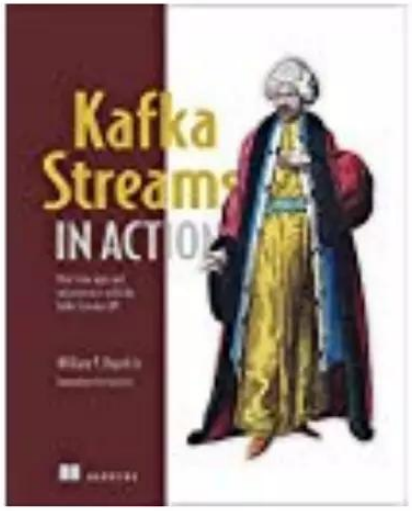
CONFLUENT

Introduction to Kafka Streams

Bill Bejeck, Integration Architect

Nice to meet you!

- Integration Architect on DevX team
- Prior to DevX ~3 years as engineer on Kafka Streams team
- Apache Kafka® Committer
- Author of “Kafka Streams in Action”



@bbejeck

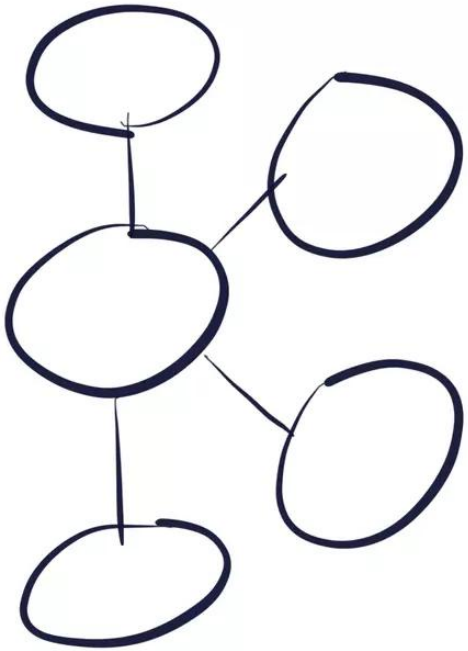
Agenda

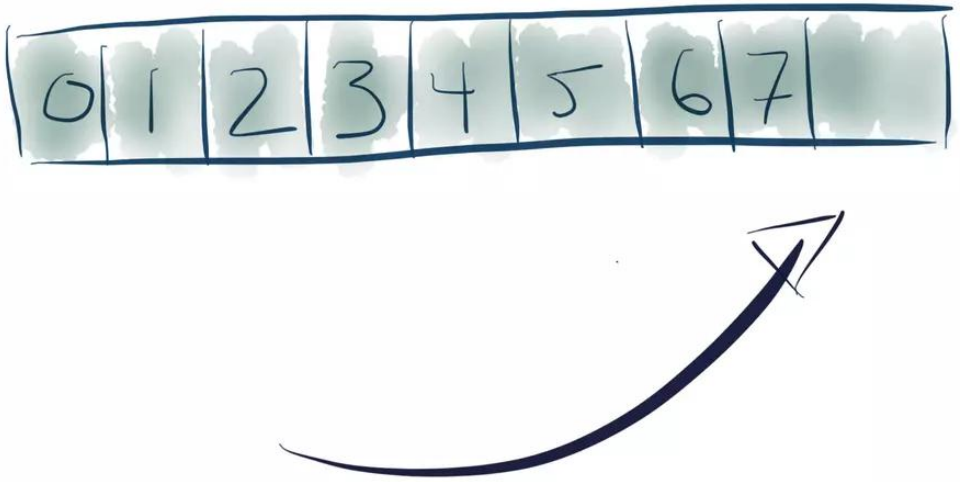
- Kafka Overview
- Life before Kafka Streams
- Kafka Streams Topology
- Kafka Streams Concepts/Architecture
- Kafka Streams API/Features

