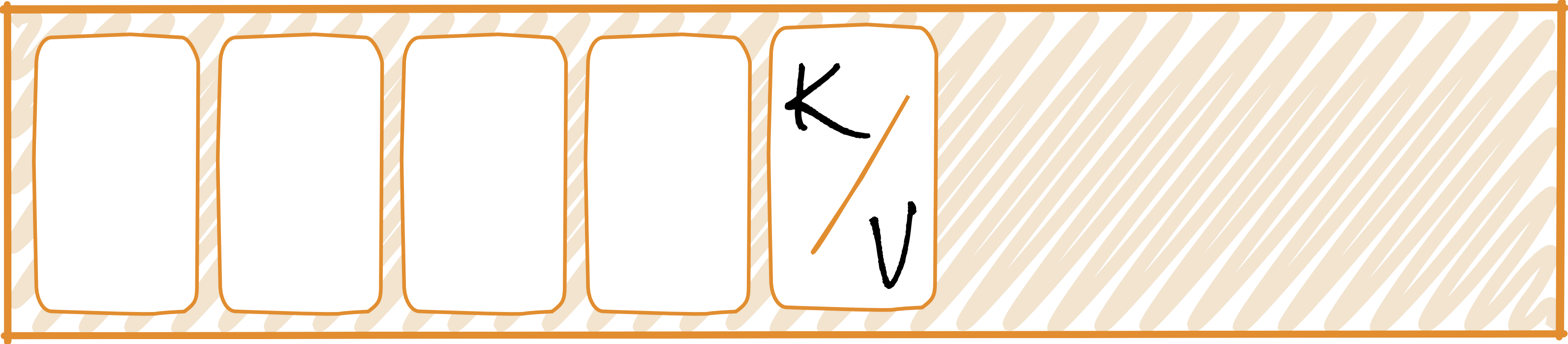


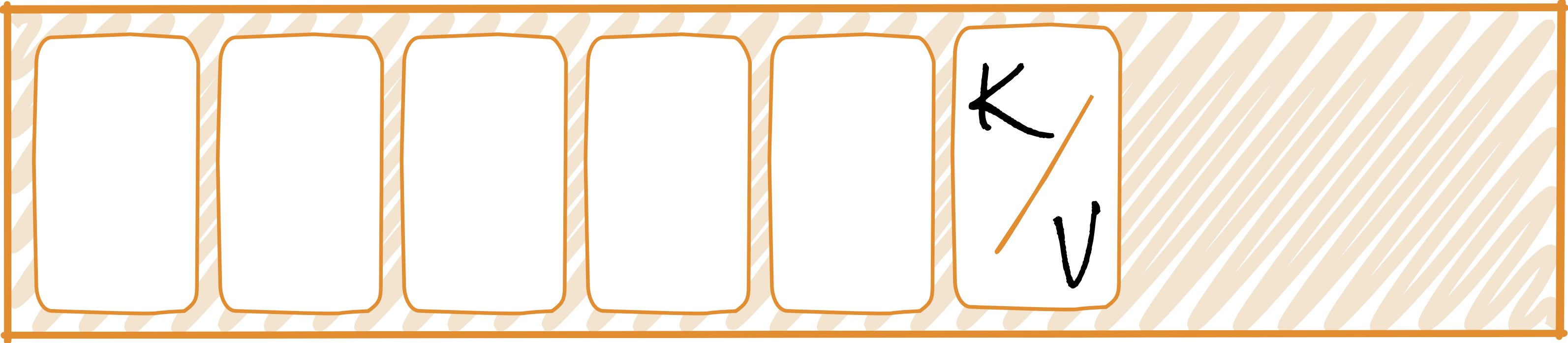


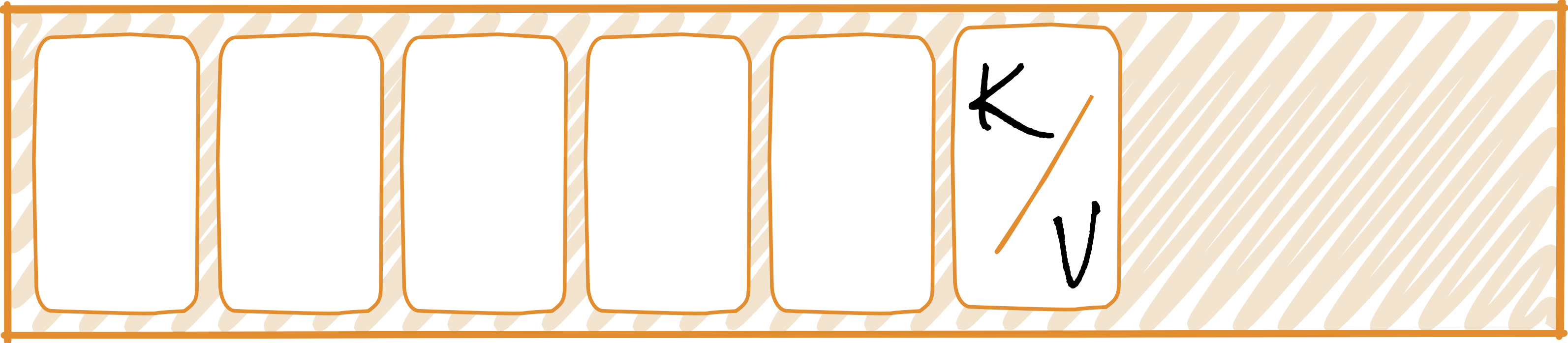










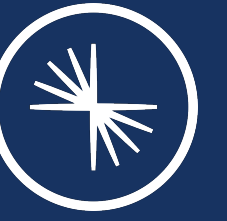


# Immutable Event Log



Events are added at the end of the log





# TOPICS

# Topics

Clicks



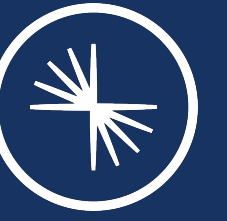
Orders



Customers



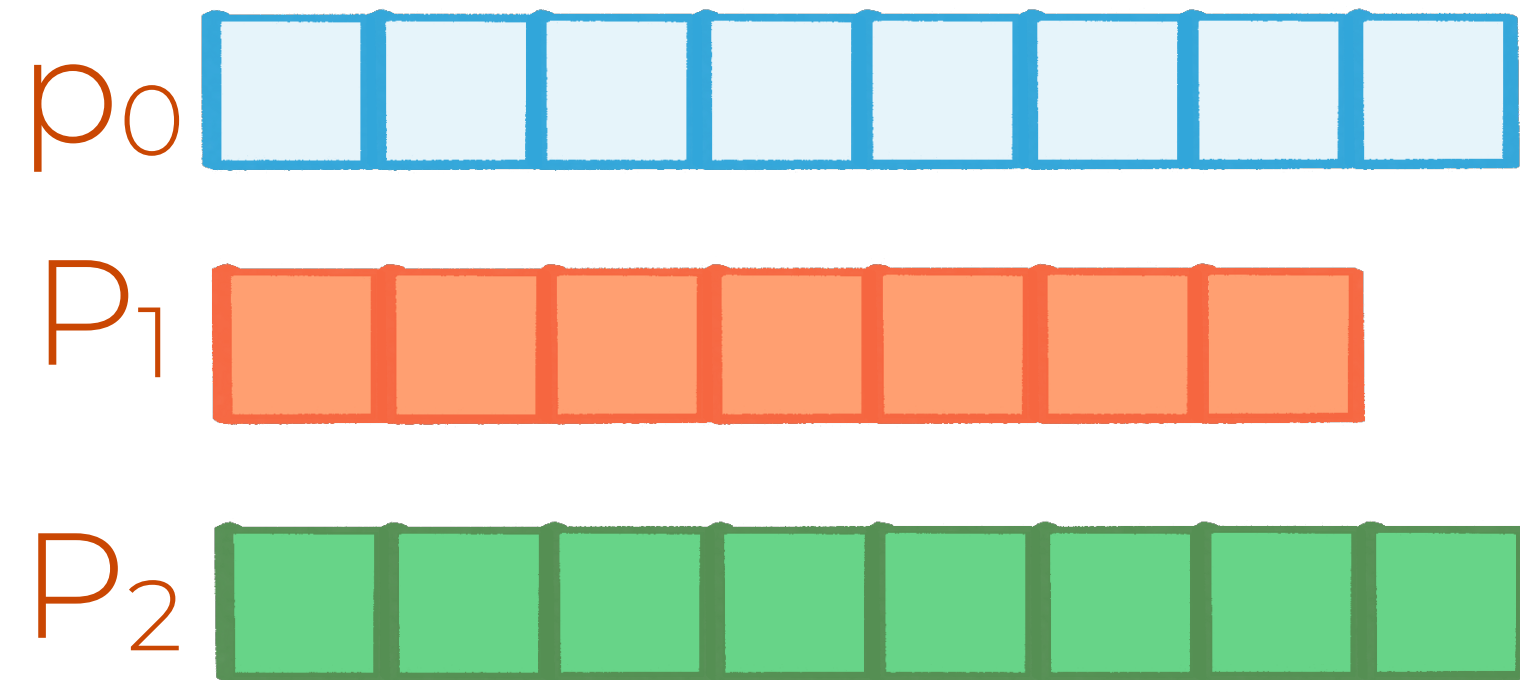
Topics are similar in  
concept to tables in a  
database



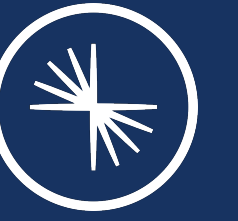
# PARTITIONS

# Partitions

Clicks

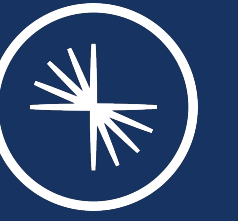


Messages are guaranteed to be strictly ordered within a partition



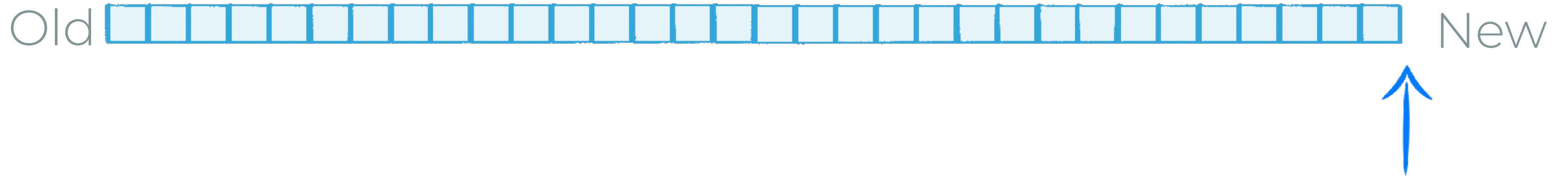
# PUB / SUB



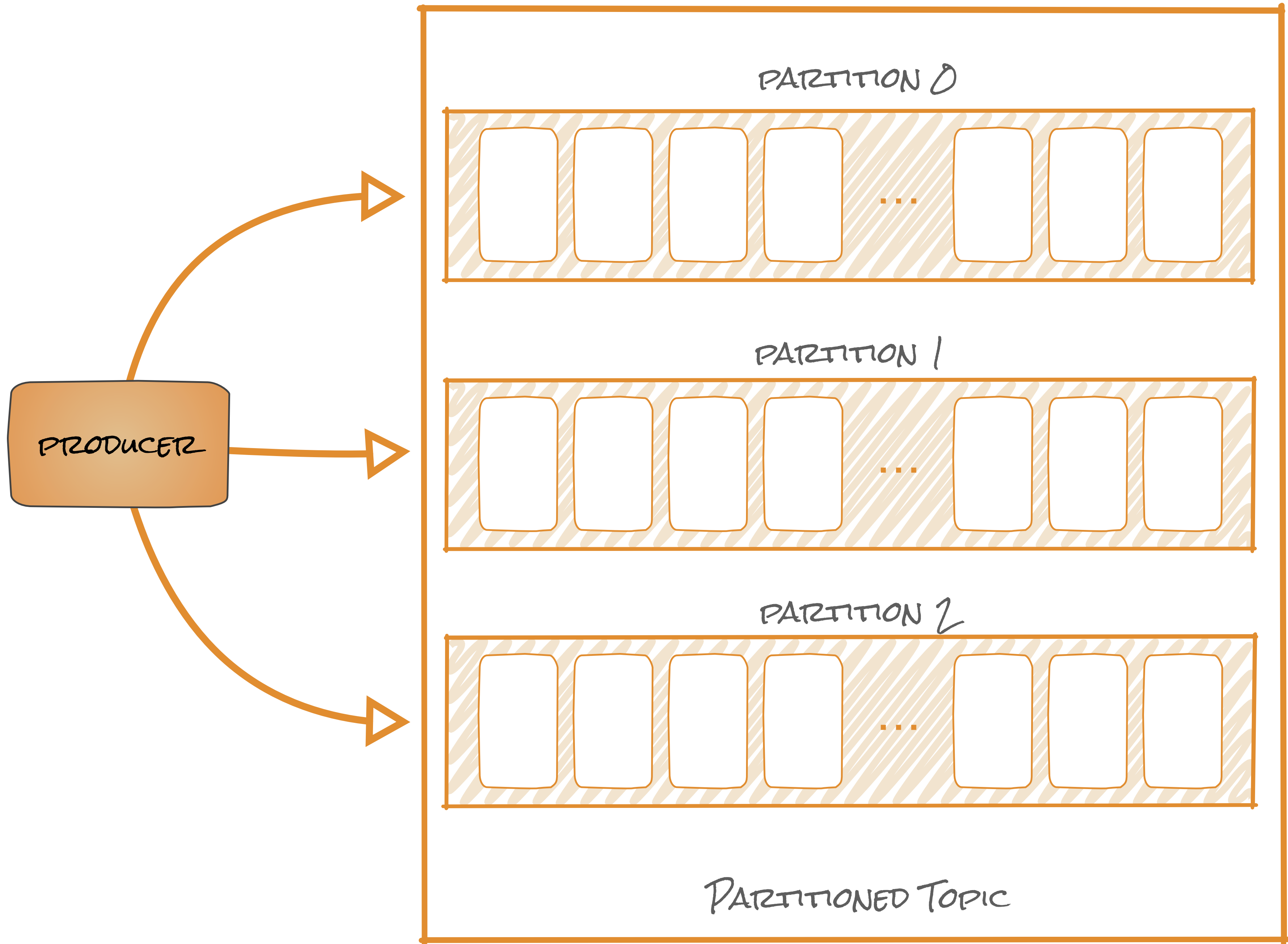


# PUB / SUB

# Producing data



Messages are added at the end of the log



```
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"})
```

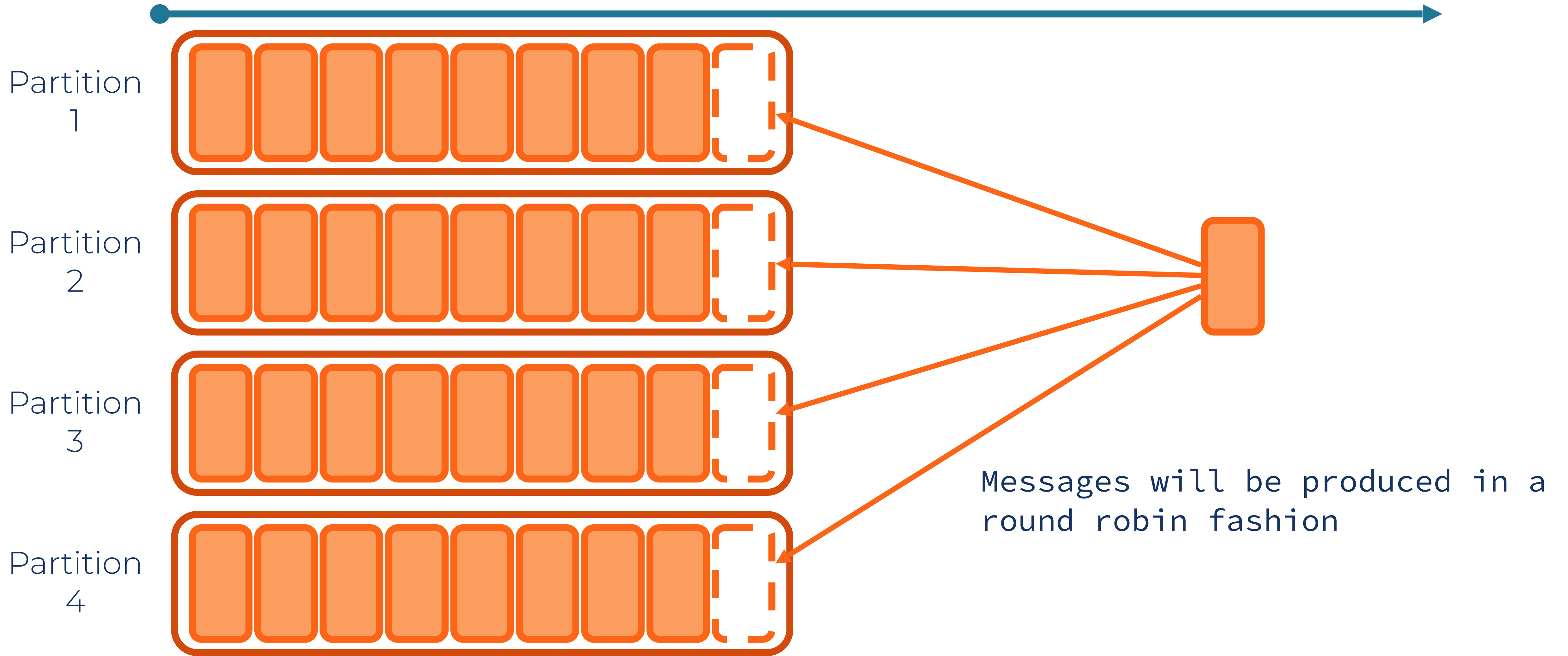
```
    defer p.Close()
```

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

```
}
```

# Producing to Kafka - No Key

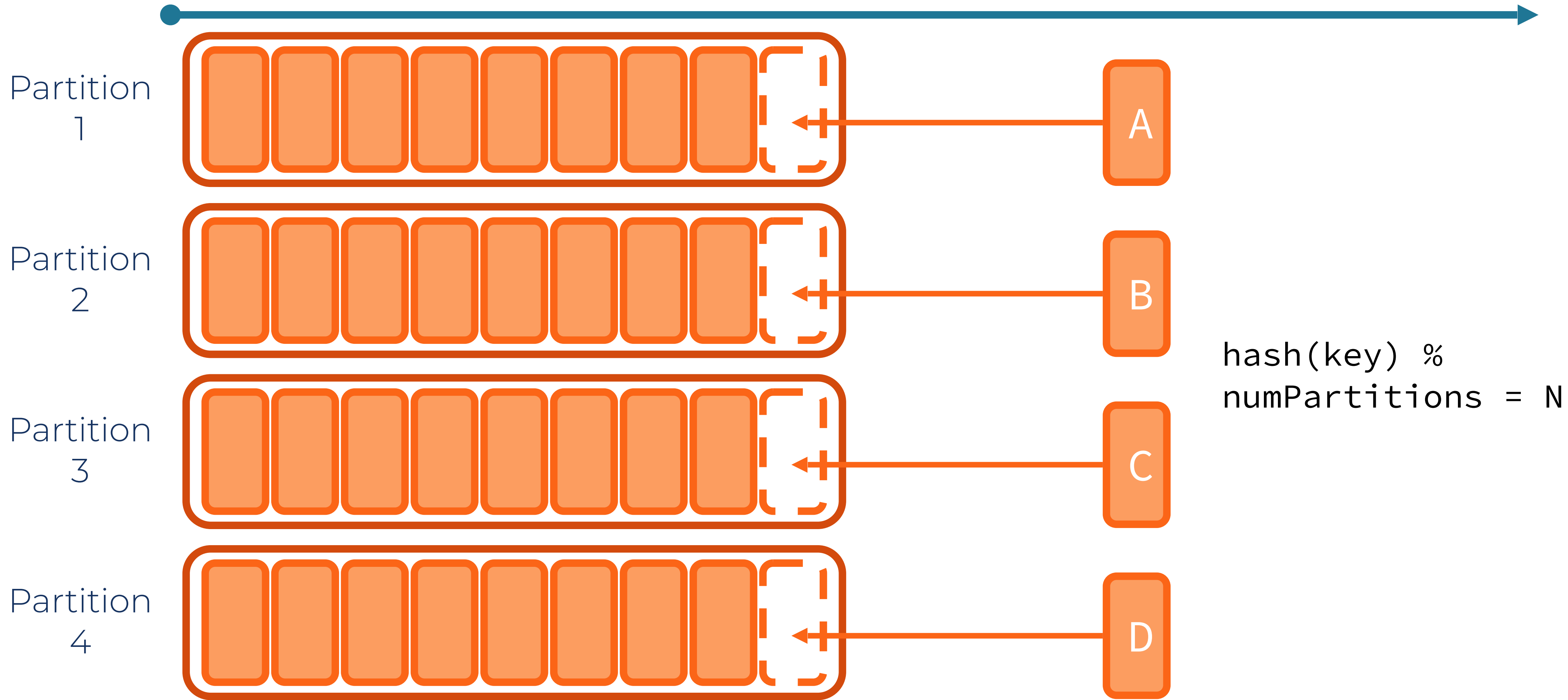
Time



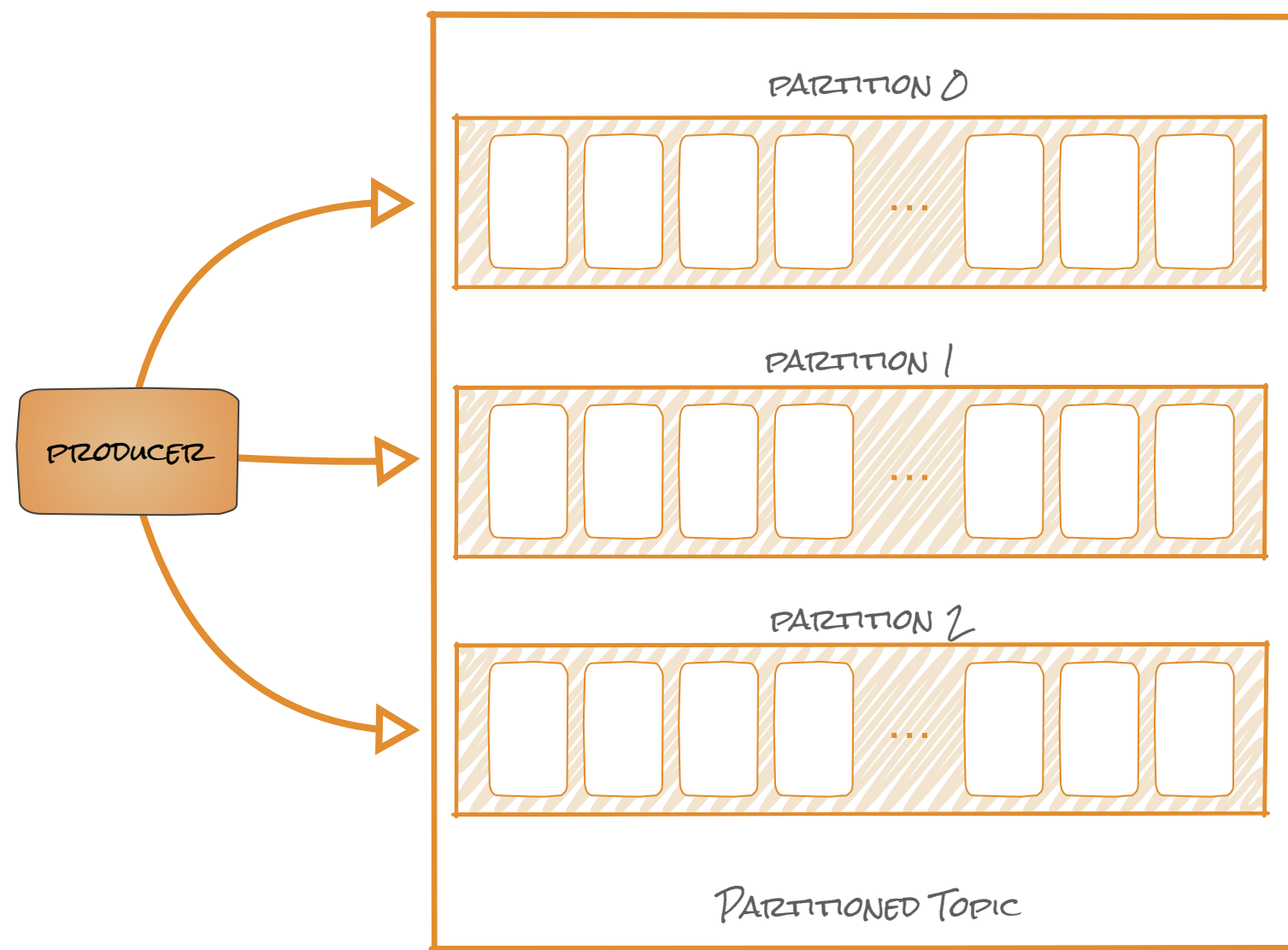


# Producing to Kafka - With Key

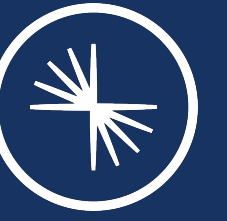
Time



# Producers



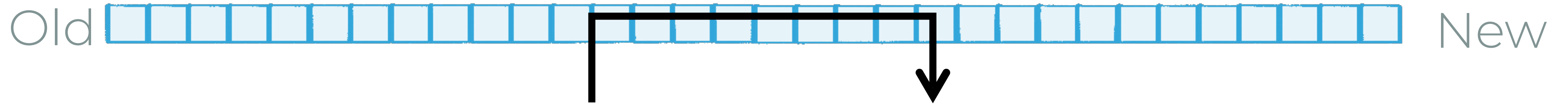
- **A client application**
- **Puts messages into topics**
- **Handles partitioning, network protocol**
- **Java, Go, .NET, C/C++, Python**
- **Also every other language**  
*Plus REST proxy if not*