CONFLUENT

Kafka Overview

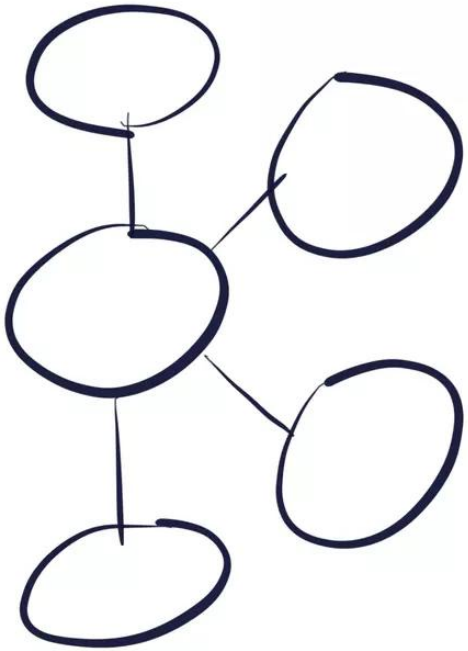
Kafka Overview

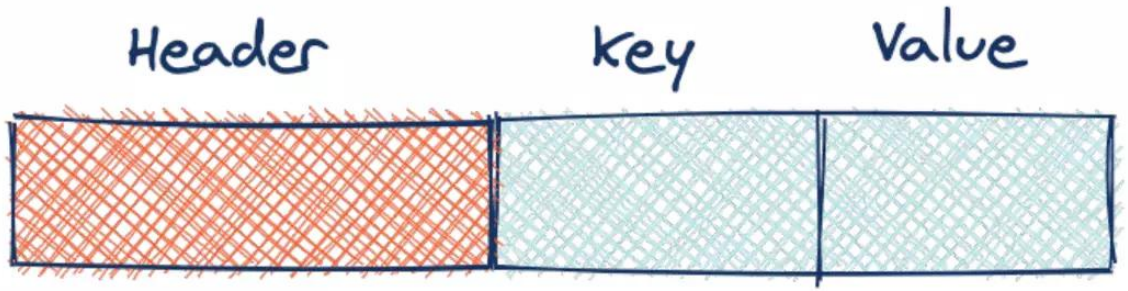




@bbejeck

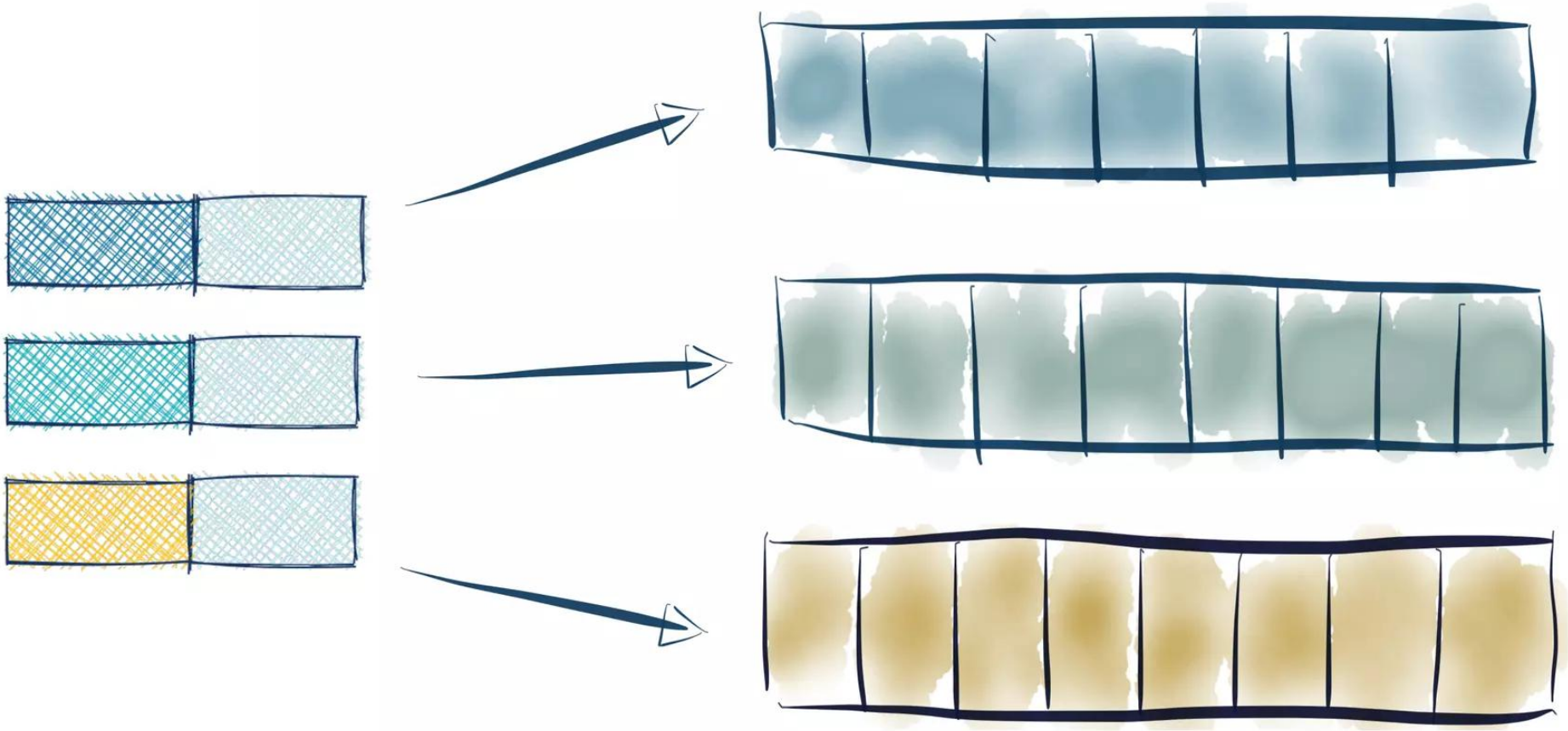
Kafka Overview





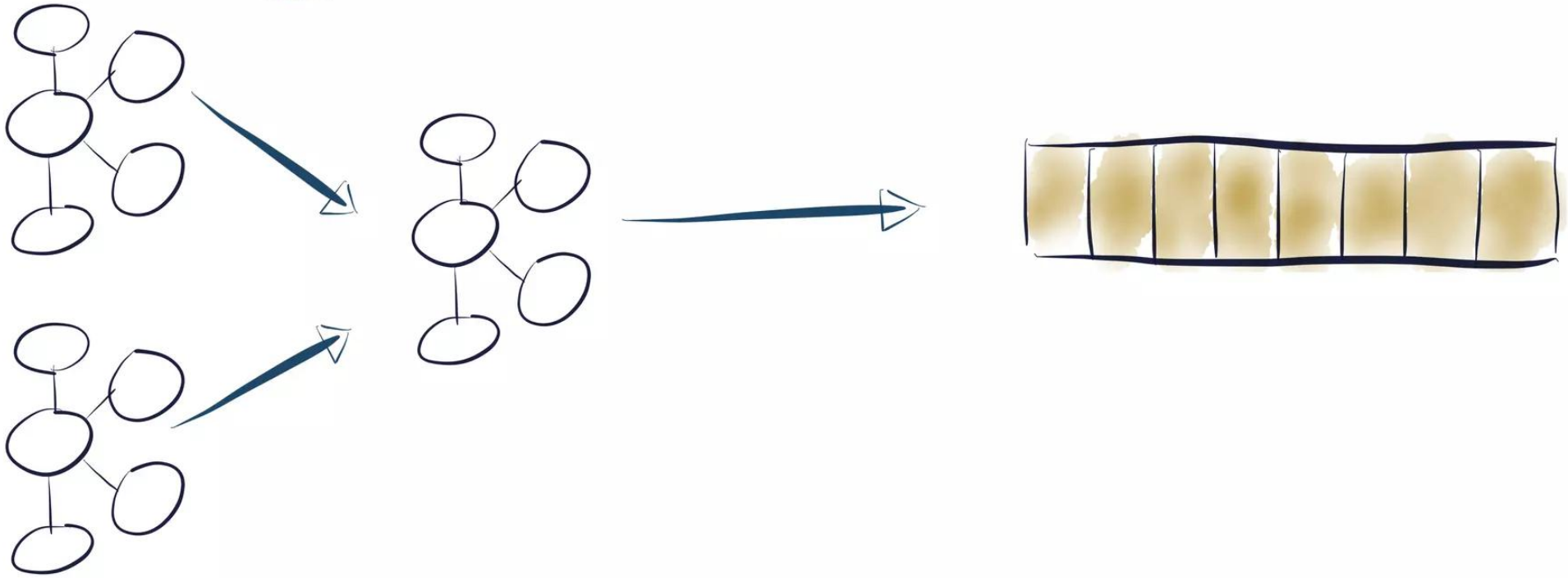
@bbejeck

Kafka Overview



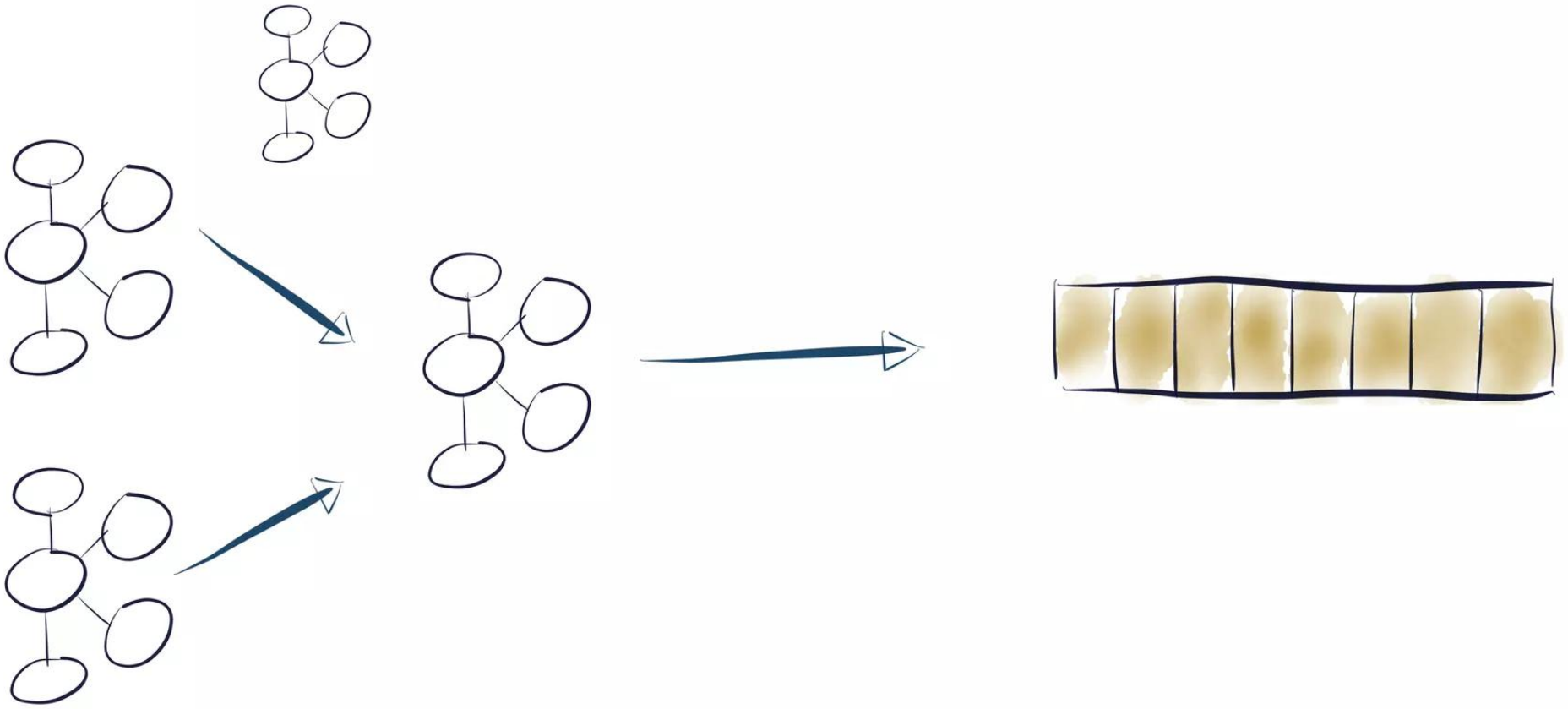
@bbejeck

Kafka Overview



@bbejeck