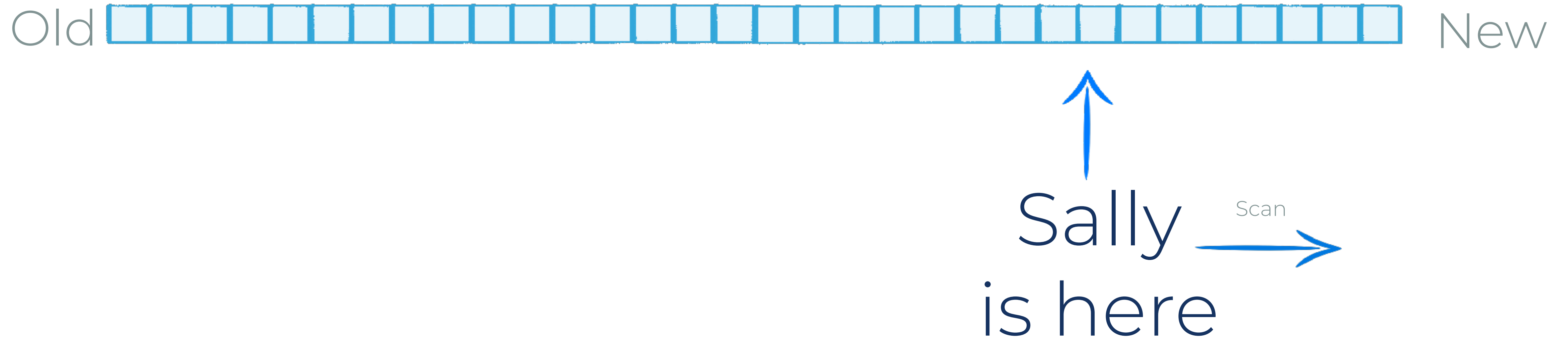
# PUB / SUB

# Consuming data - access is only sequential

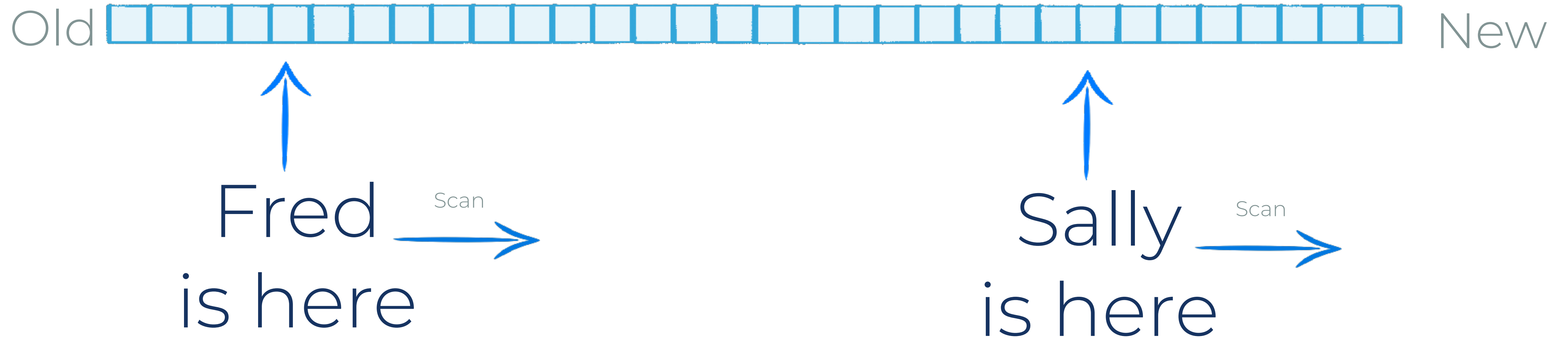
Read to offset & scan



# Consumers have a position of their own

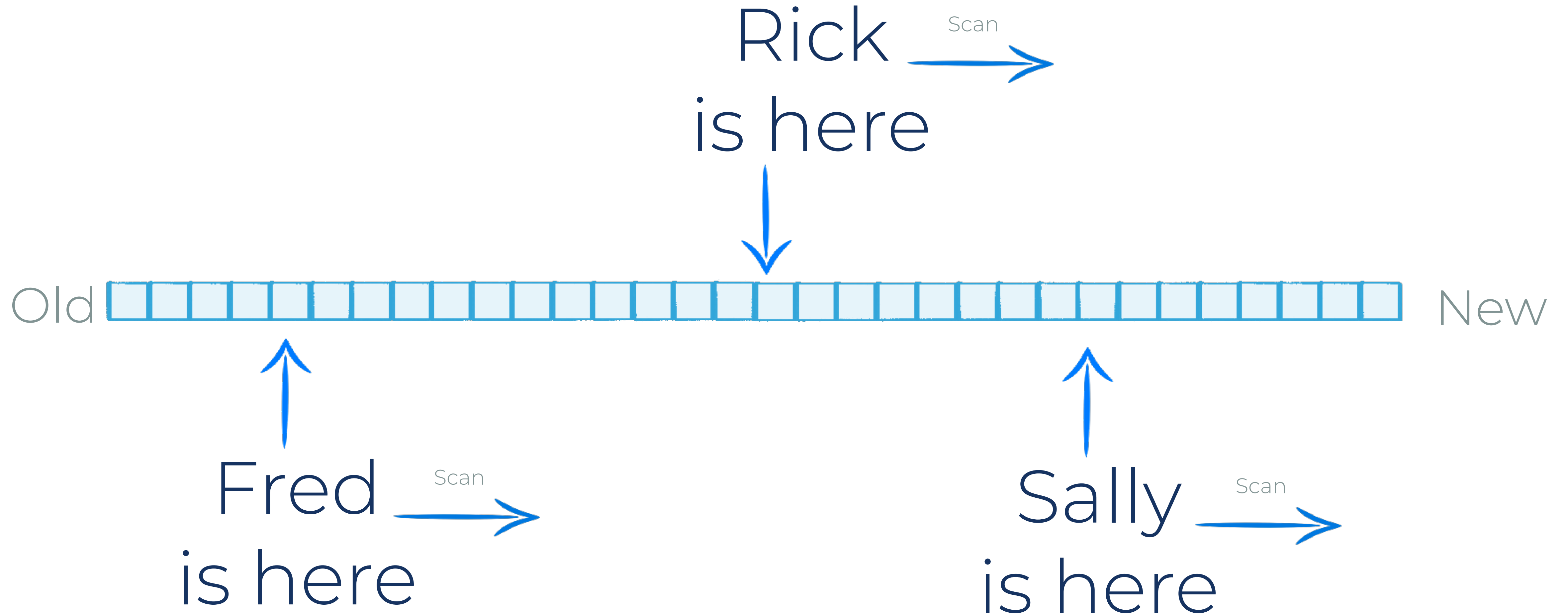


# Consumers have a position of their own





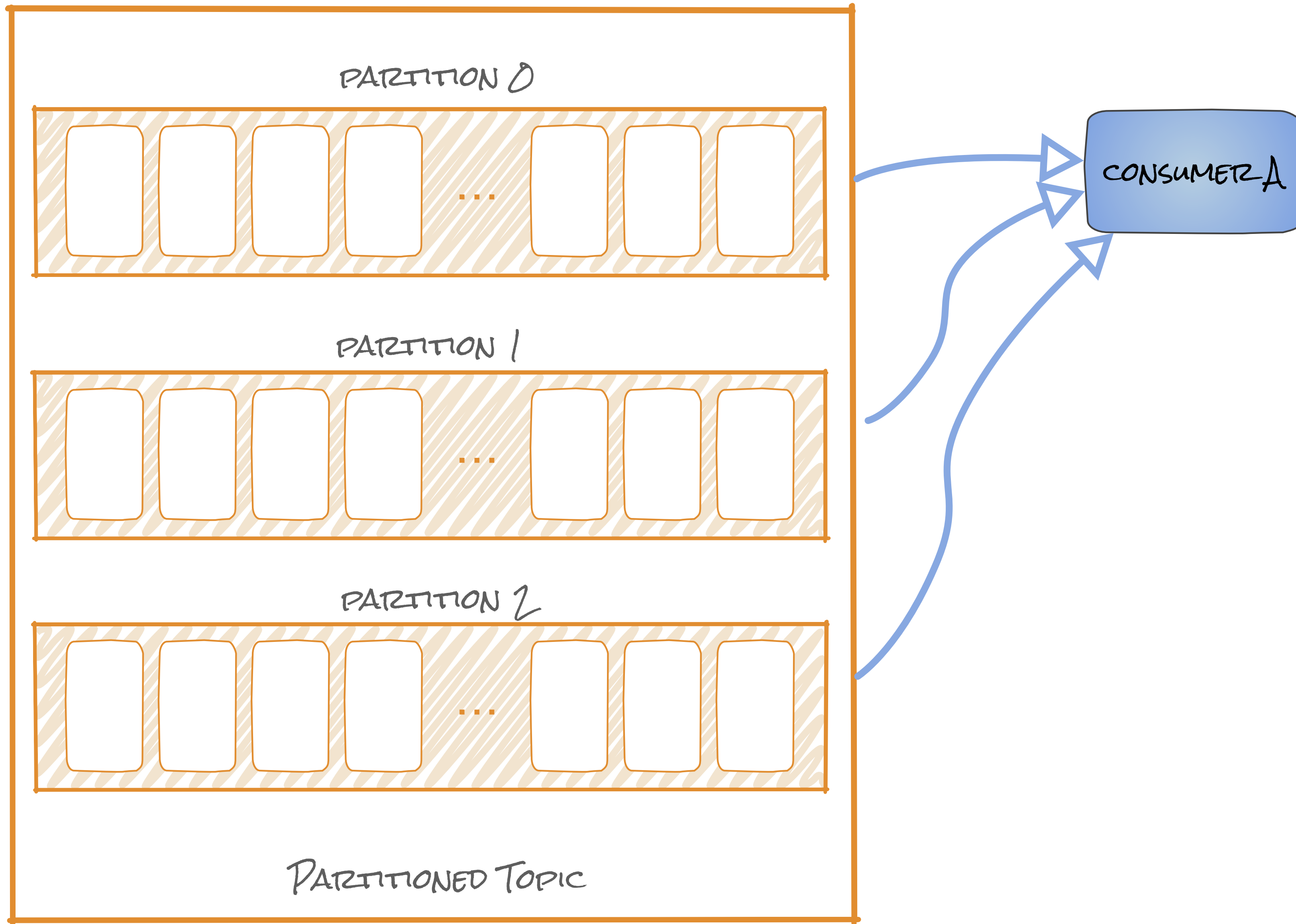
# Consumers have a position of their own



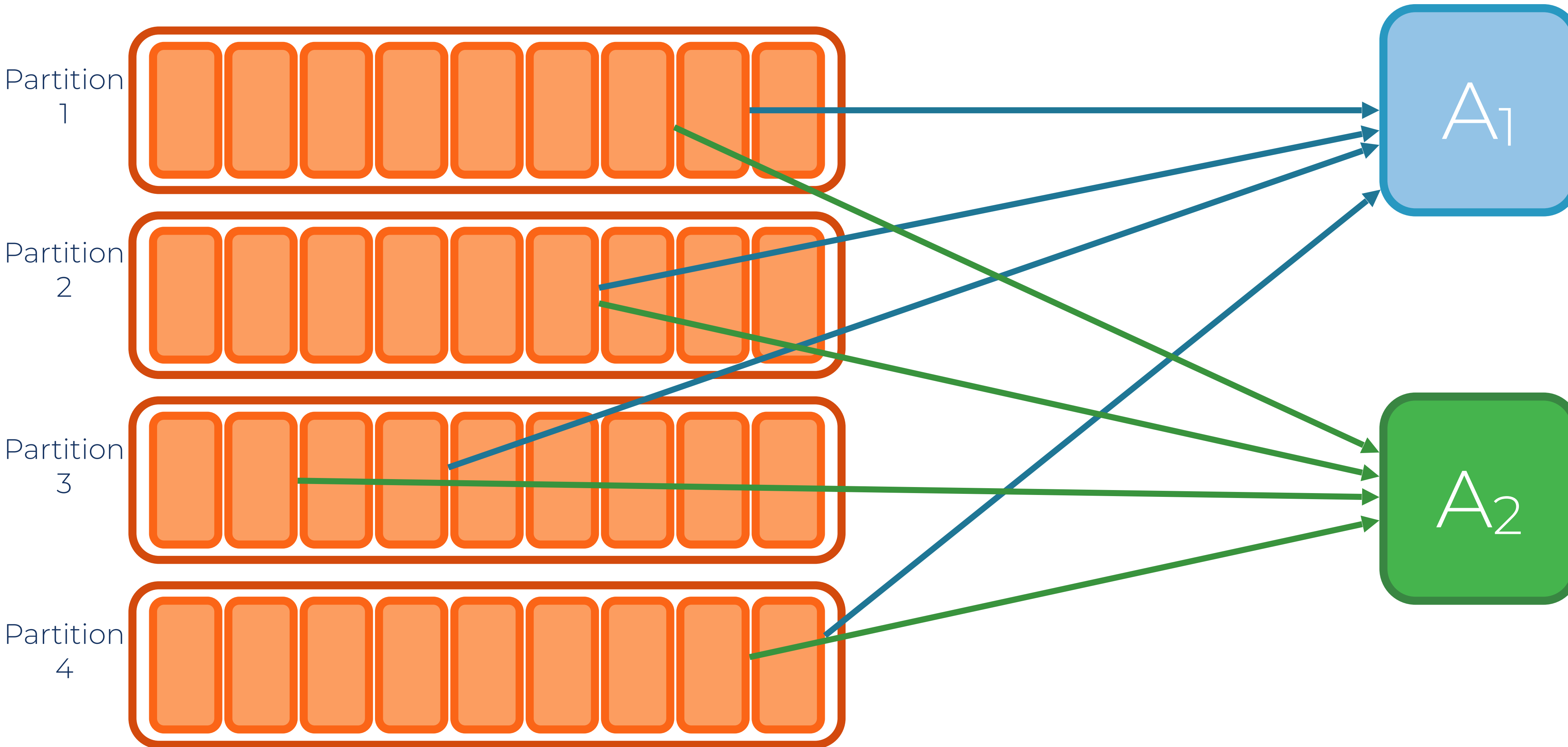
```
c, _ := kafka.NewConsumer(&cm)
defer c.Close()
c.Subscribe(topic, nil)

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

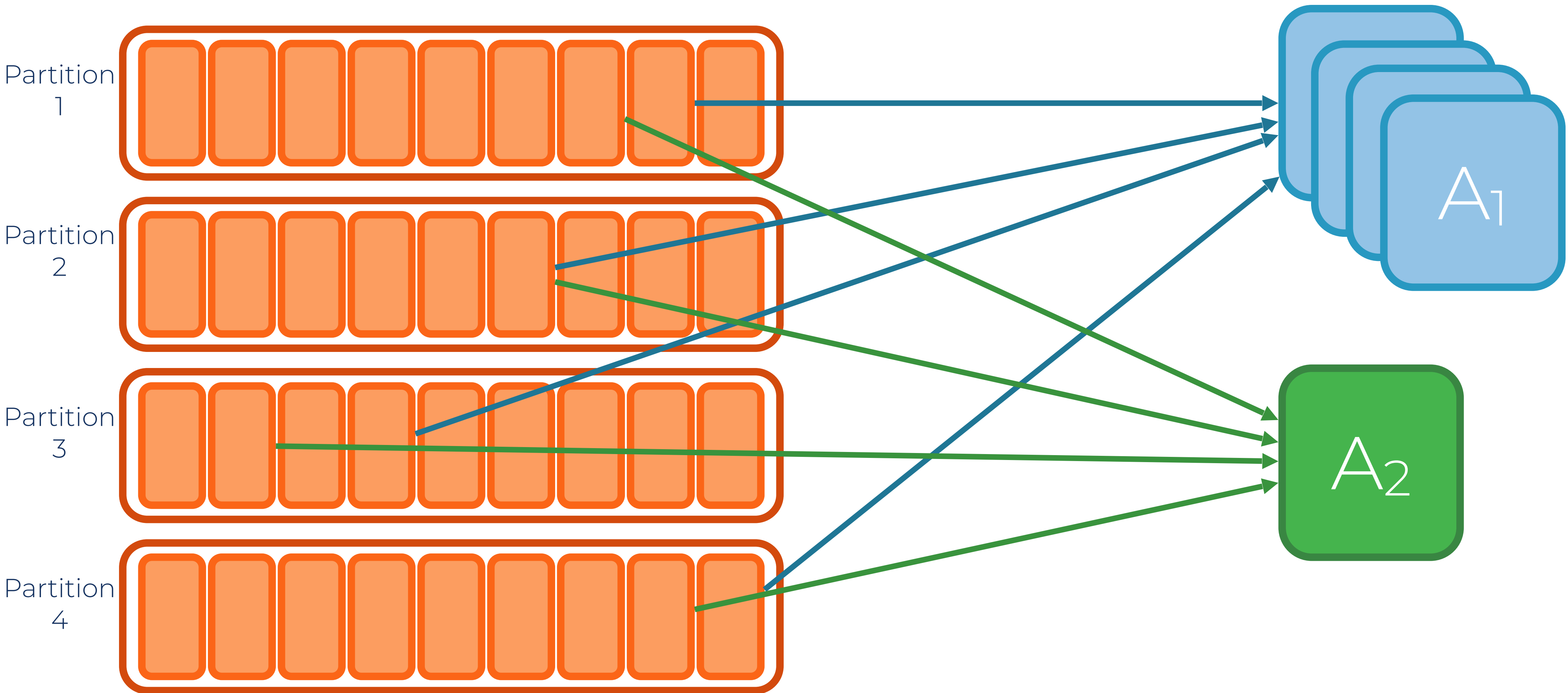
        case *kafka.Message:
            km := ev.(*kafka.Message)
            fmt.Printf("✅ Message '%v' received from topic
'%v' \n", string(km.Value), string(*km.TopicPartition.Topic))
            }
        }
    }
}
```