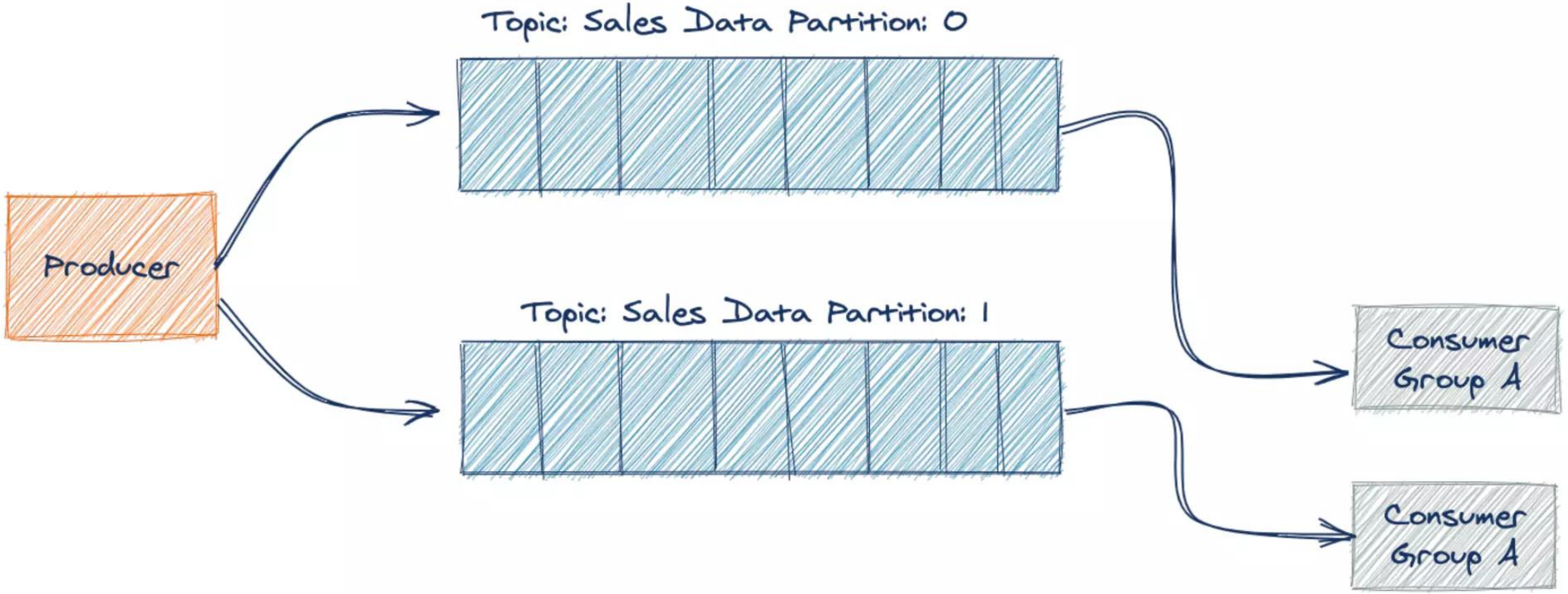
Kafka Overview



https://cwiki.apache.org/confluence/display/KAFKA/KIP-500%3A+Replace+ZooKeeper+with+a+Self-Managed+Metadata+Quorum

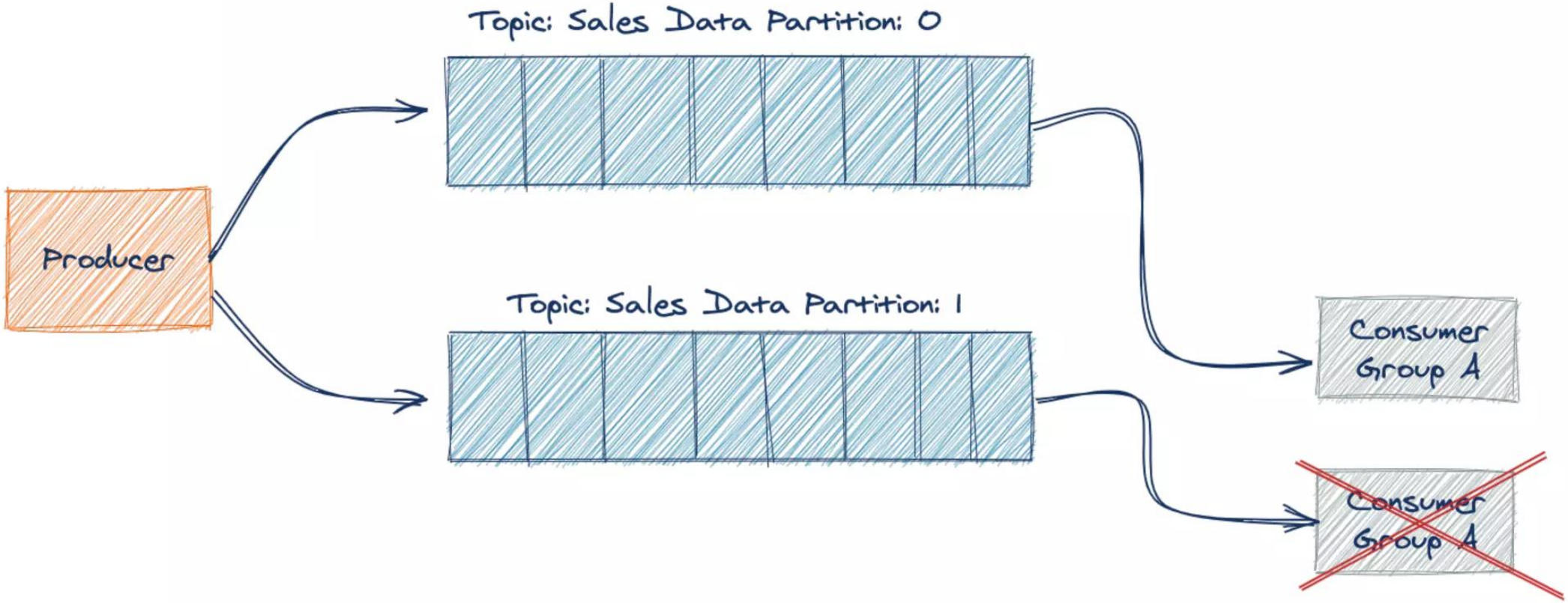
@bbejeck

Kafka Overview - Clients



@bbejeck

Kafka Overview – Clients



@bbejeck

Kafka Overview – Clients

@bbejeck

Life before Kafka Streams

Life Before Kafka Streams – GroupBy Example

```
public static void main(String[] args) {
        int counter = 0;
        int sendInterval = 15;

 Map<String, Integer> groupByCounts = new HashMap<>();

try(..consumer = new KafkaConsumer<>(consumerProperties());
   ..producer = new KafkaProducer<>(producerProperties())){

consumer.subscribe(Arrays.asList("A","B"));
```

@bbejeck

Life Before Kafka Streams – GroupBy Example

```
while (true) {
 ConsumerRecords<String, String> records =    consumer.poll(Duration.ofSeconds(5));

   for (ConsumerRecord<String, String> record : records) {
       String key = record.key();
       groupByCounts.compute(key, (k, v) -> v == null ? 1 : v + 1);
   }
```

Life Before Kafka Streams – GroupBy Example

```
if(counter++ % sendInterval == 0) {
  for(Entry<String, Integer> groupedEntry:groupByCounts.entrySet()){
    ProducerRecord<String, Integer> producerRecord =
                        new ProducerRecord<>("group-by-counts",
                                             groupedEntry.getKey(),
                                             groupedEntry.getValue());
     producer.send(producerRecord);
  }
  consumer.commitSync();
}
```

Life Before Kafka Streams – GroupBy Example