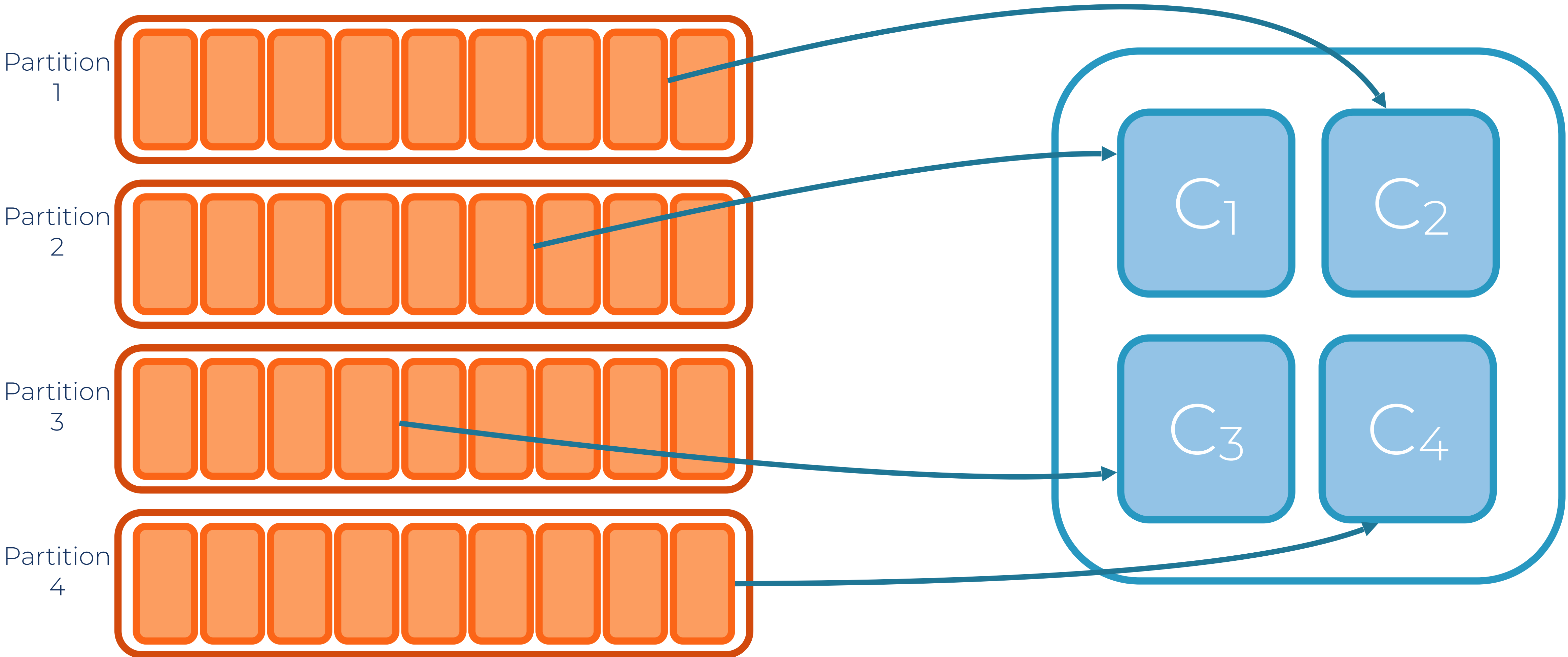
# Consuming From Kafka - Multiple Consumers



# Consuming From Kafka - Grouped Consumers

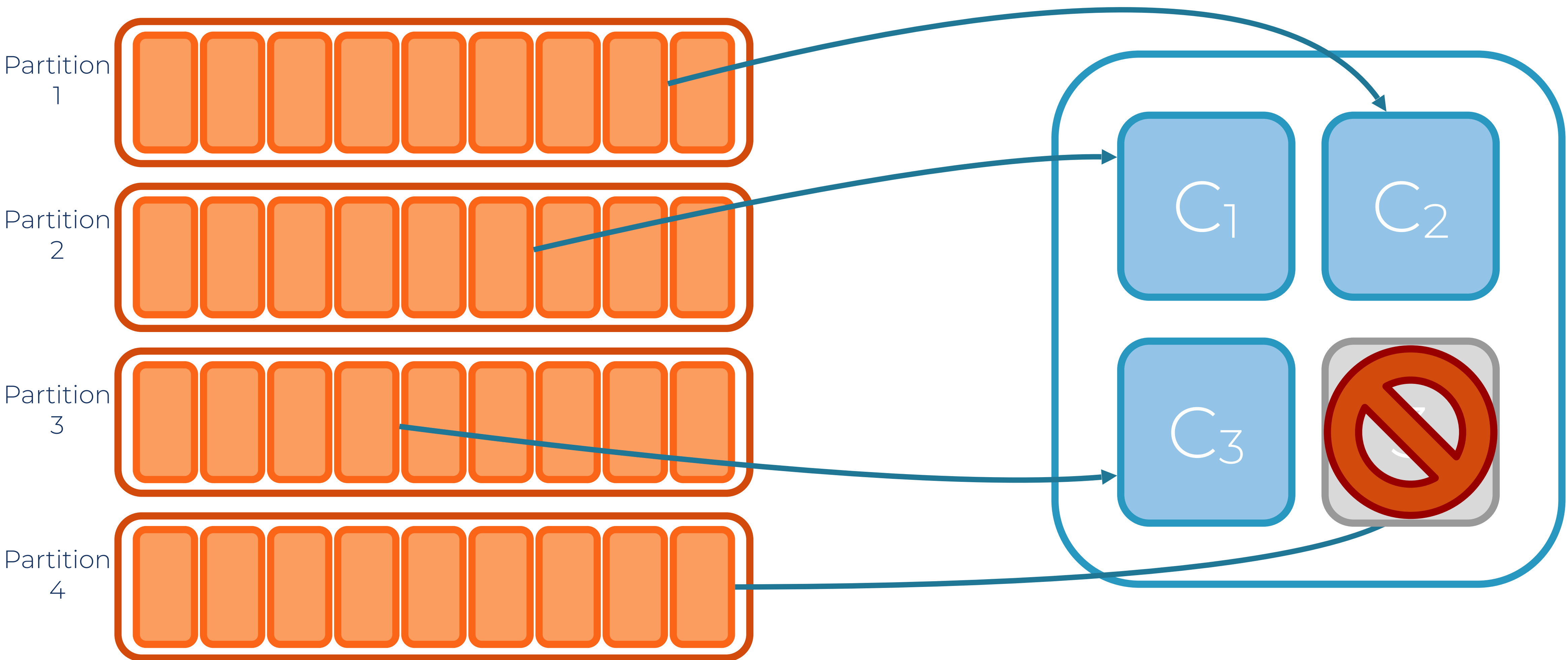


# Consuming From Kafka - Grouped Consumers

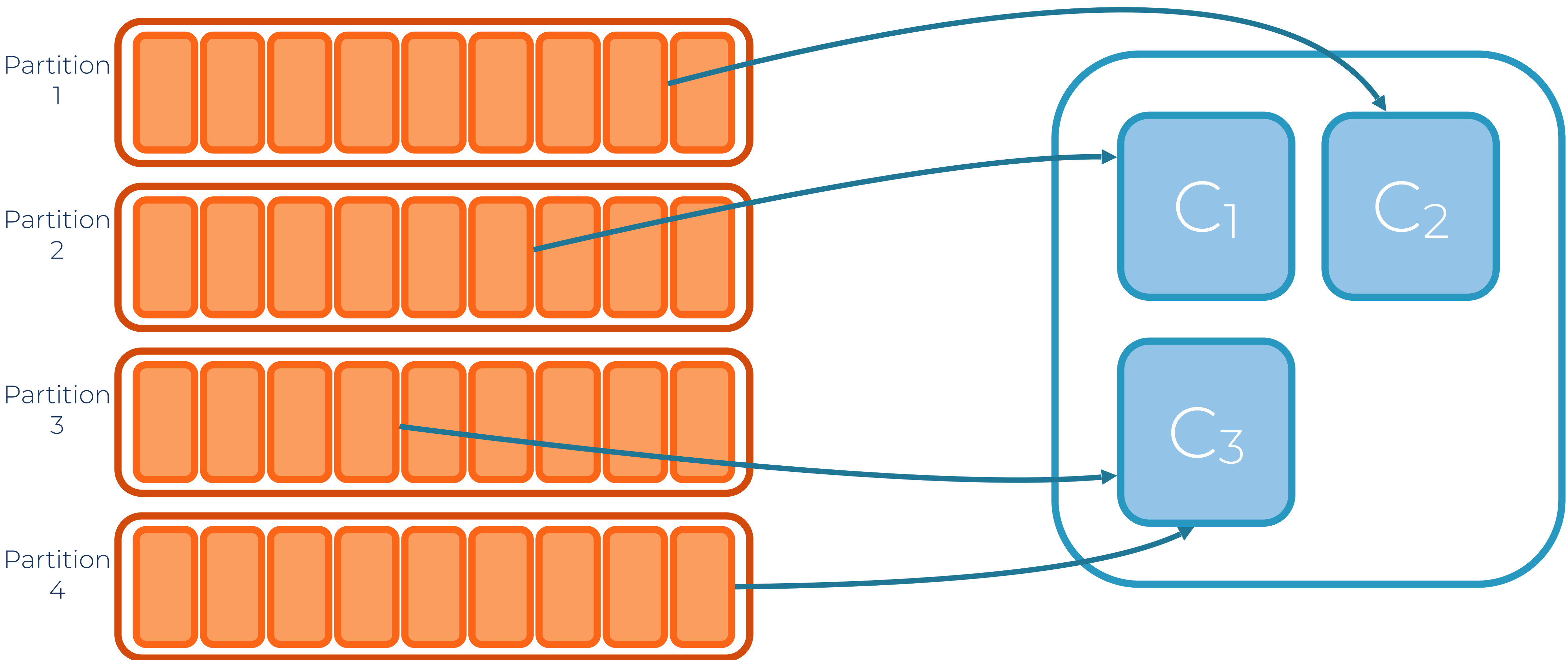




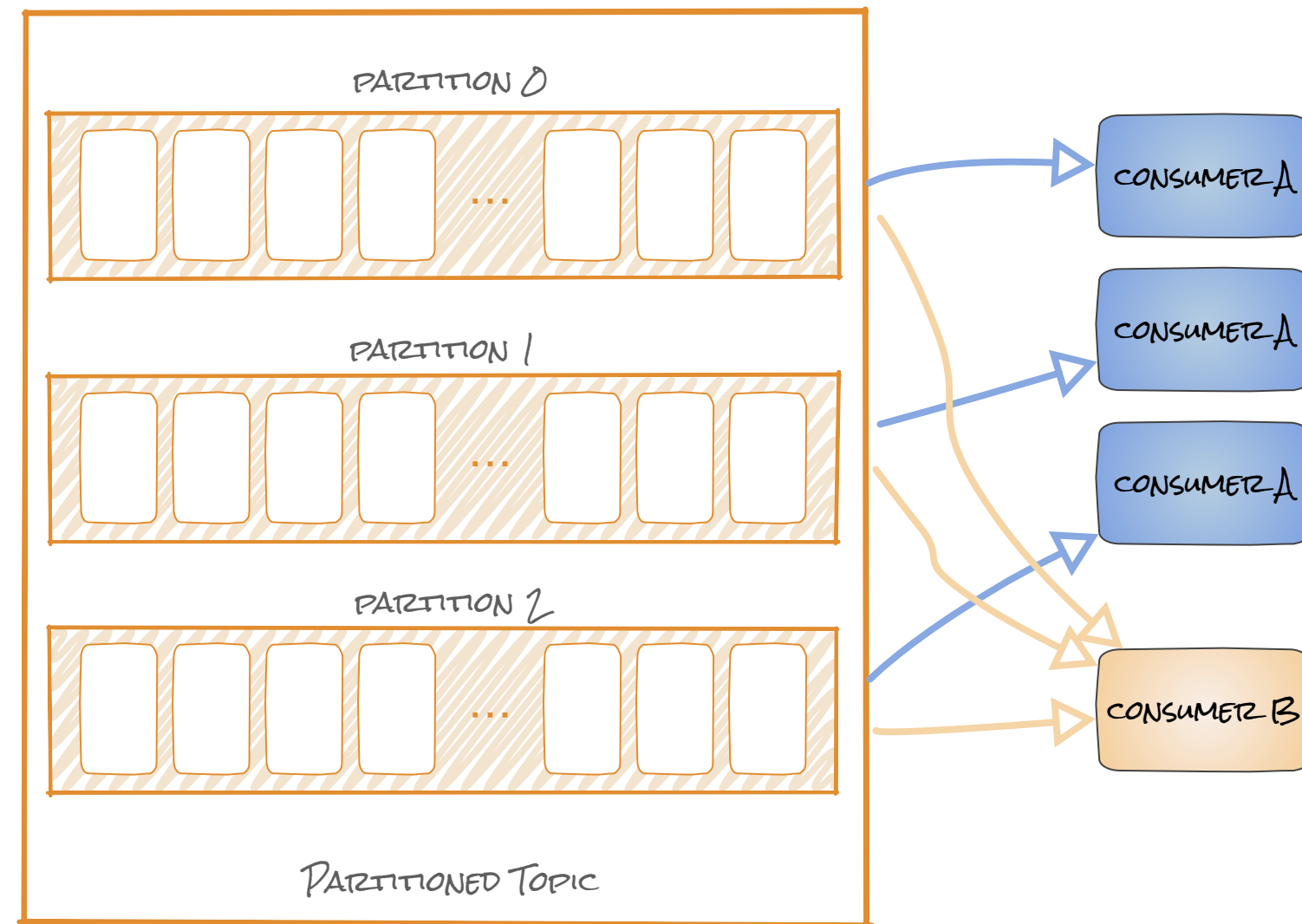
# Consuming From Kafka - Grouped Consumers



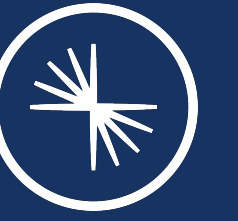
# Consuming From Kafka - Grouped Consumers