```
if(counter++ % sendInterval == 0) {
  for(Entry<String, Integer> groupedEntry:groupByCounts.entrySet()){
    ProducerRecord<String, Integer> producerRecord =
                          new ProducerRecord<>("group-by-counts",
                                               groupedEntry.getKey(),
                                               groupedEntry.getValue());
     producer.send(producerRecord);
  }
  consumer.commitSync();
}
```

Sample Streams Application

```
...
stream = streamBuilder.stream(Arrays.asList("A", "B"))

stream.groupByKey()
      .count(Materialized.as("count-store"))
      .toStream()
      .to("output-topic",
          Produced.with(Serdes.String(), Serdes.Long()))
```

Kafka Streams Topology

View the Topology

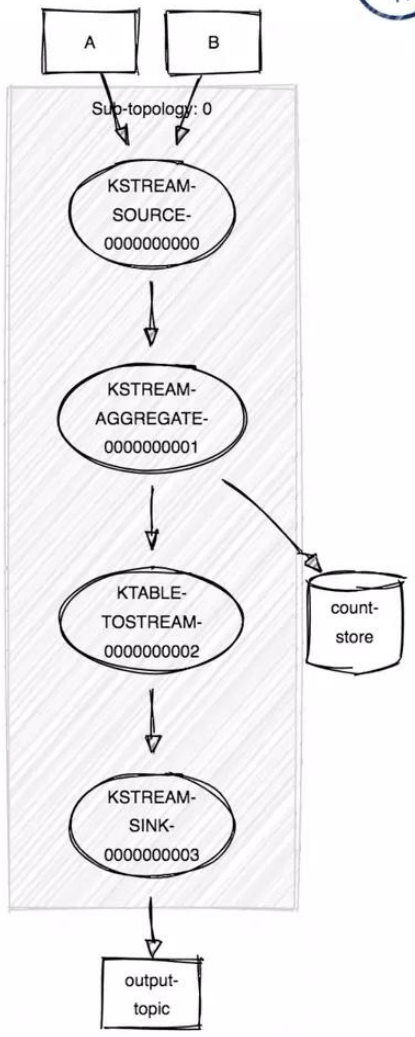
```
Topology topology = streamBuilder.build()

topology.describe().toString()
```

View the Topology

```
Topologies:
   Sub-topology: 0
    Source: KSTREAM-SOURCE-0000000000 (topics: [A, B])
      --> KSTREAM-AGGREGATE-0000000001
    Processor: KSTREAM-AGGREGATE-0000000001 (stores: [count-store])
      --> KTABLE-TOSTREAM-0000000002
      <-- KSTREAM-SOURCE-0000000000
    Processor: KTABLE-TOSTREAM-0000000002 (stores: [])
      --> KSTREAM-SINK-0000000003
      <-- KSTREAM-AGGREGATE-0000000001
    Sink: KSTREAM-SINK-0000000003 (topic: output-topic)
      <-- KTABLE-TOSTREAM-0000000002
```