# 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*



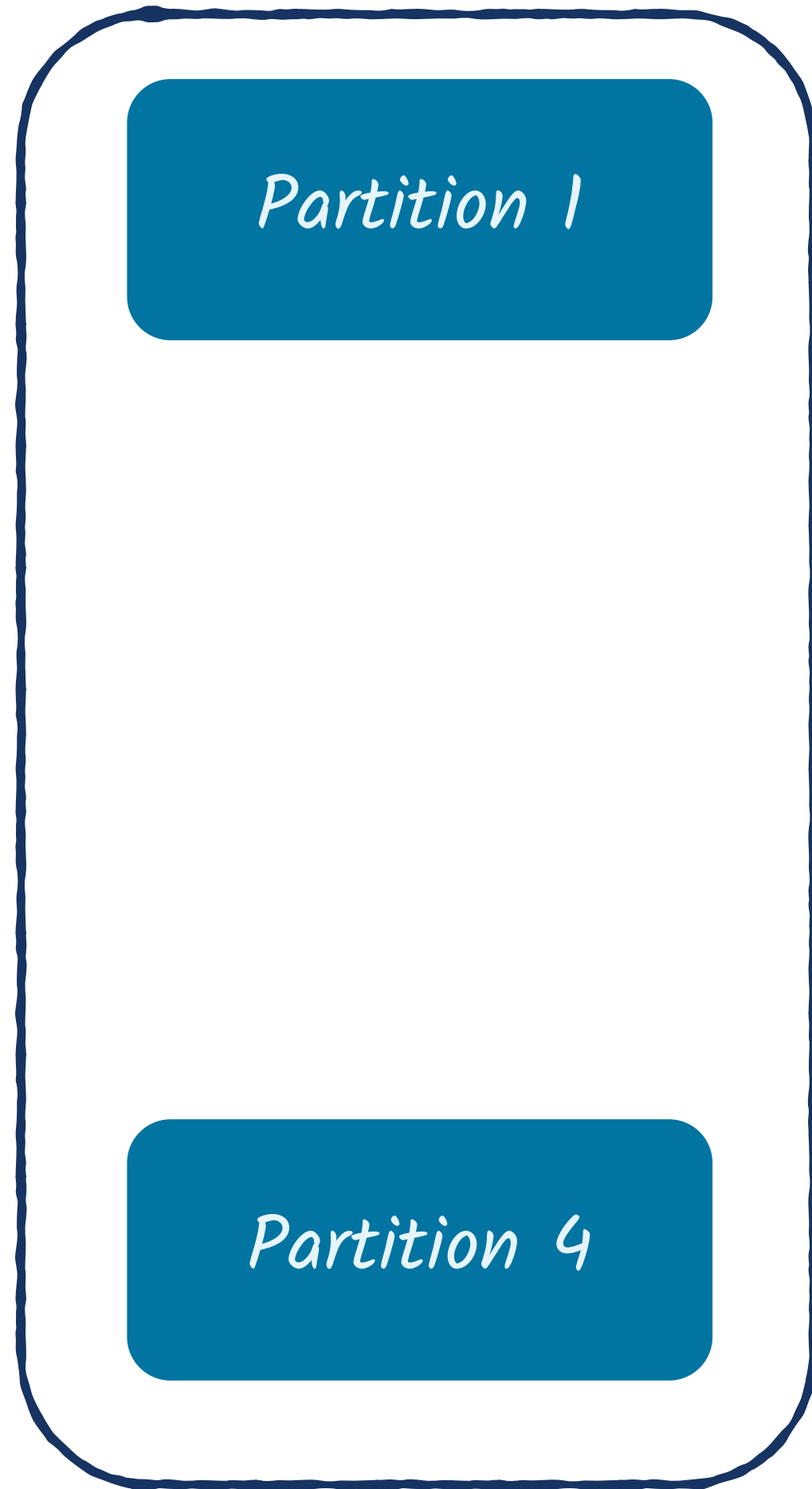
# BROKERS

and REPLICATION

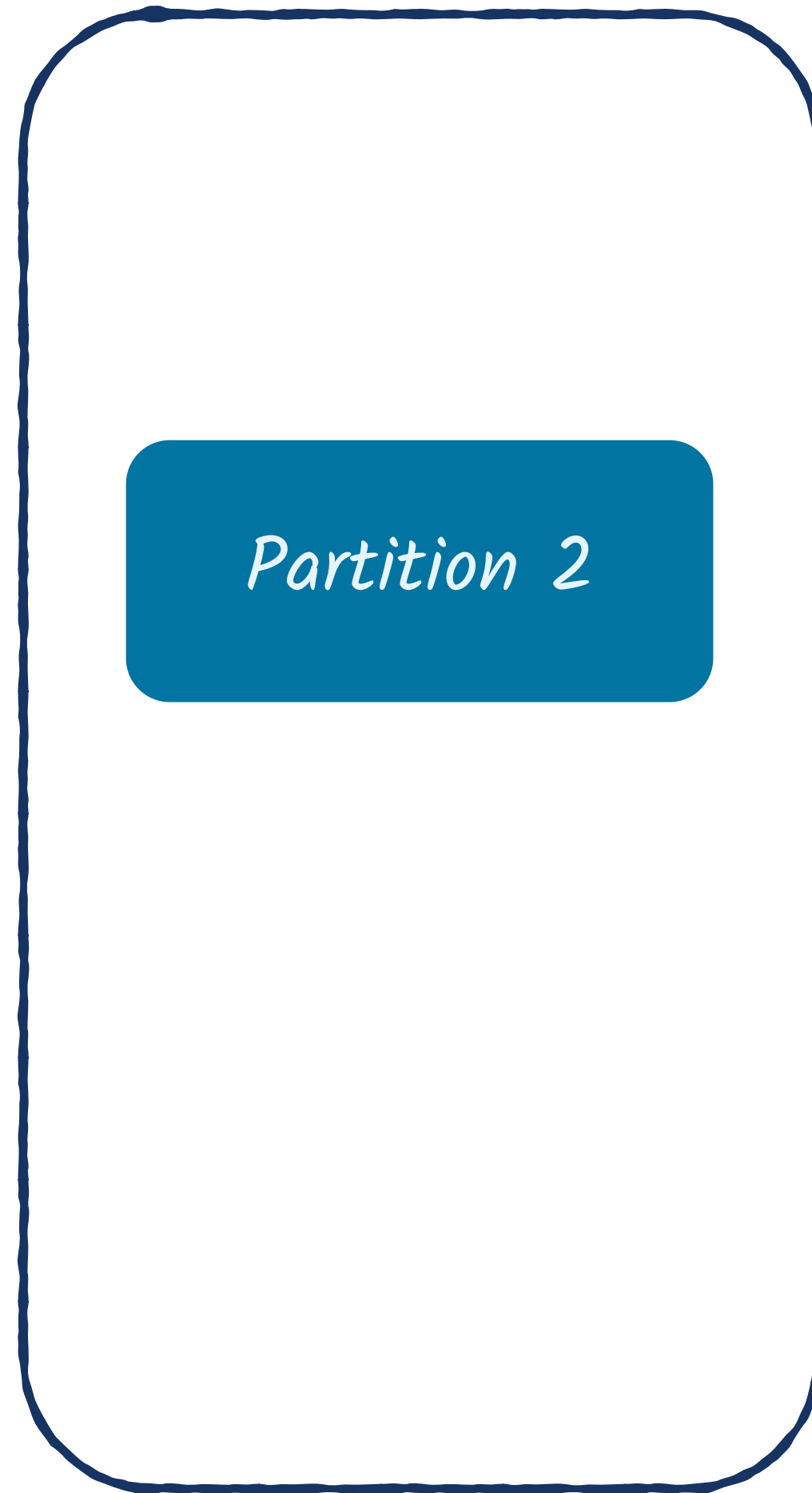
# Partition Leadership and Replication

Leader

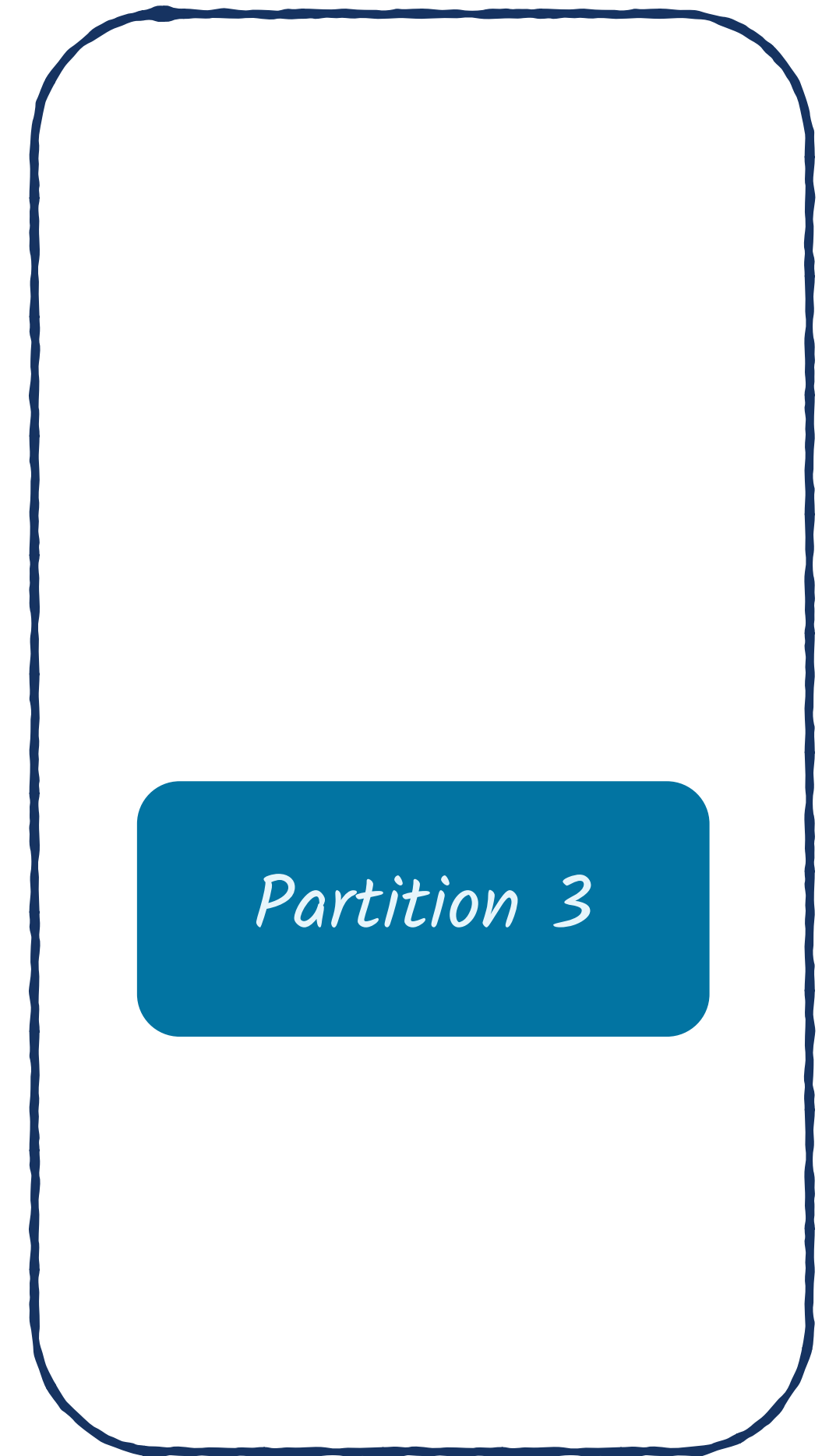
Follower



Broker 1



Broker 2

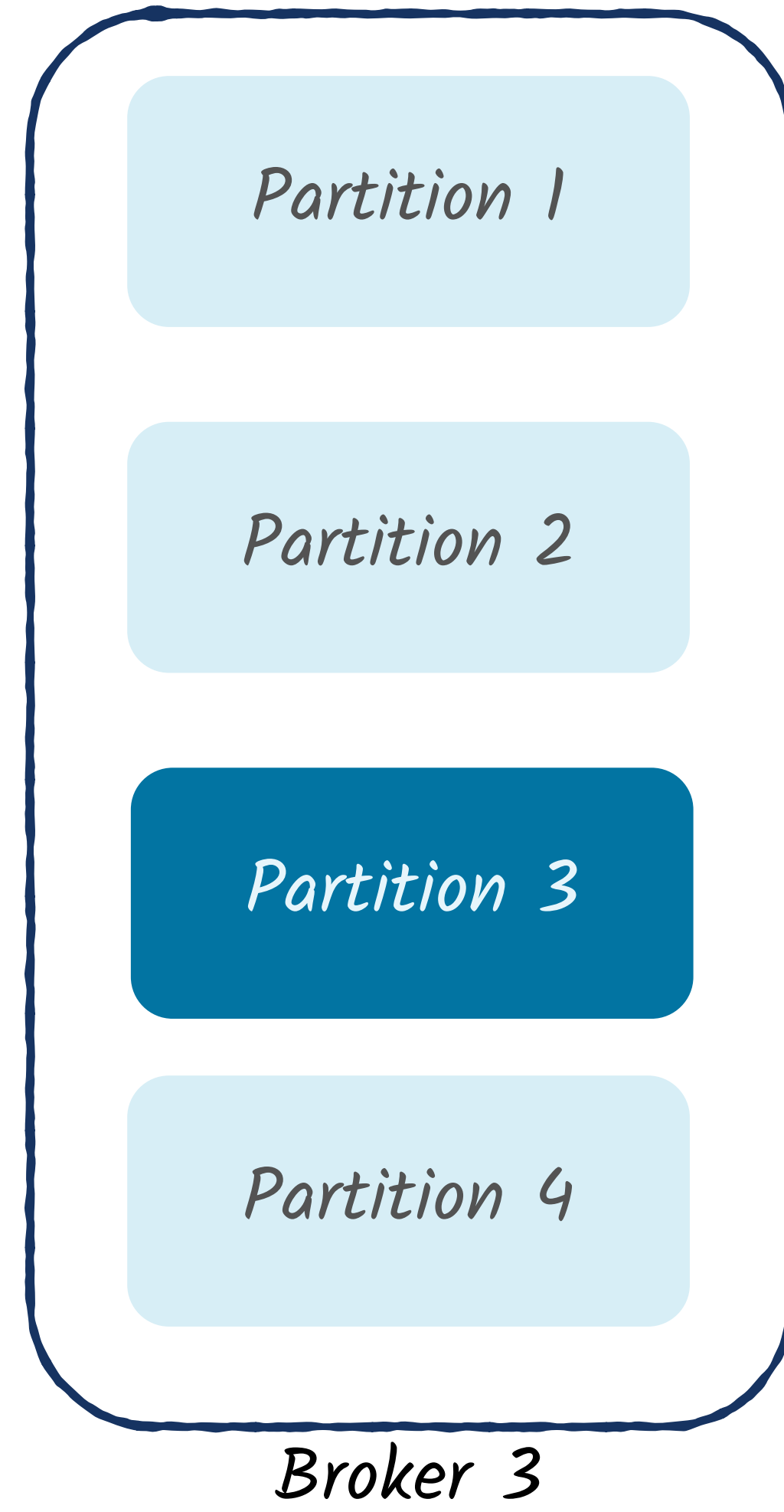
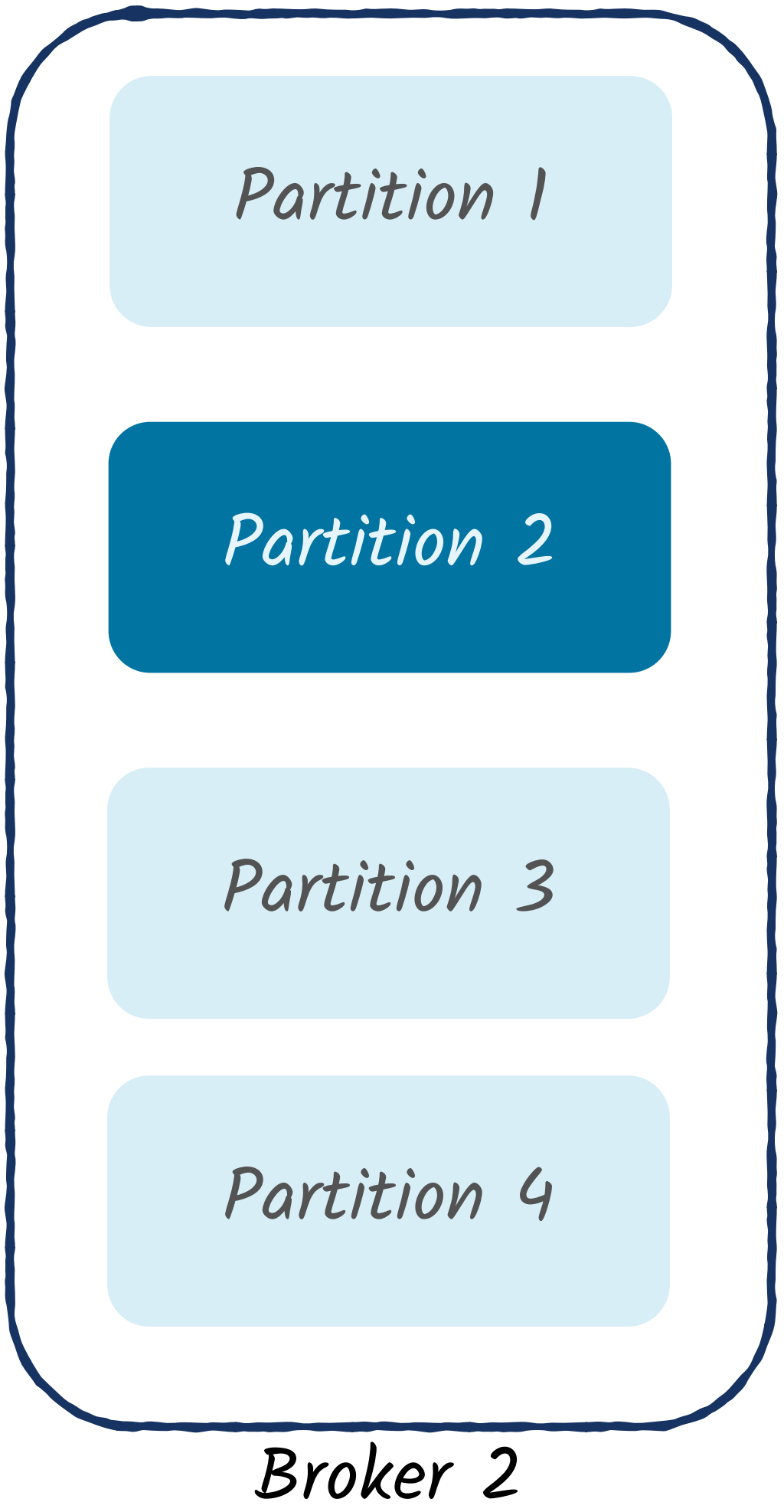
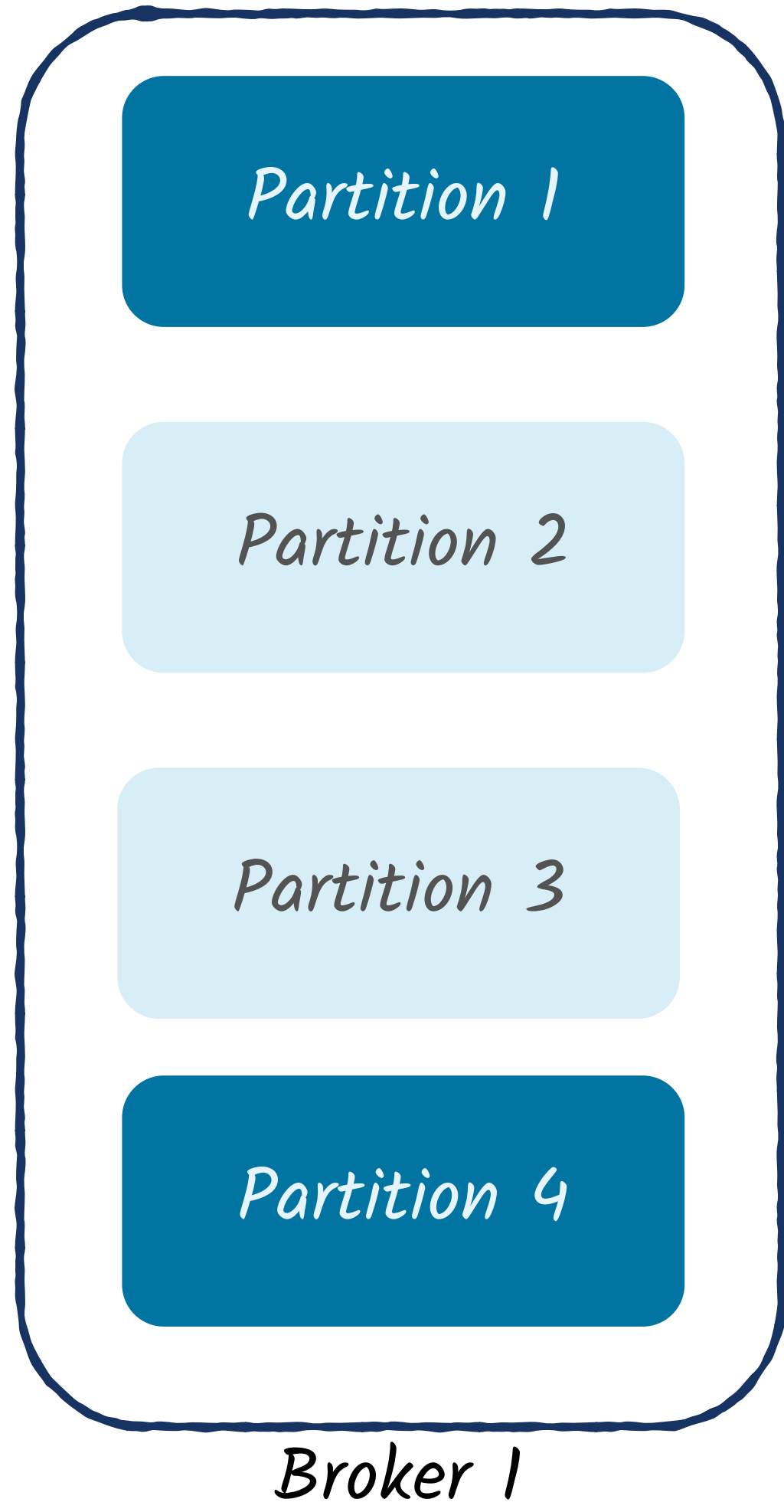


Broker 3

# Partition Leadership and Replication

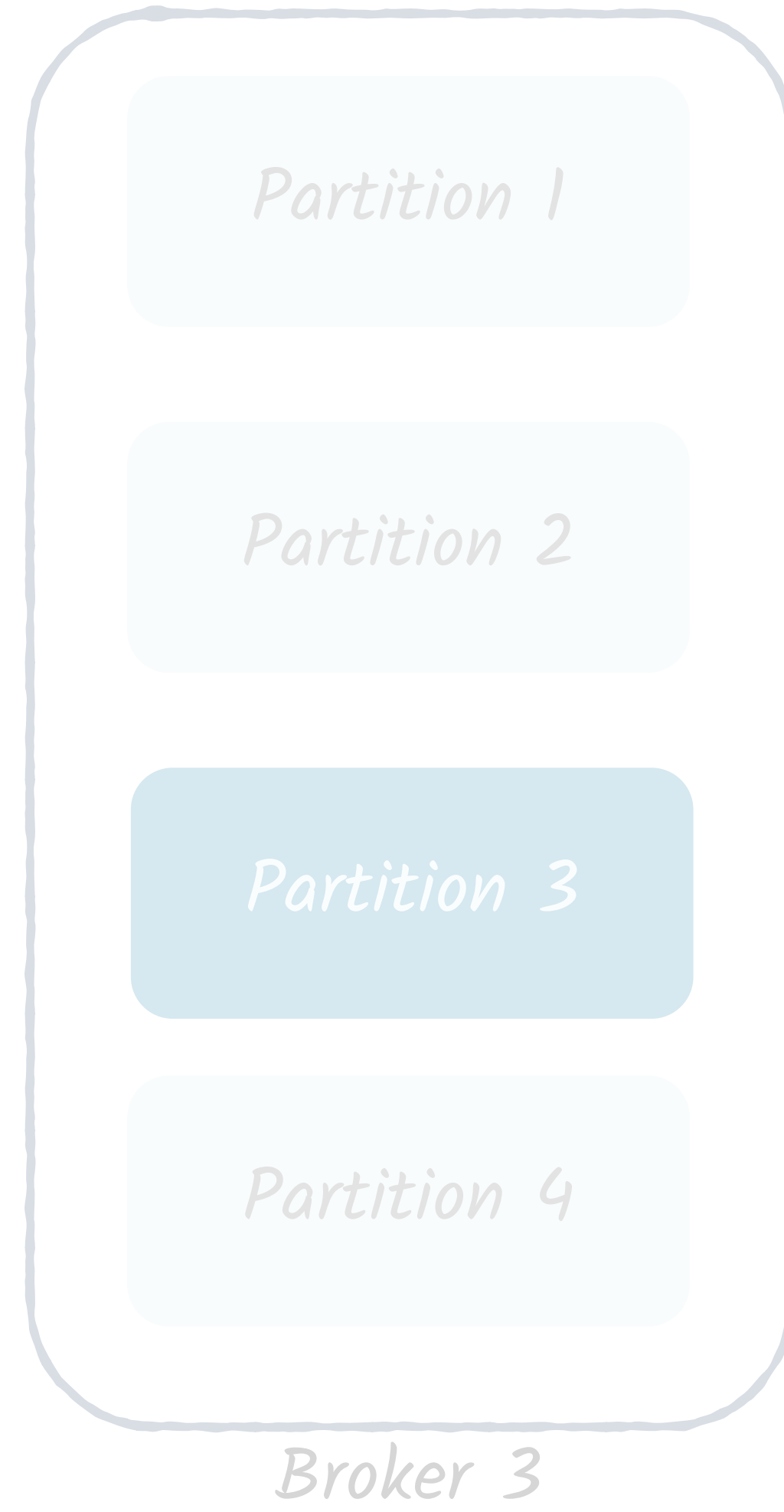
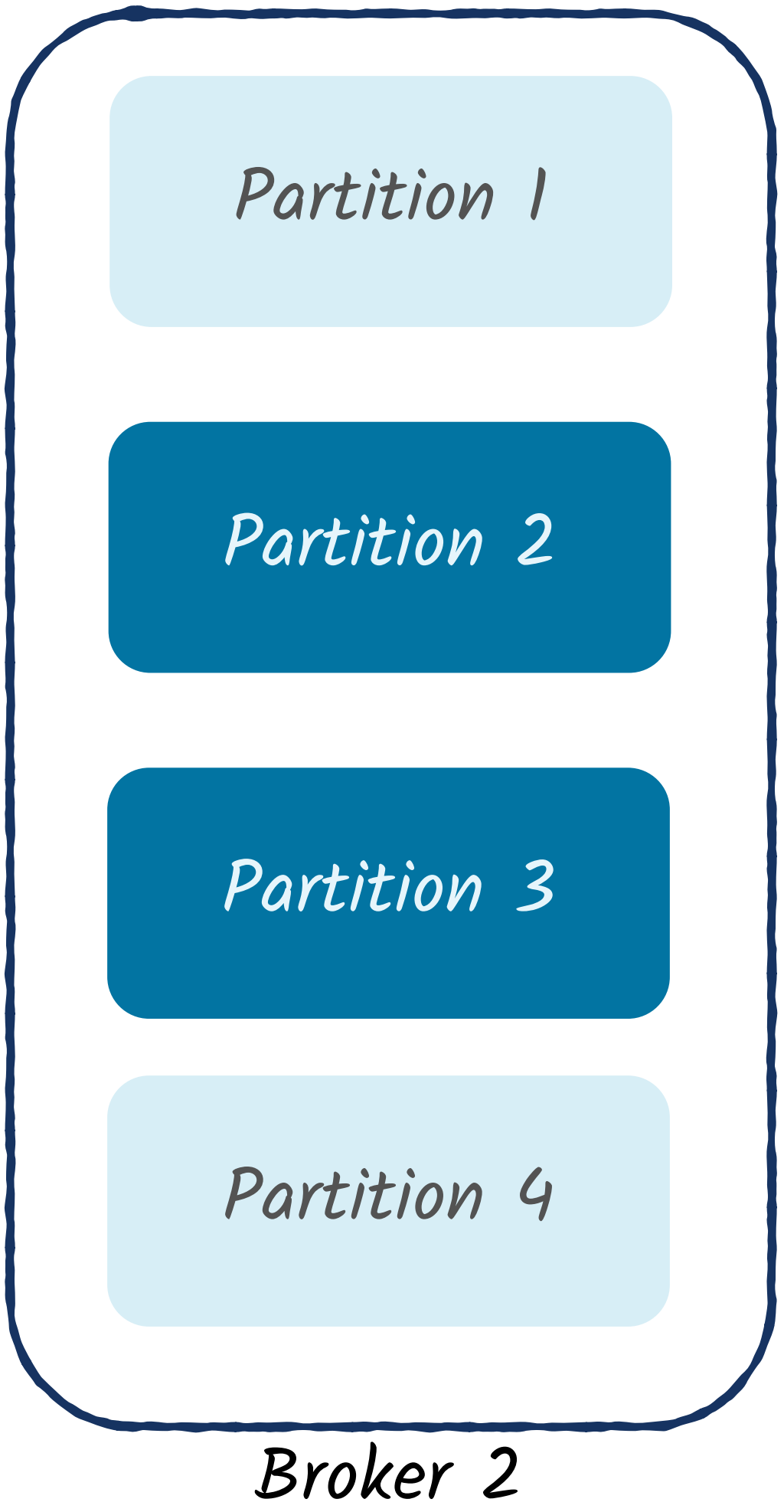
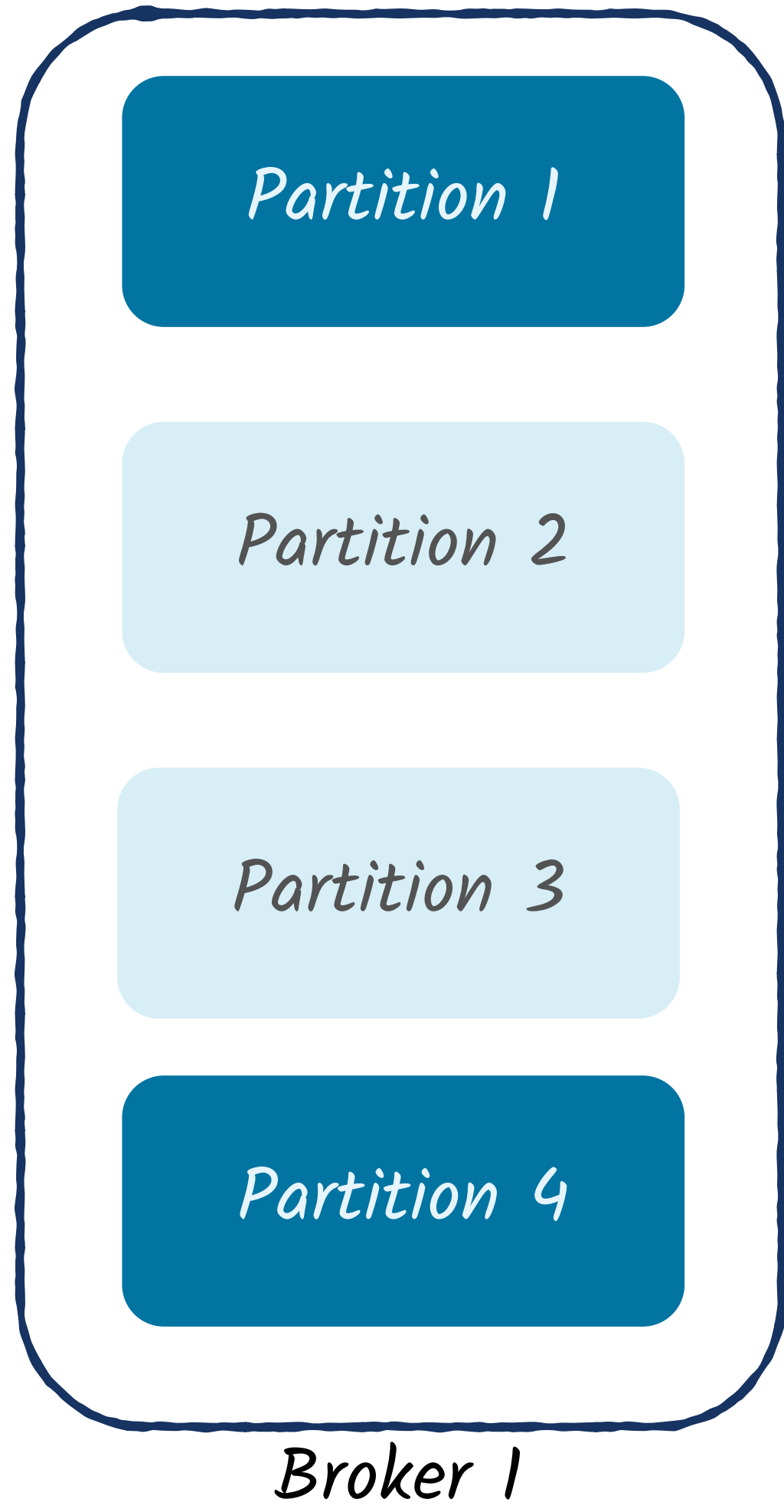
Leader

Follower

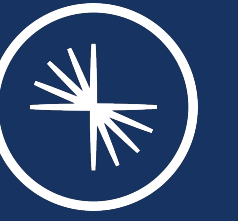


# Partition Leadership and Replication

Leader  
Follower

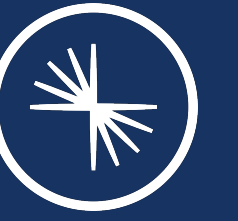






So far, this is

**Pretty good**

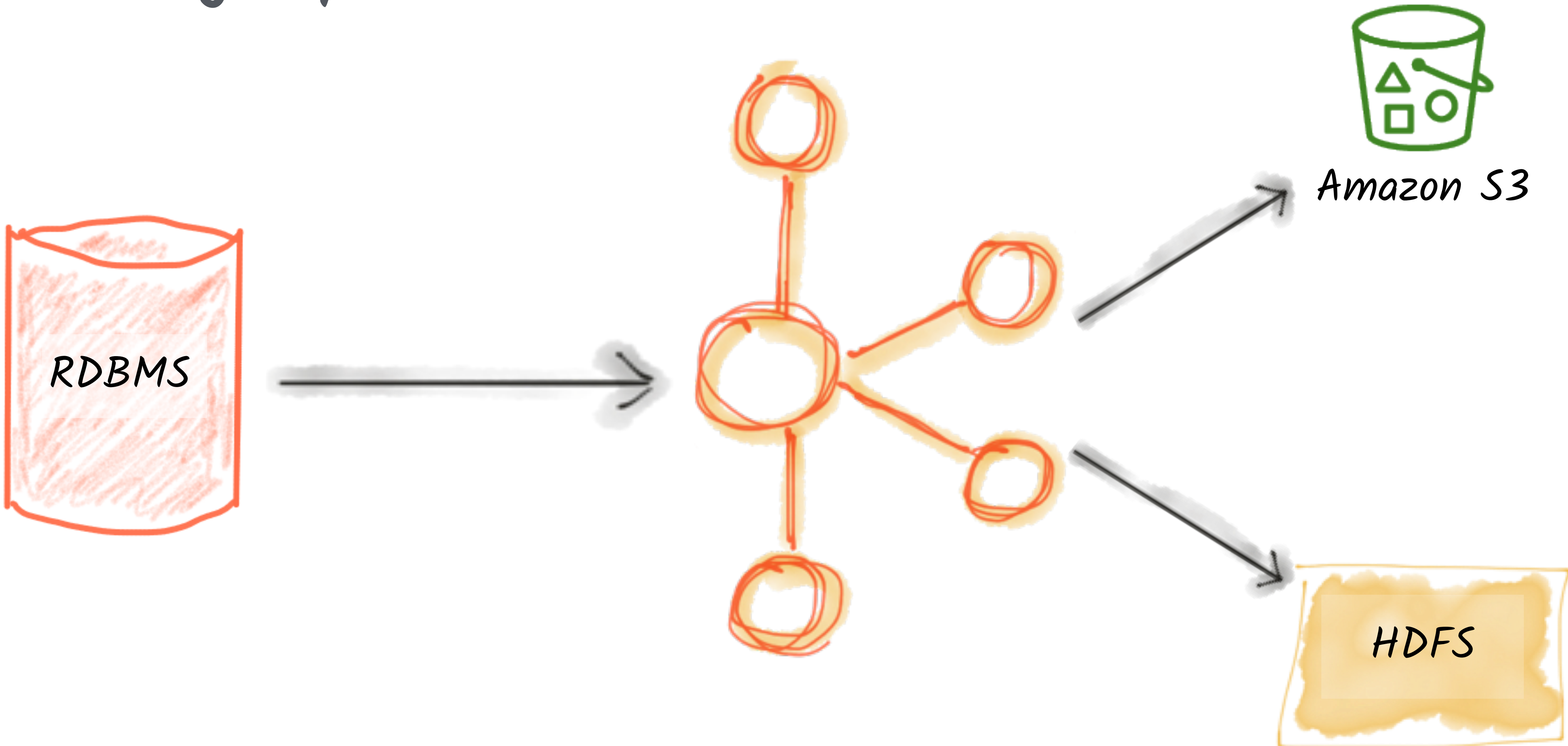


So far, this is

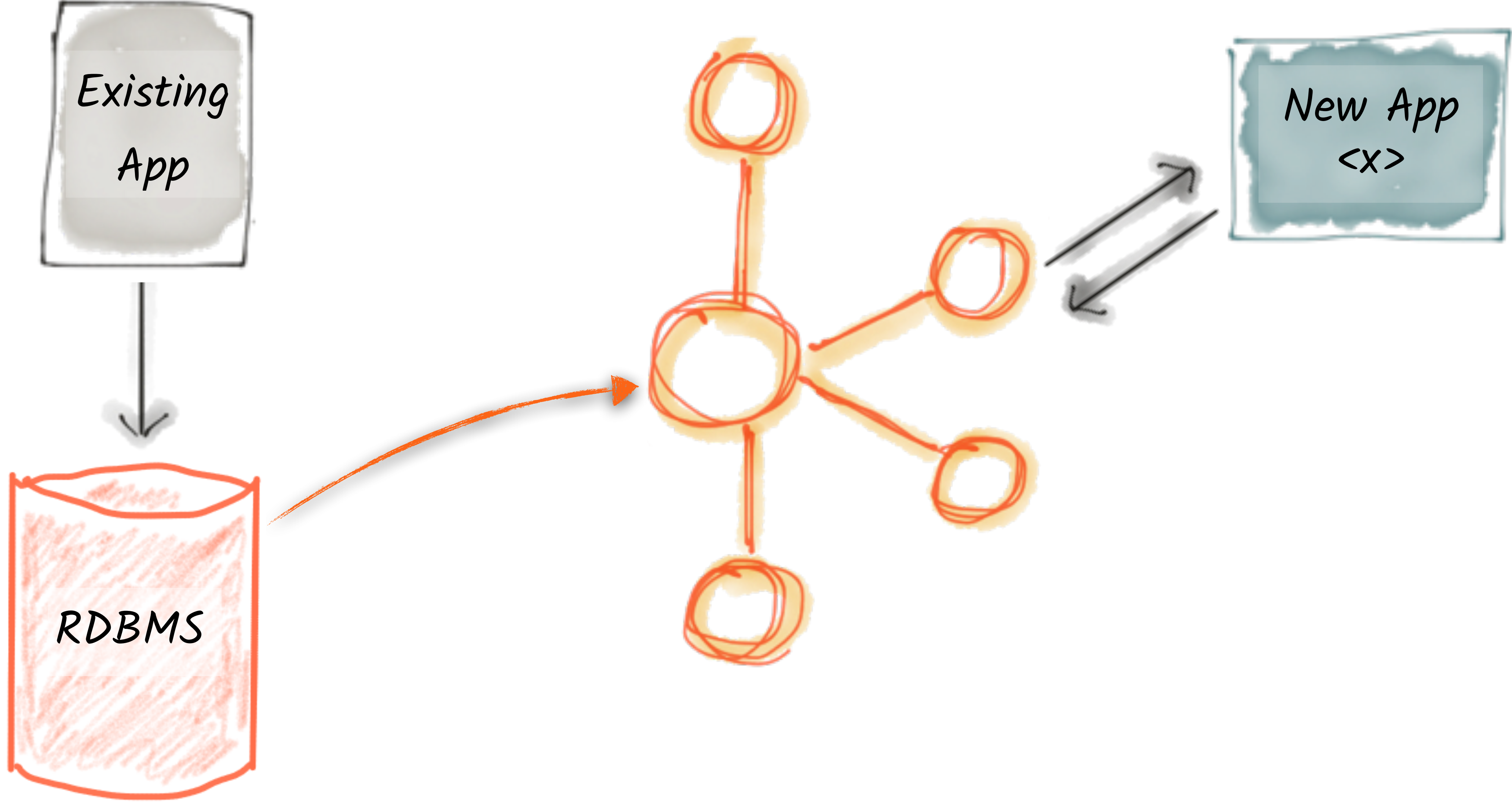
**Pretty good**

but I've not finished yet...

# Streaming Pipelines



# Evolve processing from old systems to new







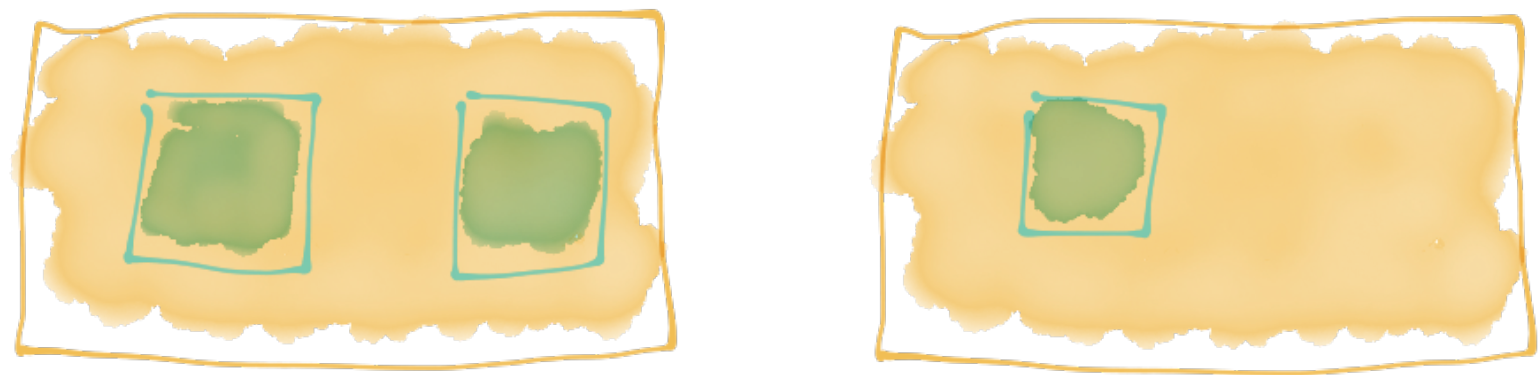


# Streaming Integration with Kafka Connect

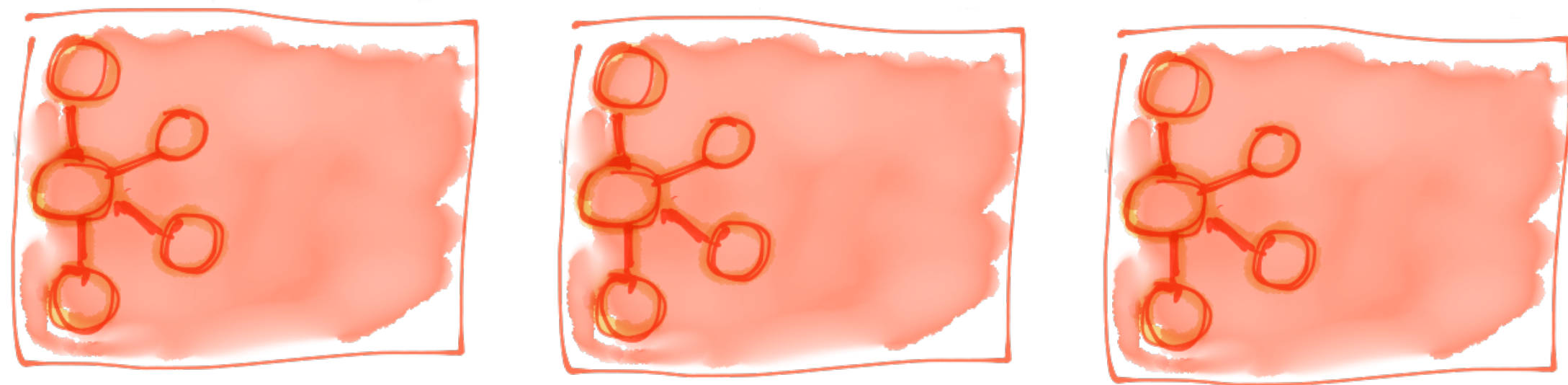


MySQL ORACLE syslog http:// CSV salesforce Microsoft SQL Server JSON MQTT

Sources



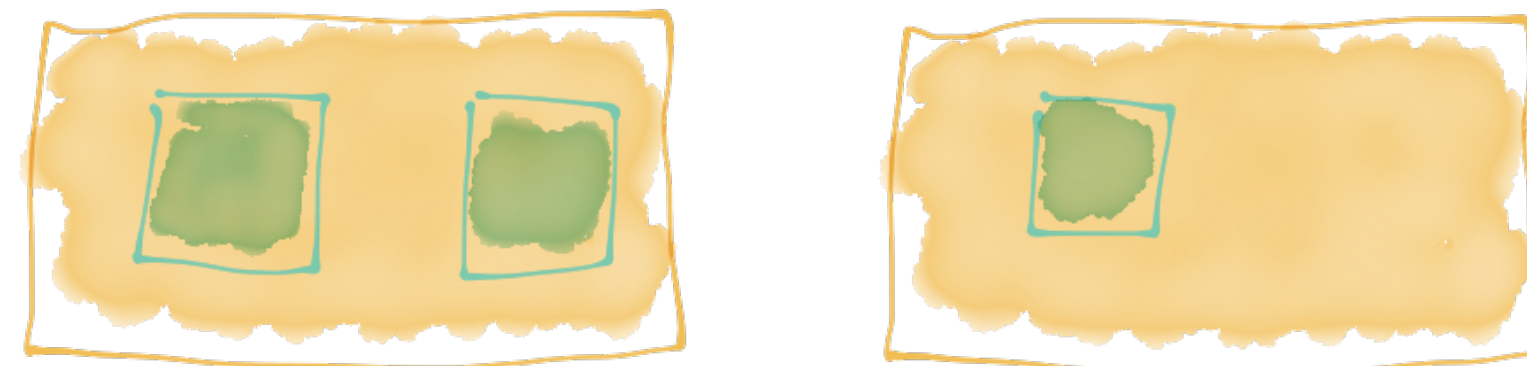
Kafka Connect



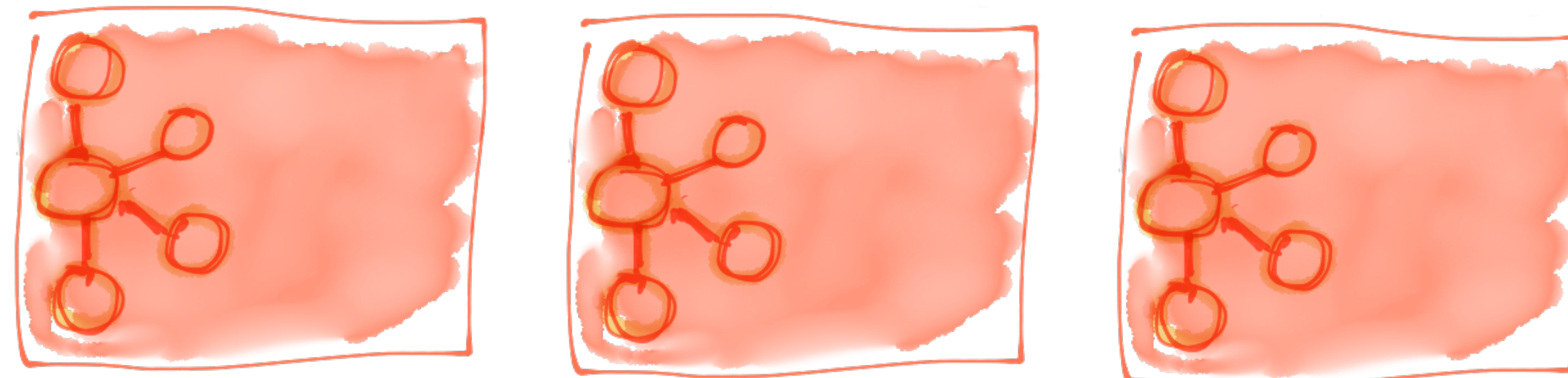
Kafka Brokers

# Streaming Integration with Kafka Connect

Sinks



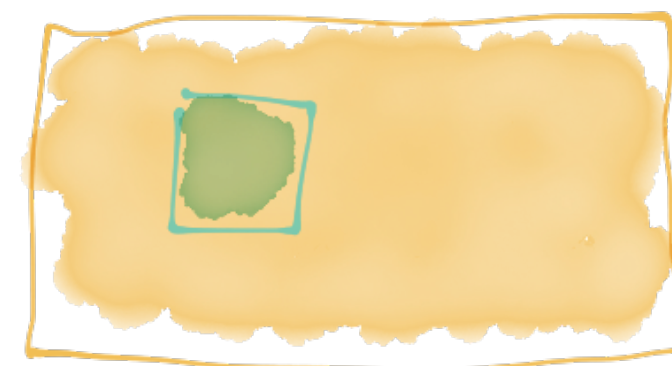
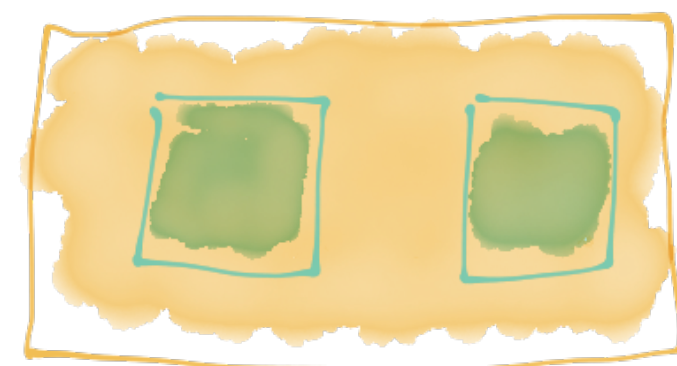
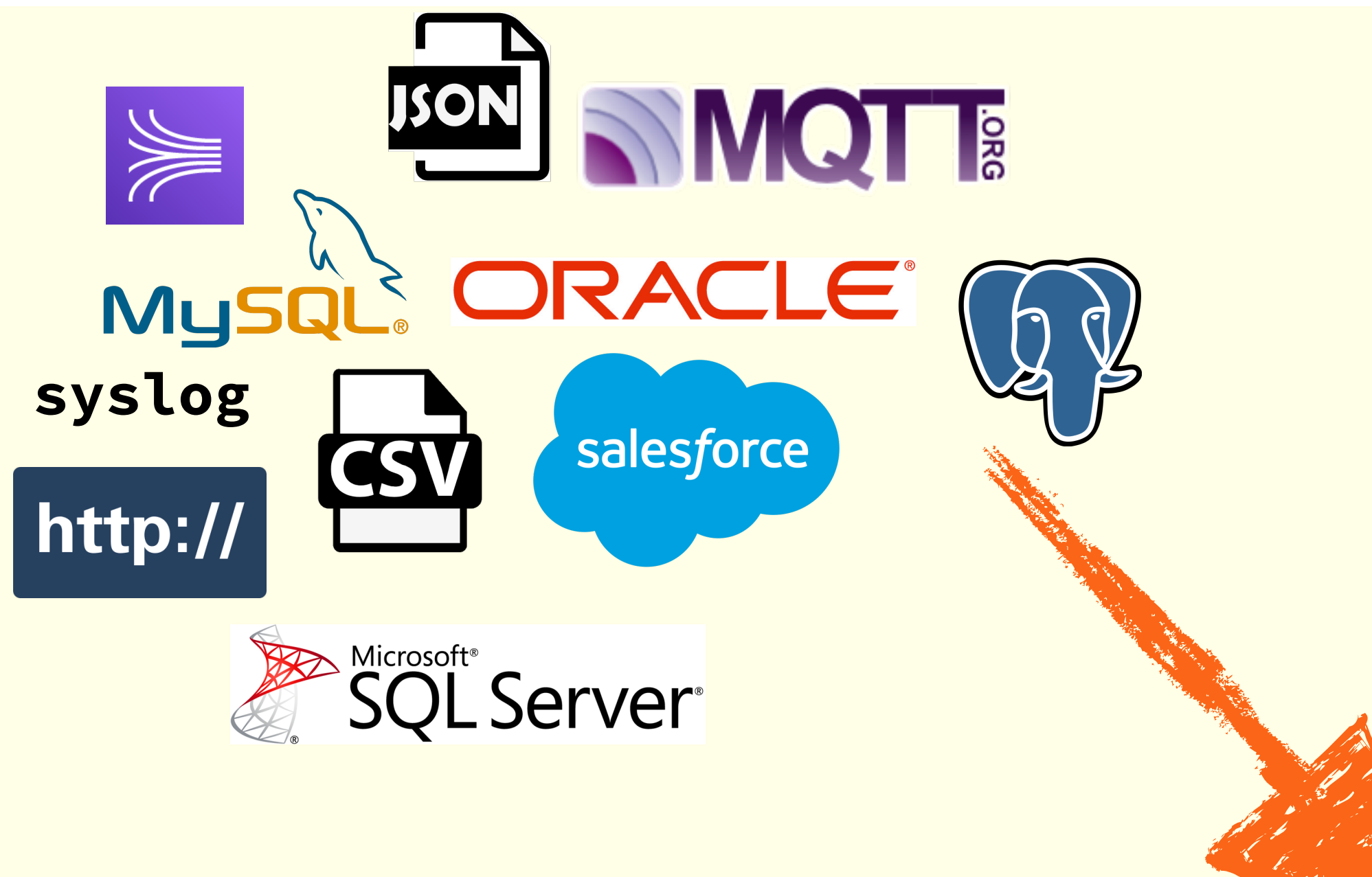
Kafka Connect