View Graphic Topology

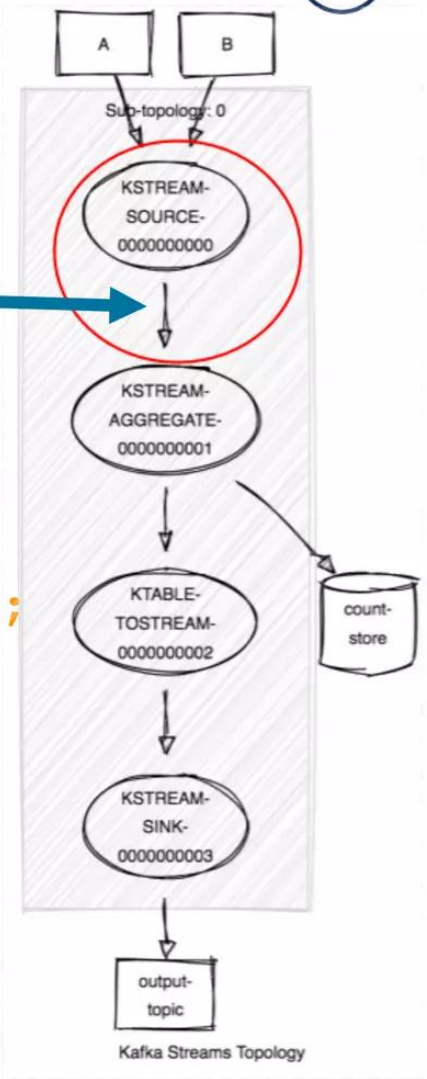
- https://github.com/zz85/kafka-streams-viz



Kafka Streams Topology

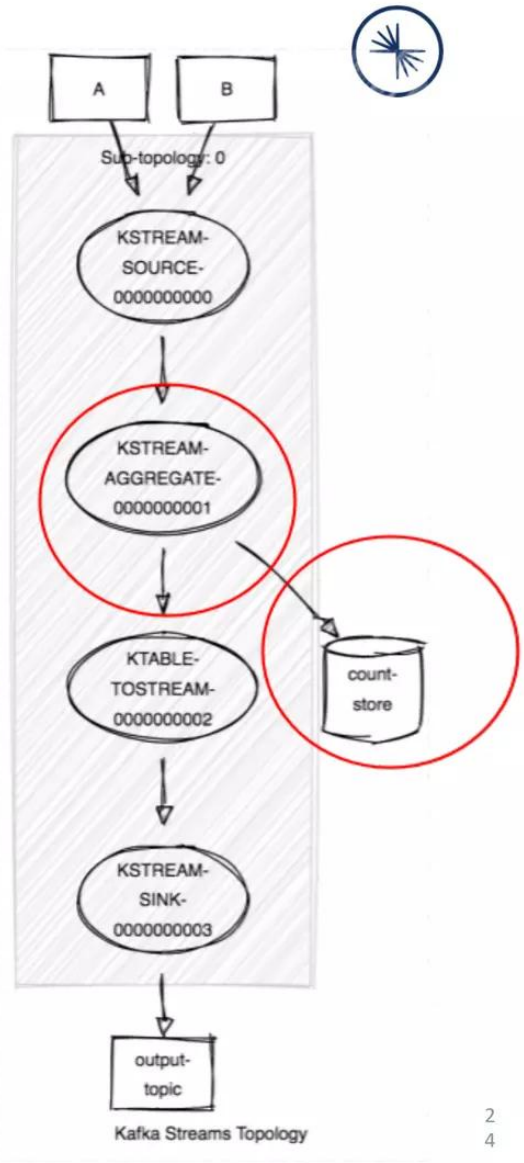
View the Topology

```
stream = streamBuilder.stream(Arrays.asList("A", "B"));


 stream.groupByKey()
       .count(Materialized.as("count-store"))
       .toStream()
       .to("output-topic",
           Produced.with(Serdes.String(), Serdes.Long()));
```