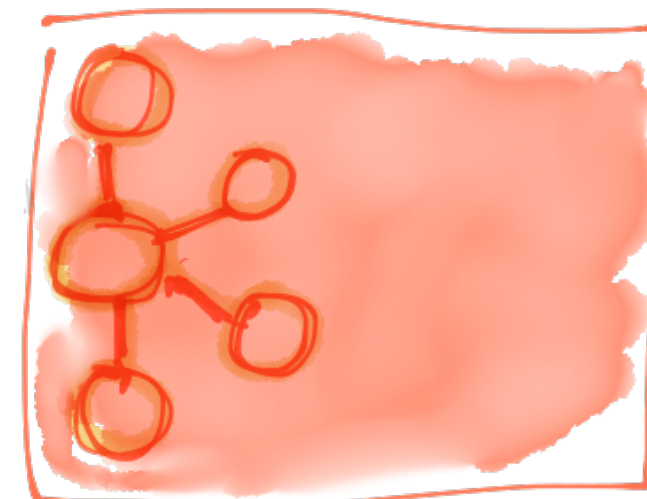
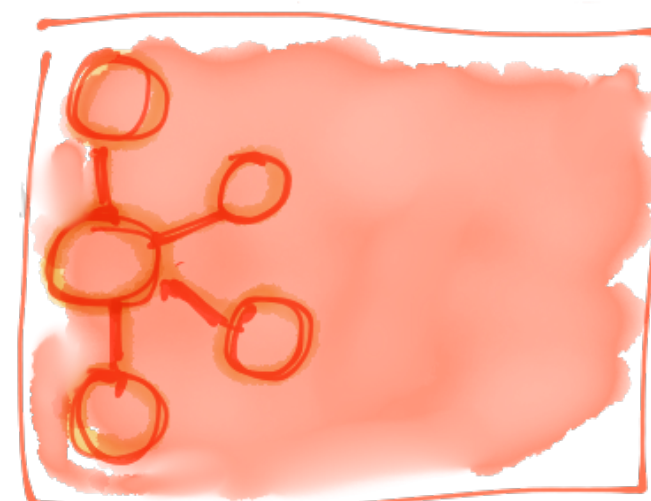
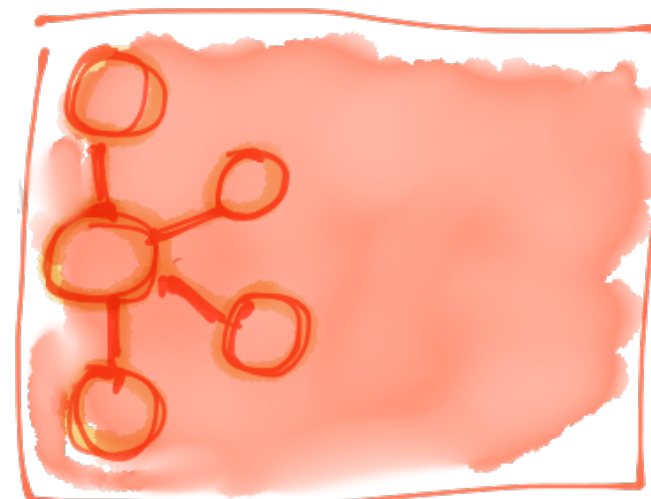
Kafka Brokers



# 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",
```

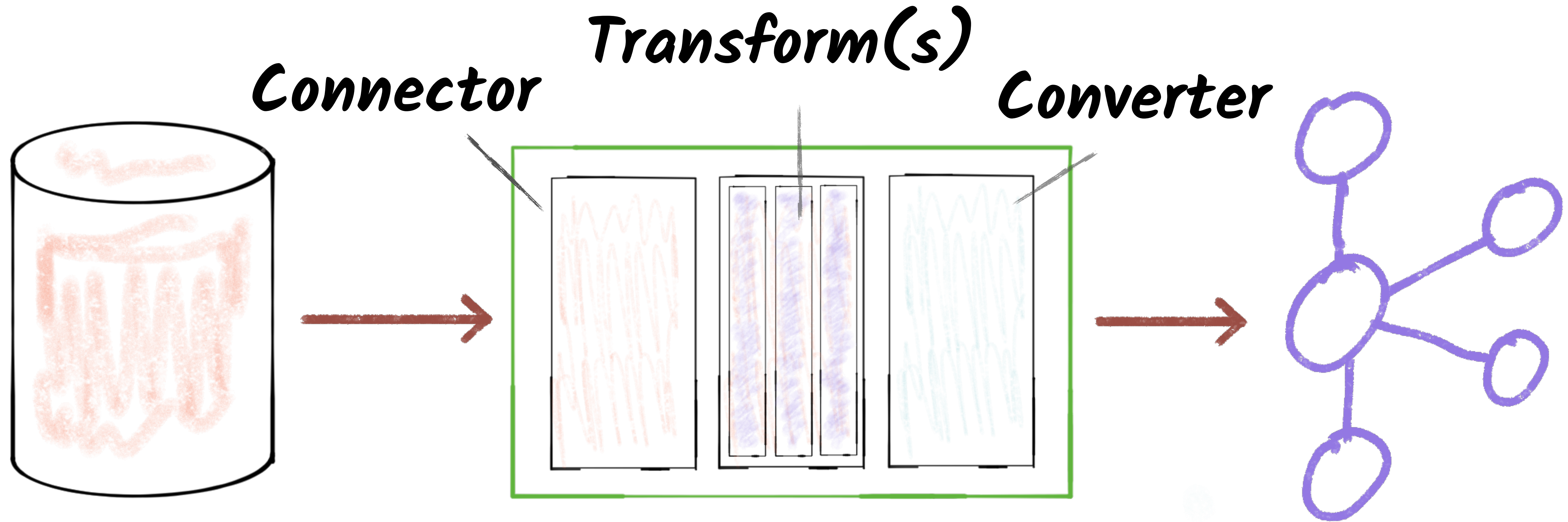
```
  "table.whitelist":
```

```
    "sales,orders,customers"
```

```
}
```



# Extensible



# hub.confluent.io

Confluent Hub

Discover Kafka® connectors  
and more

### Filters

Plugin type ⊕

- Sink
- Source
- Transform
- Converter

Enterprise support ⊕

- Confluent supported
- Partner supported
- None

### Results (1)


[+ Submit a plugin](#)

## Kafka Connect JDBC

SINK, SOURCE CONNECTOR

The JDBC source and sink connectors allow you to exchange data between relational databases and Kafka. The JDBC source connector allows you to import data from any relational database with a JDBC driver into Kafka topics

<b>Enterprise support:</b> Confluent supported	<b>Verification:</b> Confluent built	<b>License:</b> Free
<b>Installation:</b> Confluent Hub CLI, Download	<b>Author:</b> Confluent, Inc.	<b>Version:</b> 5.4.1



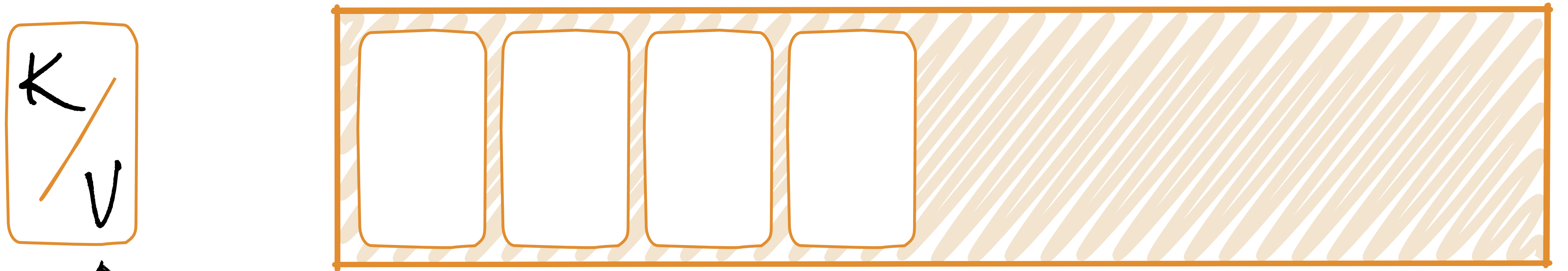












Wait...what's this?

# *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 Blvd 95610 38 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

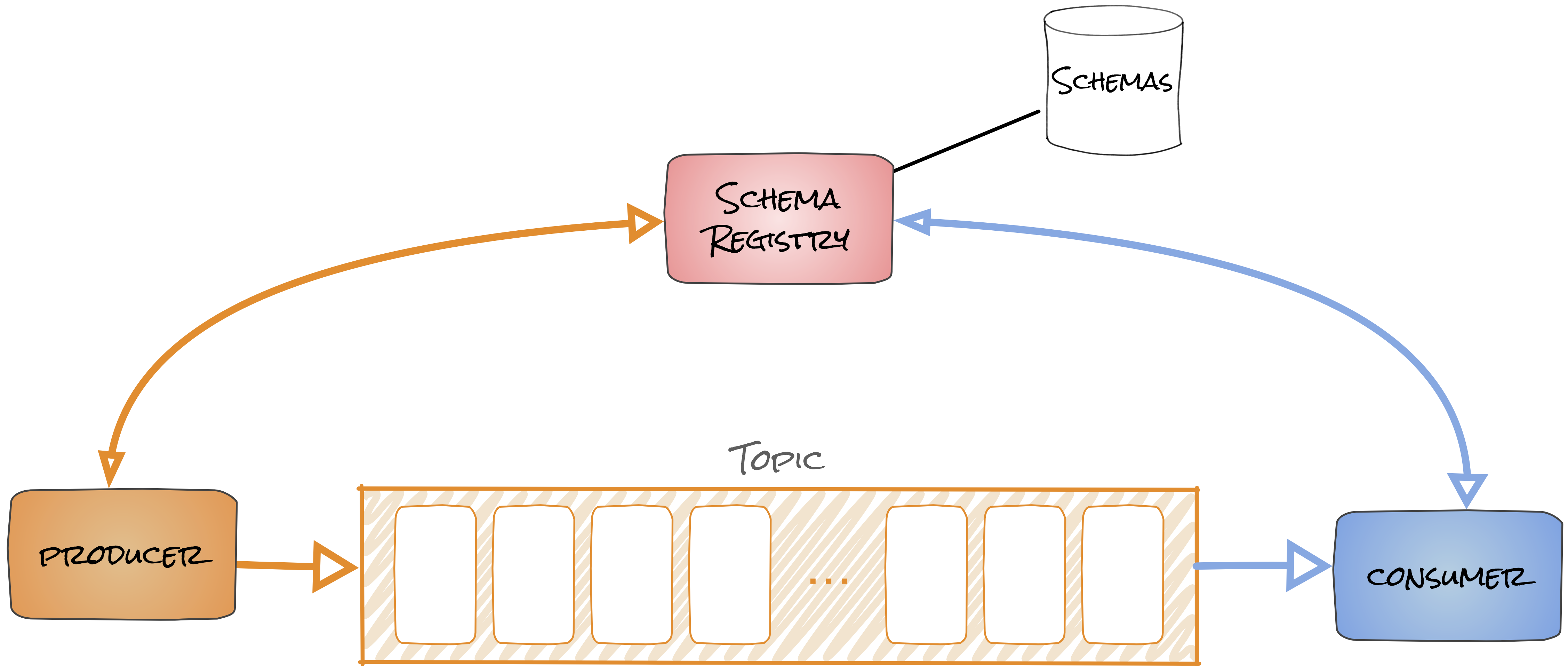


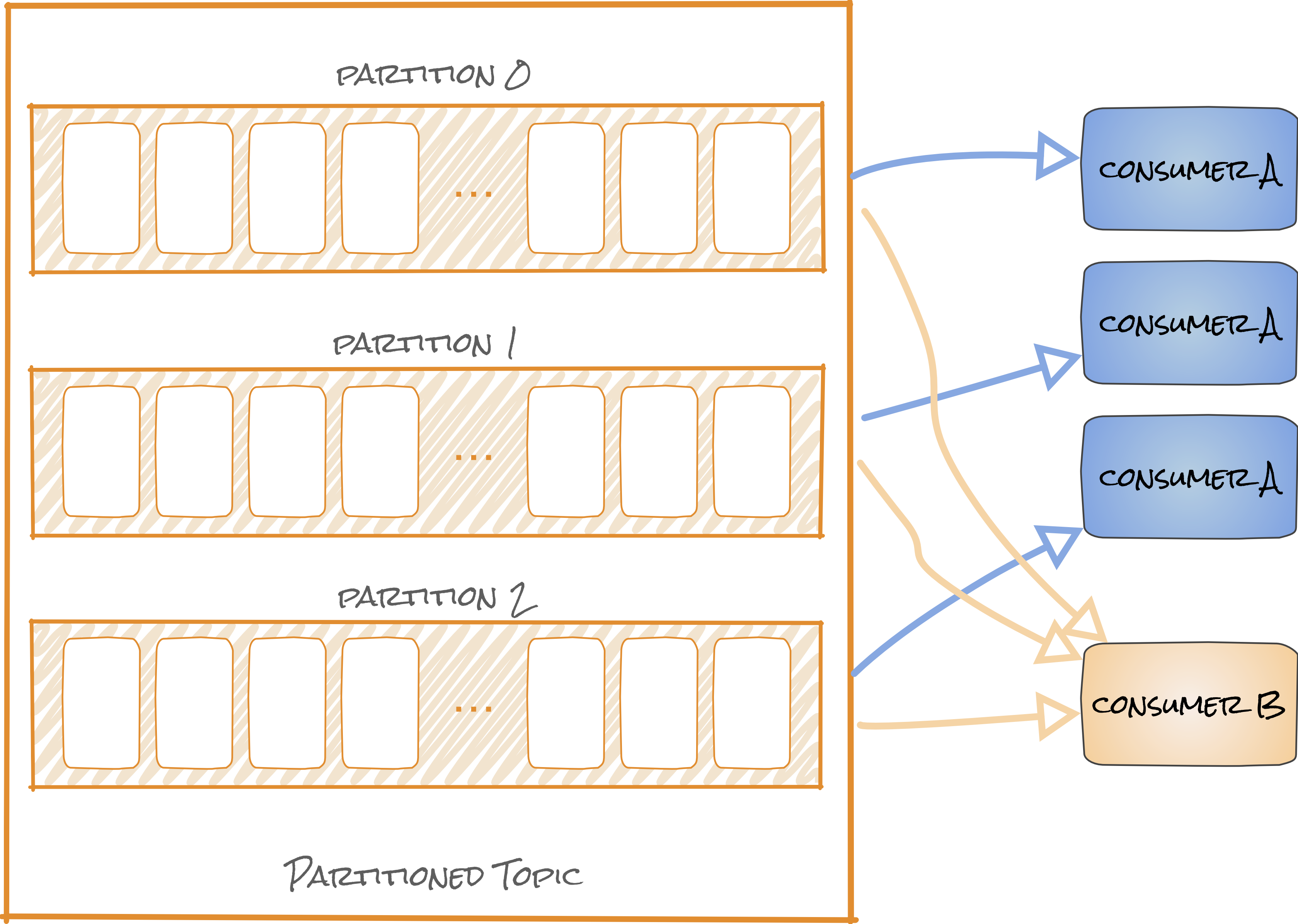
 **Gwen (Chen) Shapira**  
@gwenshap

If your dev process doesn't validate schema compatibility somewhere between your IDE and production - you are screwed and don't know it.

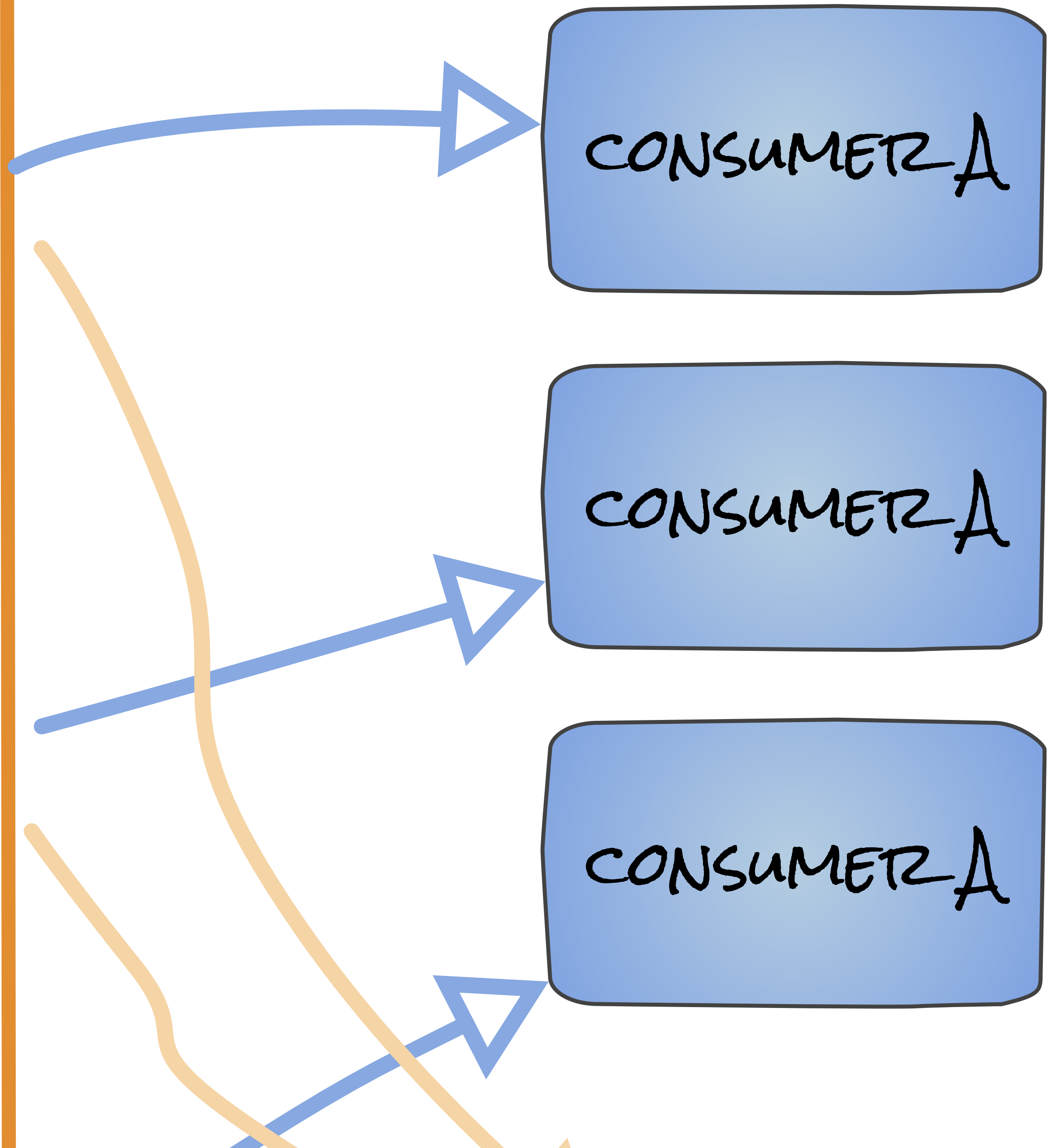
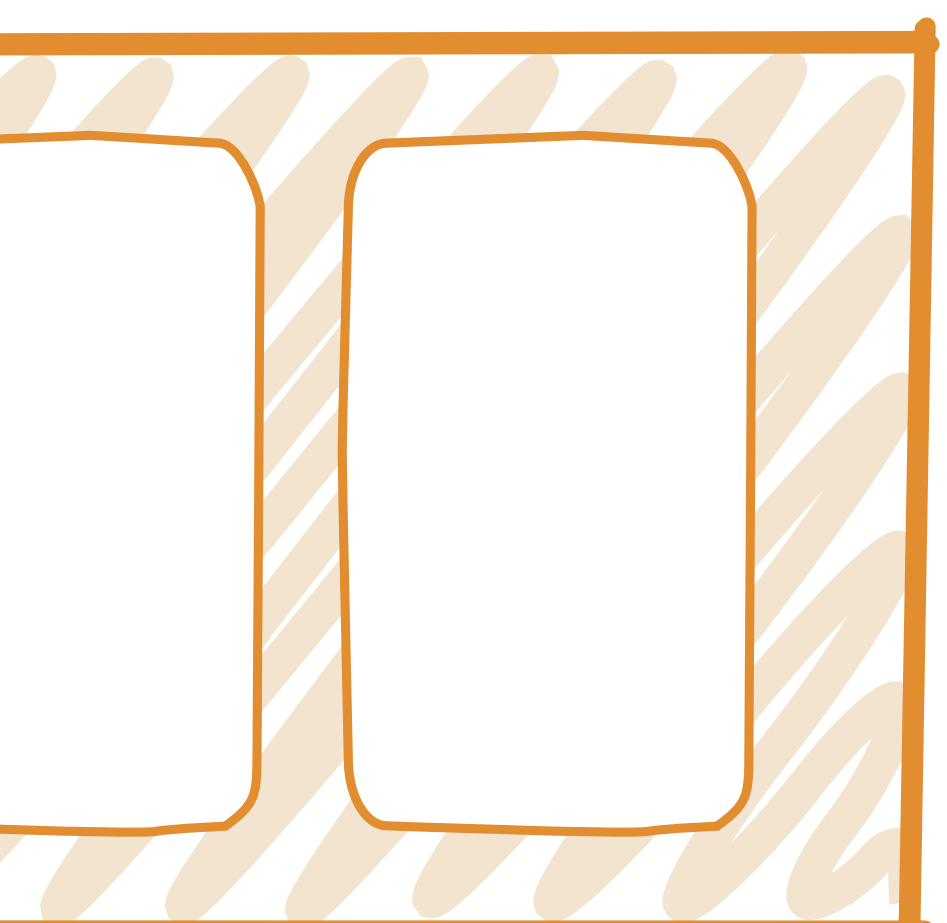
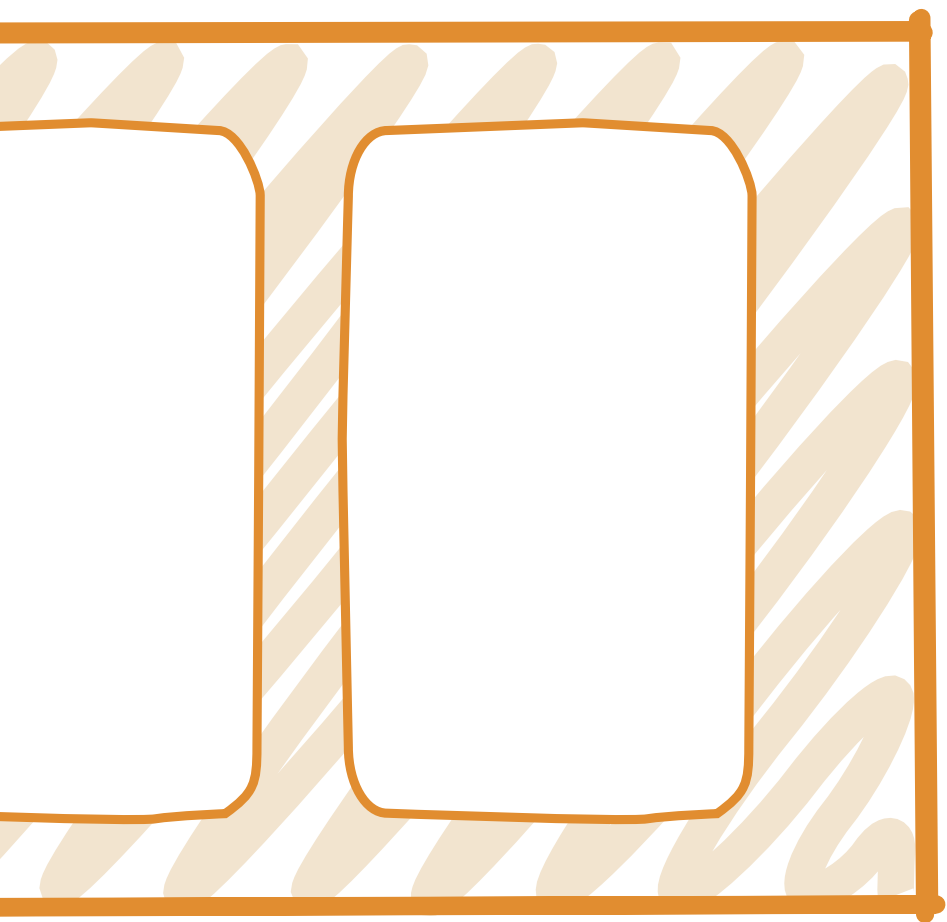
5:50 AM - 5 Apr 2017

<https://rmoff.dev/qcon-schemas>









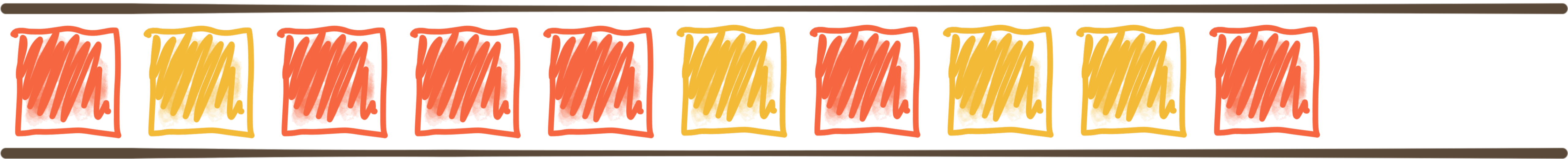


```
{  
  "reading_ts": "2020-02-14T12:19:27Z",  
  "sensor_id": "aa-101",  
  "production_line": "w01",  
  "widget_type": "acme94",  
  "temp_celcius": 23,  
  "widget_weight_g": 100  
}
```



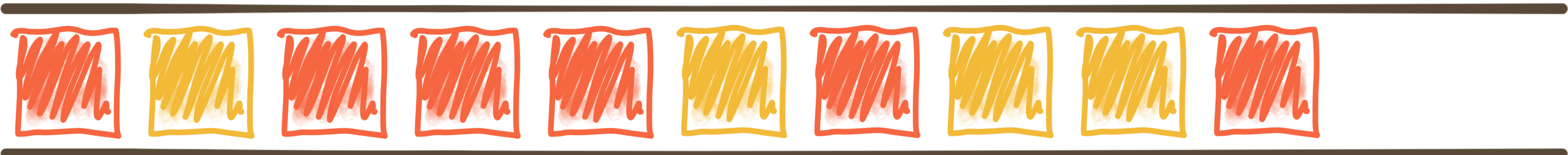


# Streams of events



Time →

# Stream Processing



Stream: widgets



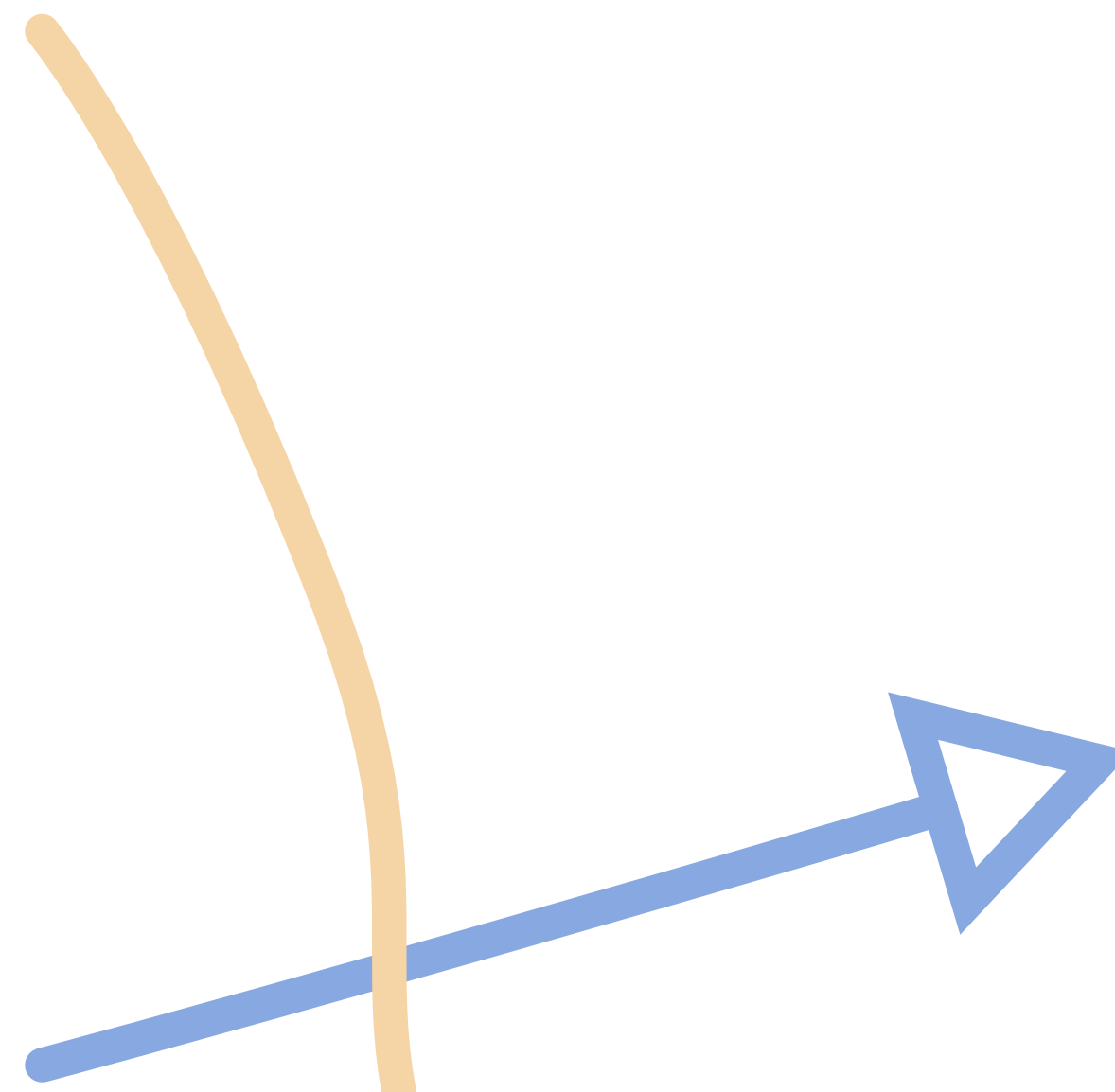
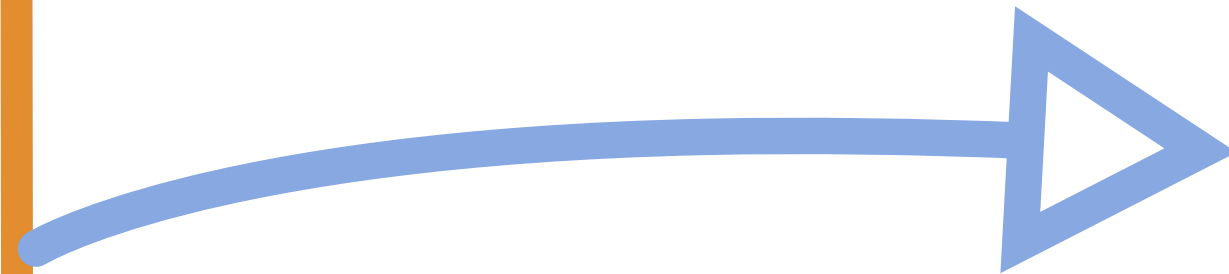
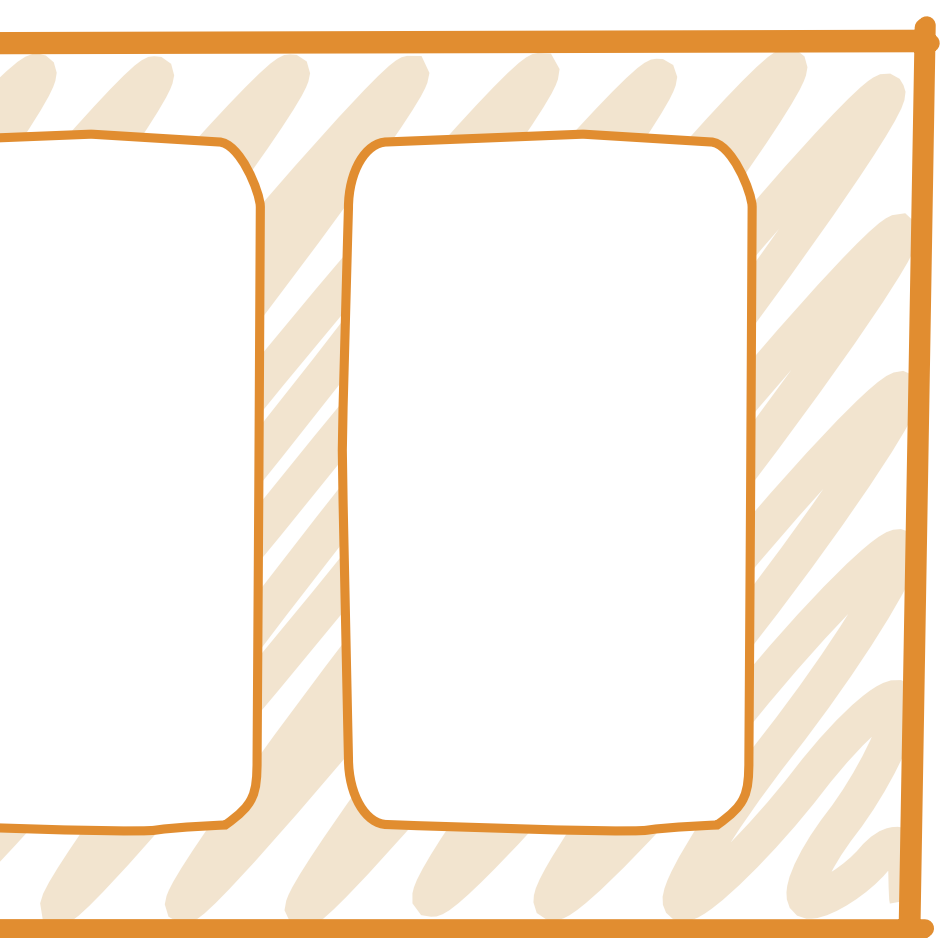
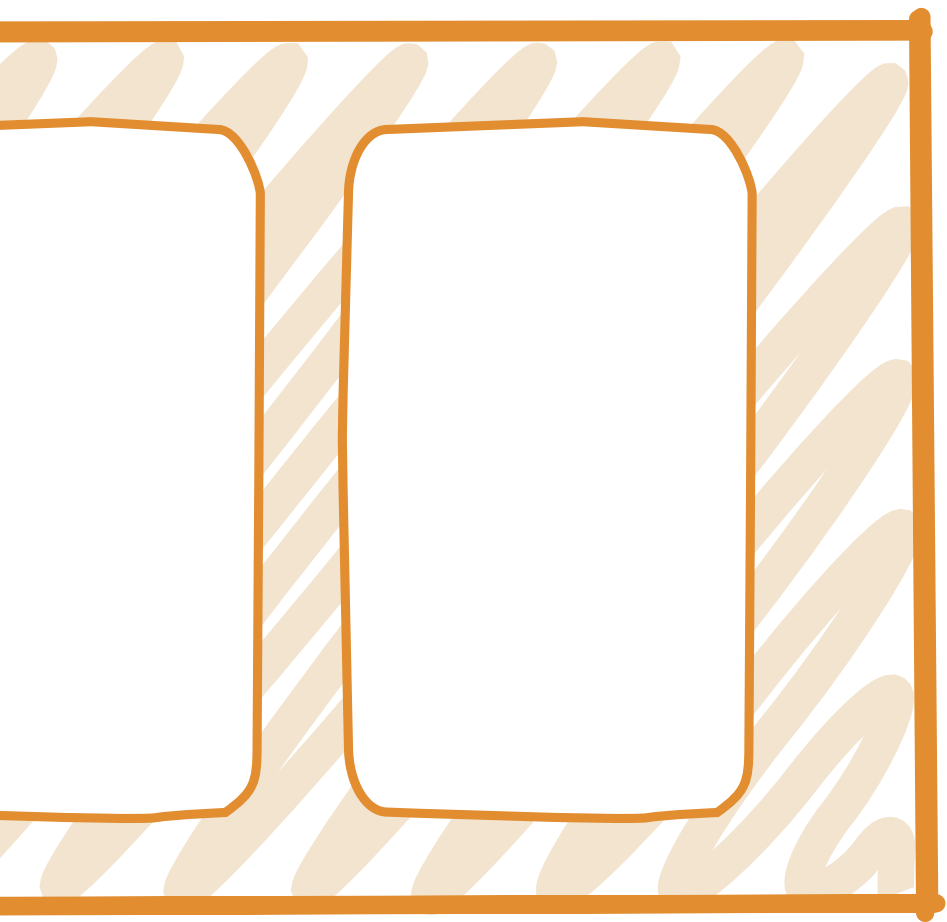
Stream: widgets\_red

# Stream Processing with Kafka Streams



```
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));
```





STREAMS  
APPLICATION

STREAMS  
APPLICATION

STREAMS  
APPLICATION



# Stream Processing with ksqlDB



ksqlDB

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





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







```
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
```

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

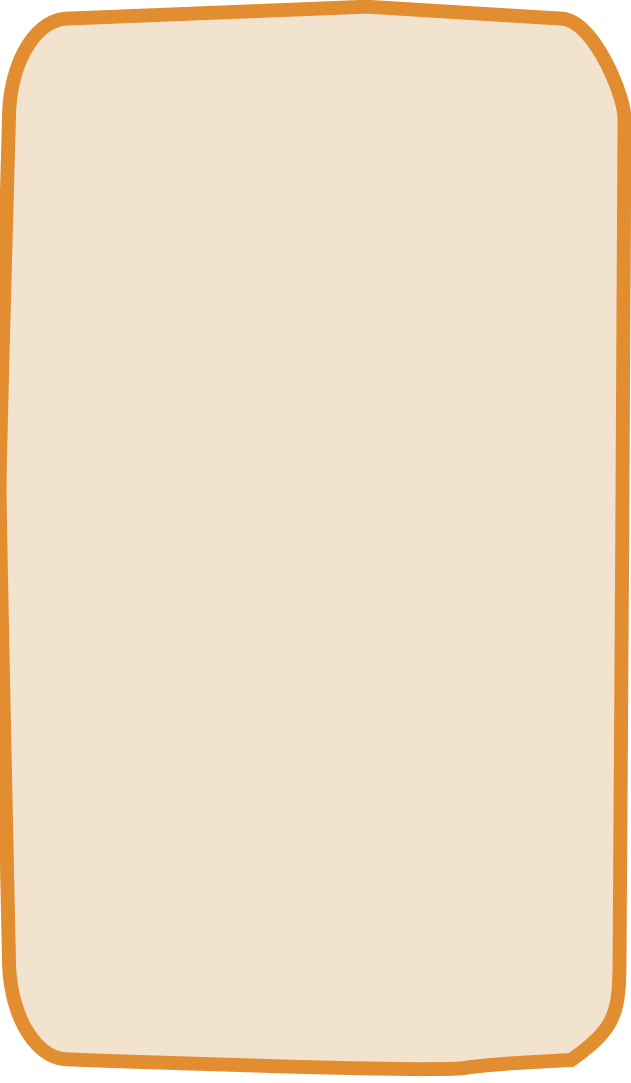


*Object store,  
data warehouse,  
RDBMS*

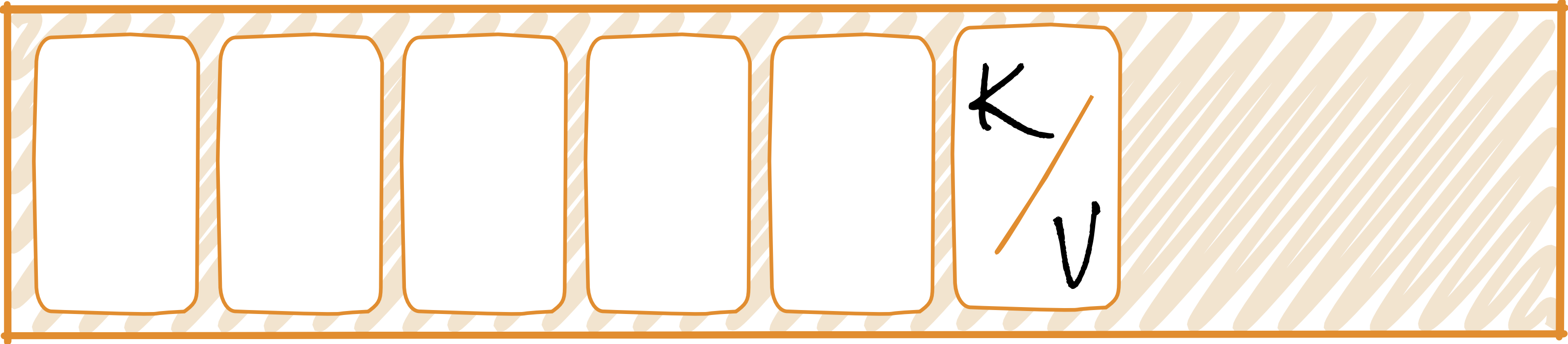
```
CREATE SINK CONNECTOR dw WITH (  
  'connector.class' = 'S3Connector',  
  'topics' = 'widgets'  
...);
```

# Summary

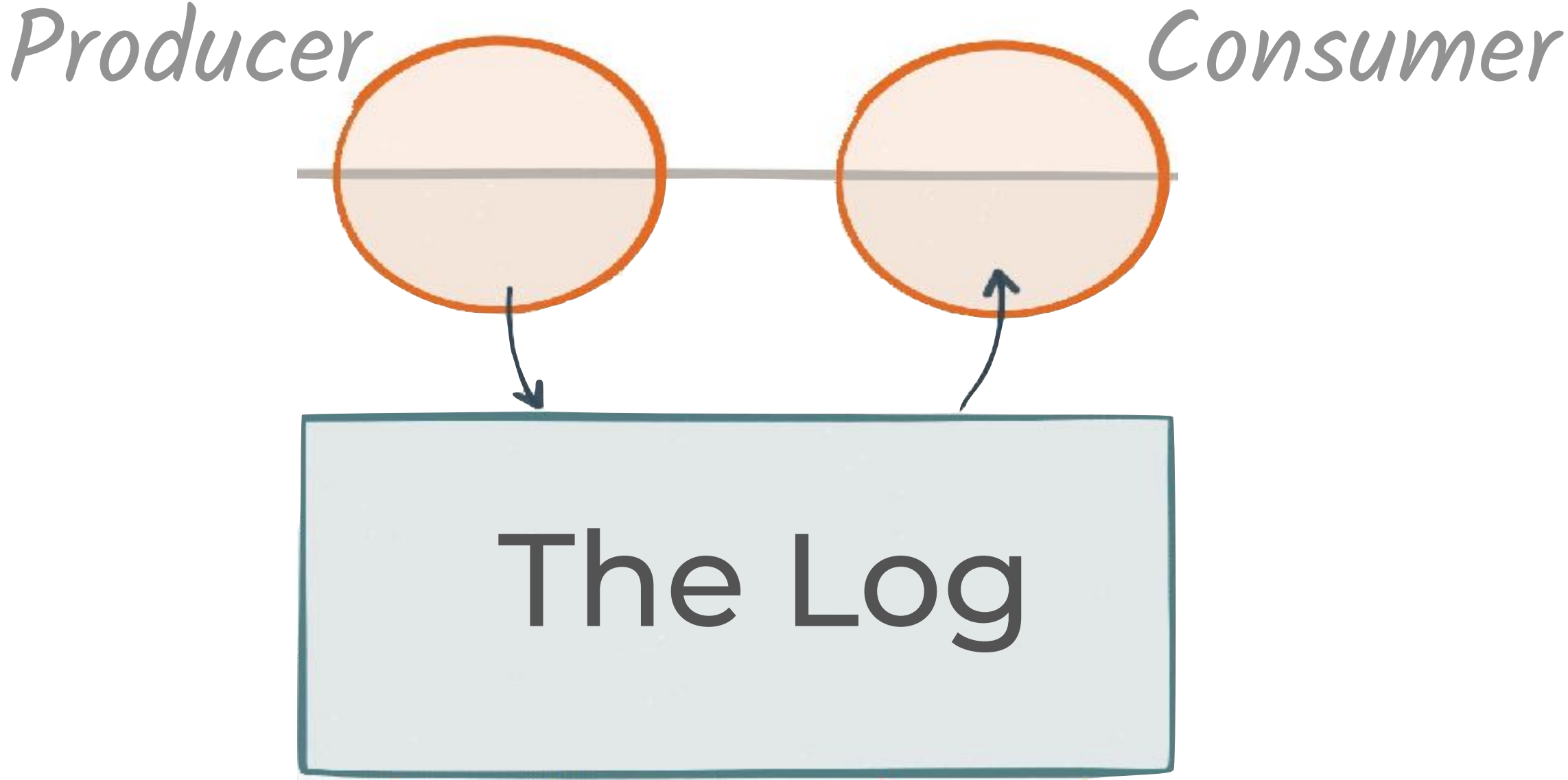


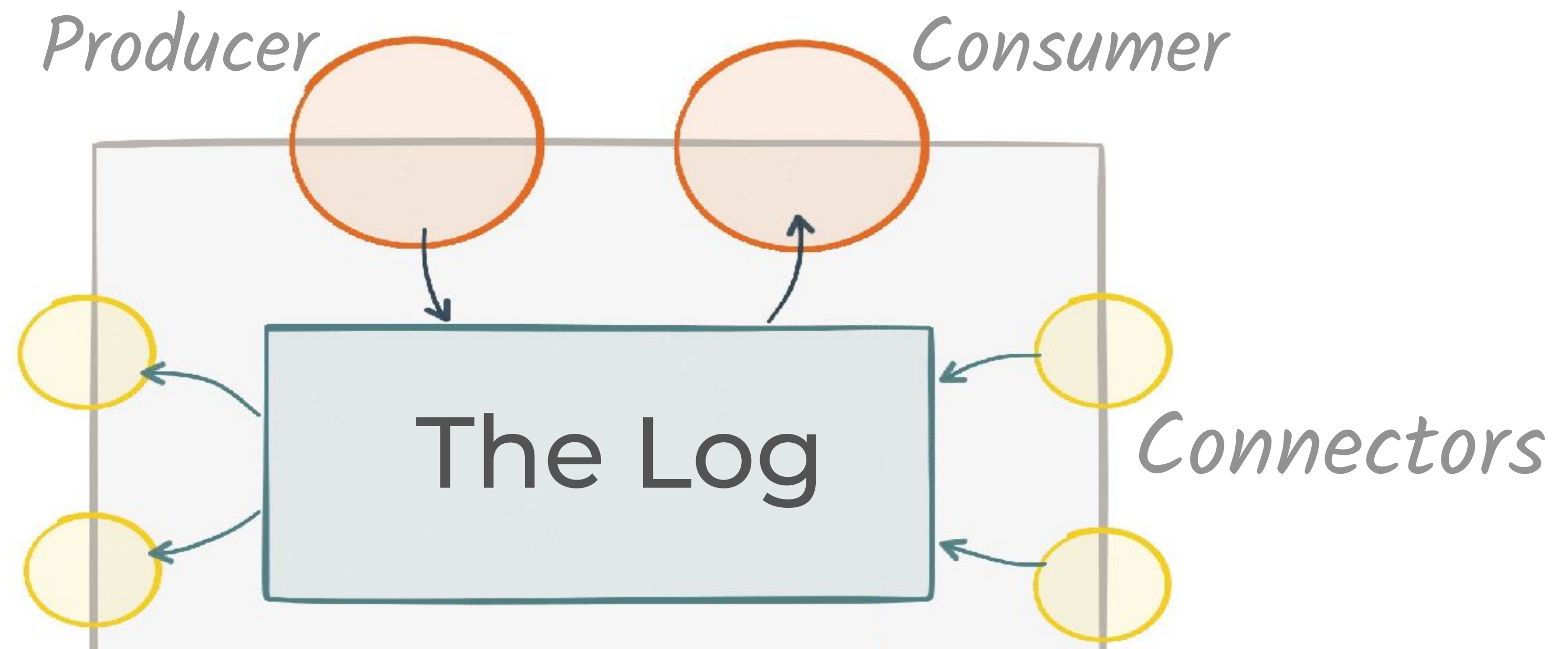


K  
/  
V

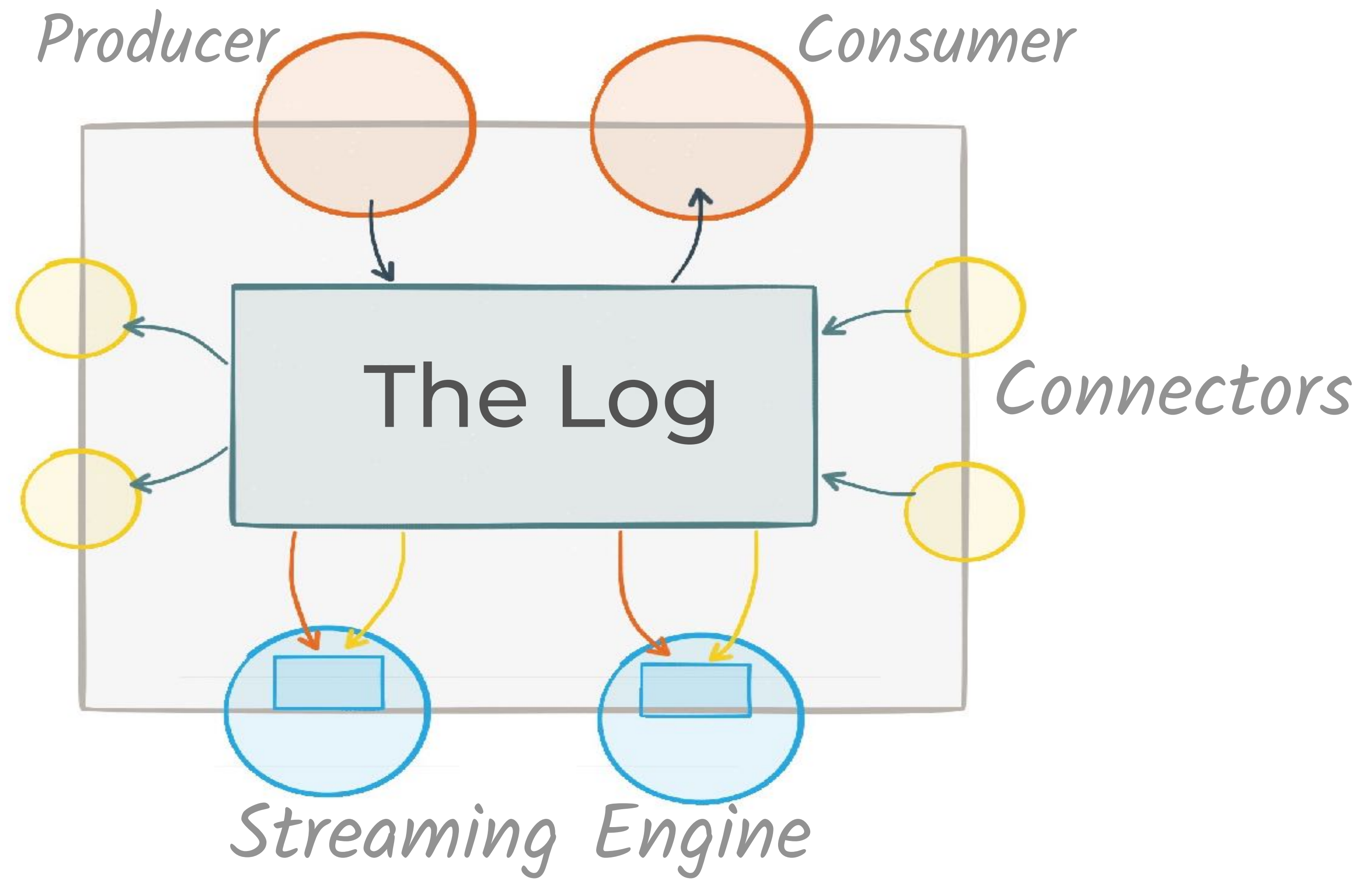


# The Log

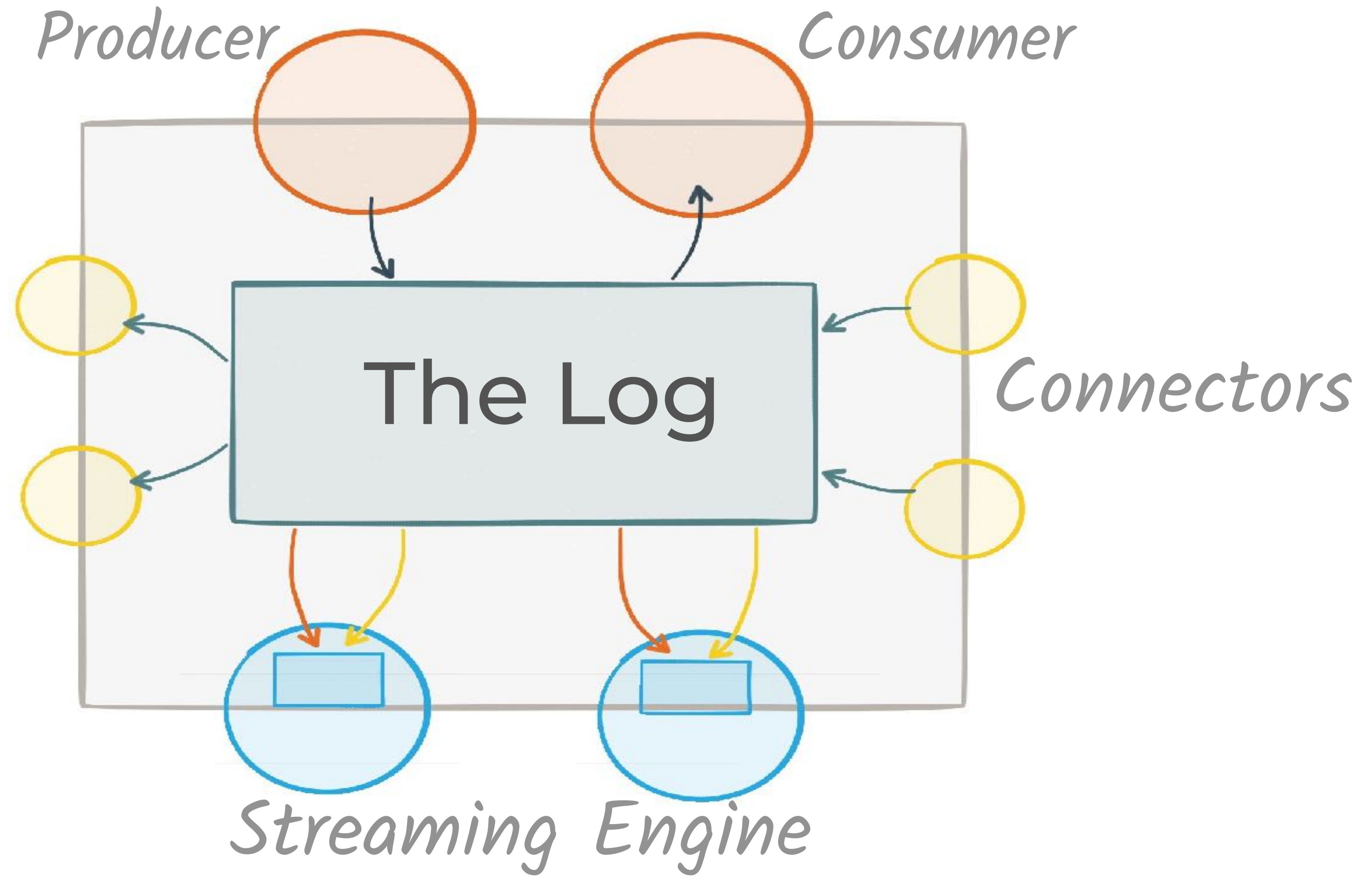
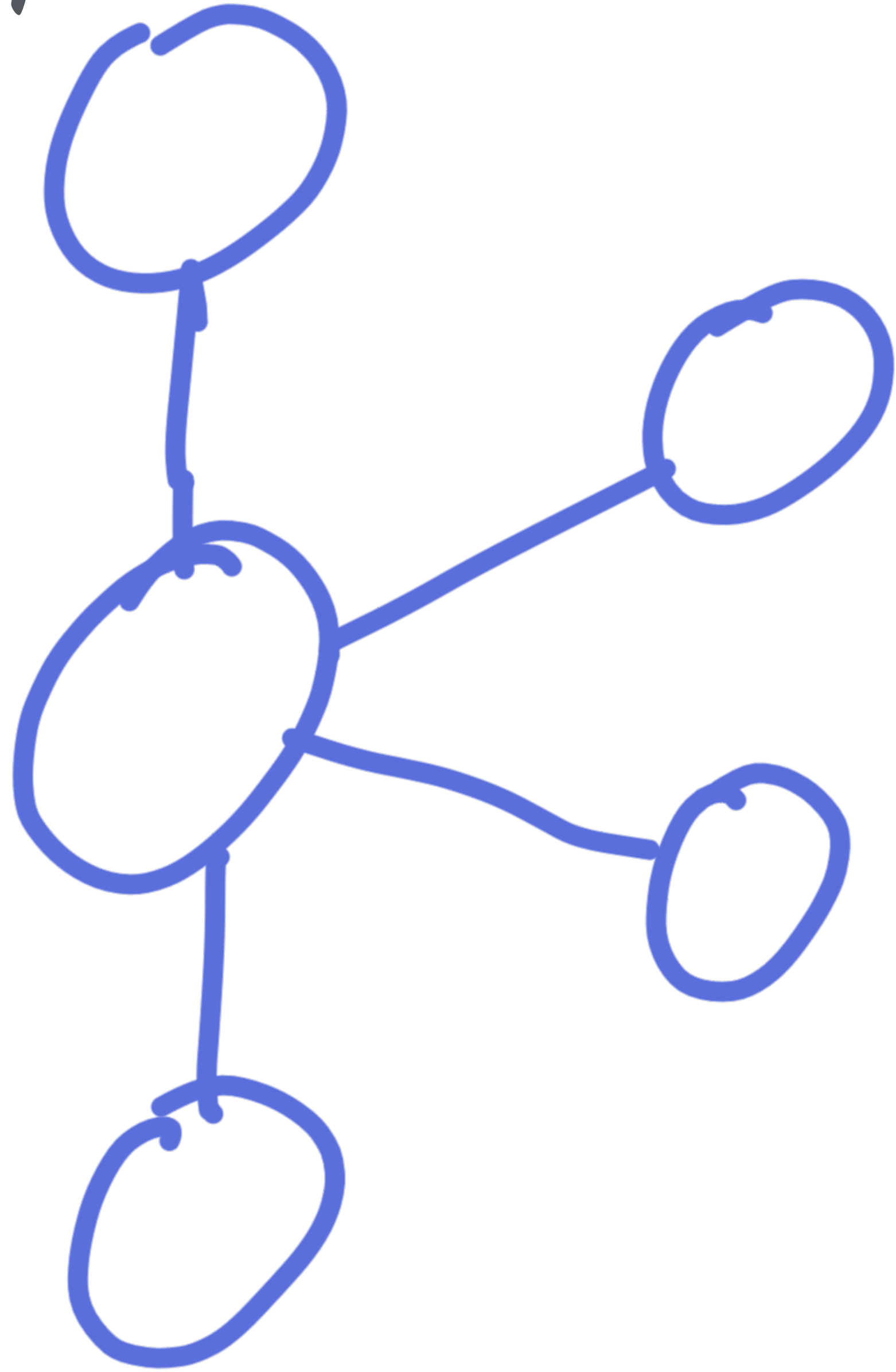






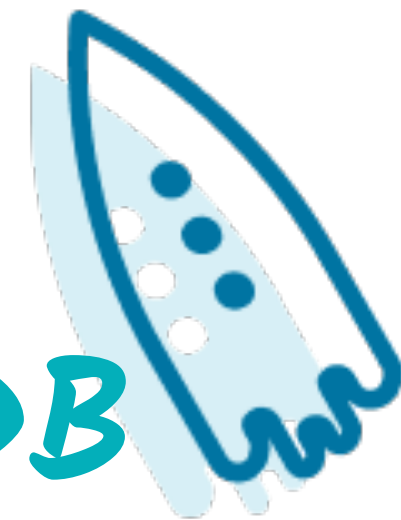
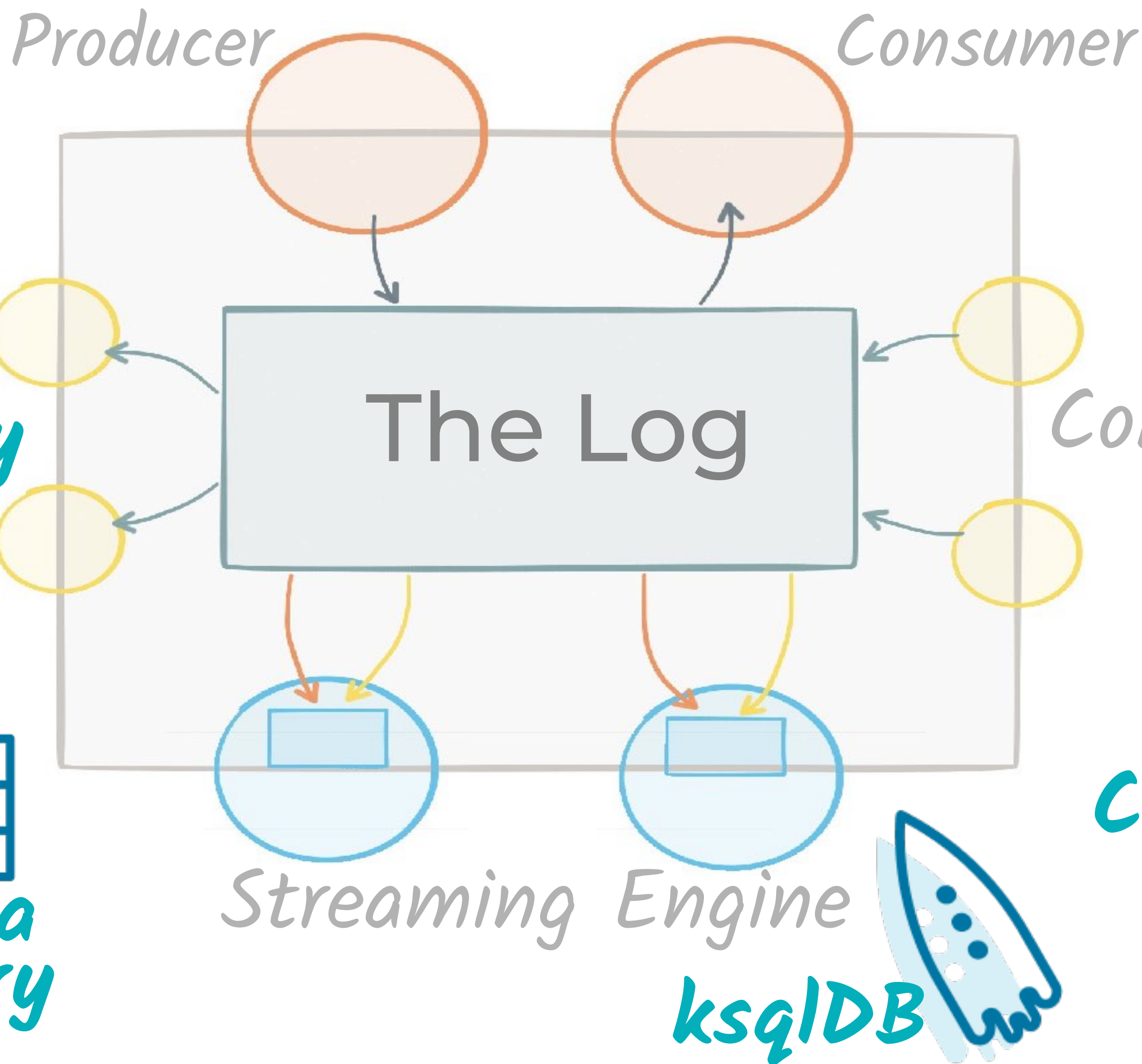
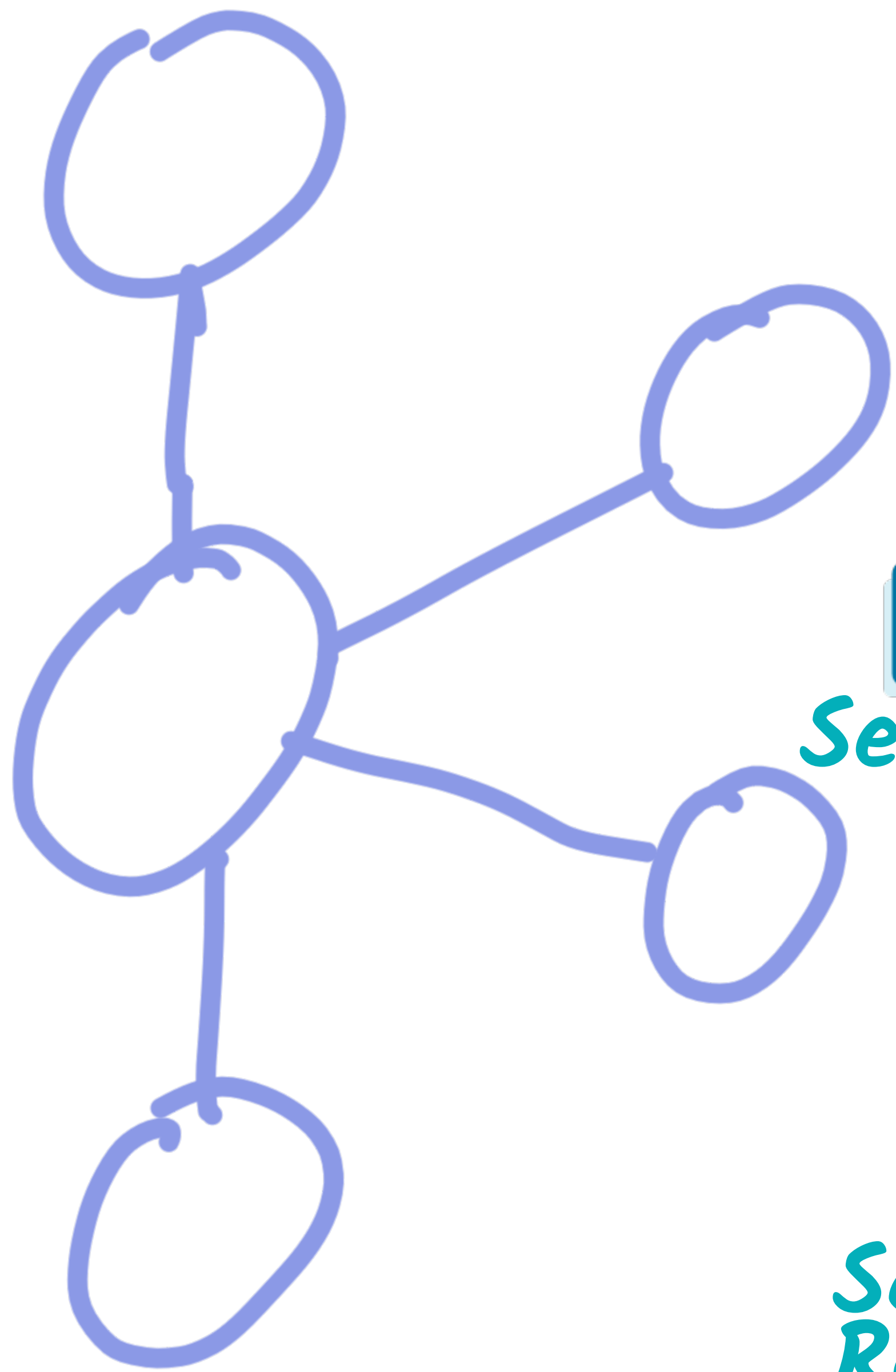


# Apache Kafka





# CONFLUENT





60DEVADV

\$200 USD off your bill each calendar month  
for the first three months when you sign up

<https://rmoff.dev/cc1oud>

Free money!

(additional \$60 towards  
your bill 😊)



confluent cloud

# Fully Managed Kafka as a Service





# Learn Kafka.

Start building with  
Apache Kafka at  
Confluent Developer.



[developer.confluent.io](https://developer.confluent.io)



**#**

**EOF**

*@rmoff*

*rmoff.dev/talks*

*youtube.com/rmoff*