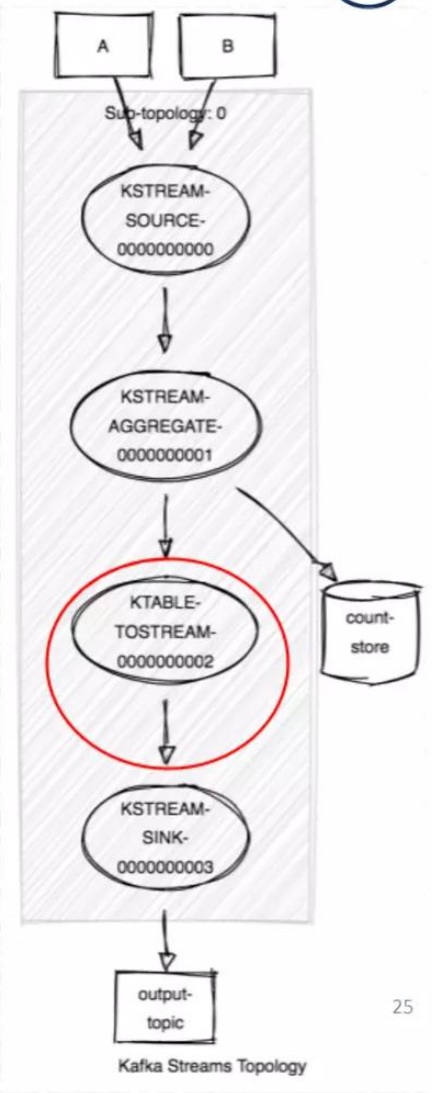
View the Topology

```
stream = builder.stream(Arrays.asList("A", "B"));


stream.groupByKey()
      .count(Materialized.as("count-store"))
      .toStream()
      .to("output-topic",
        Produced.with(Serdes.String(), Serdes.Long()));
```



View the Topology

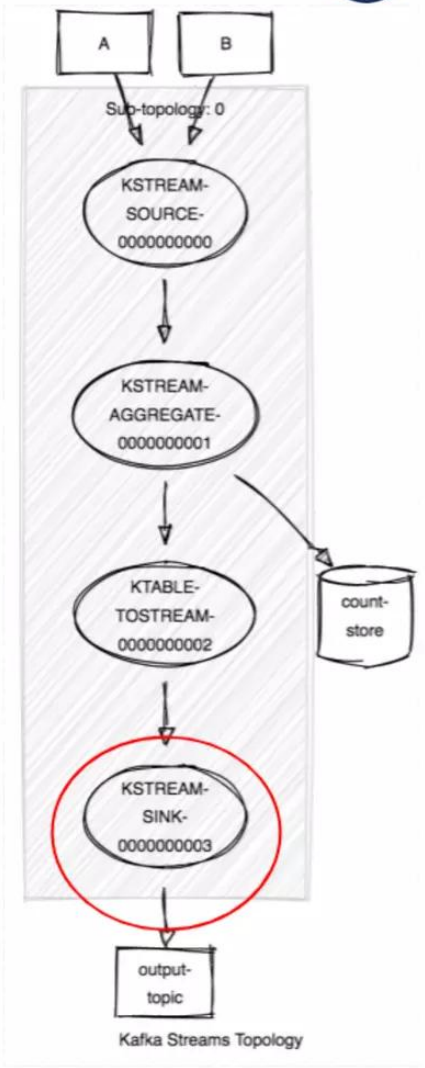
```
stream = builder.stream(Arrays.asList("A", "B"));


stream.groupByKey()
      .count(Materialized.as("count-store"))
      .toStream()
      .to("output-topic",
         Produced.with(Serdes.String(), Serdes.Long()));
```



Kafka Streams Topology

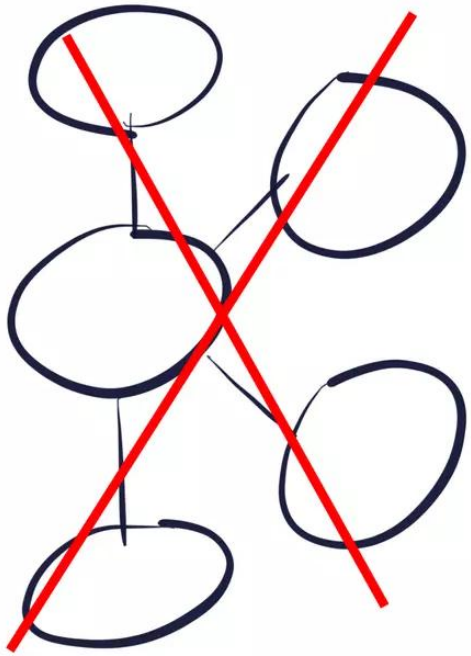
View the Topology

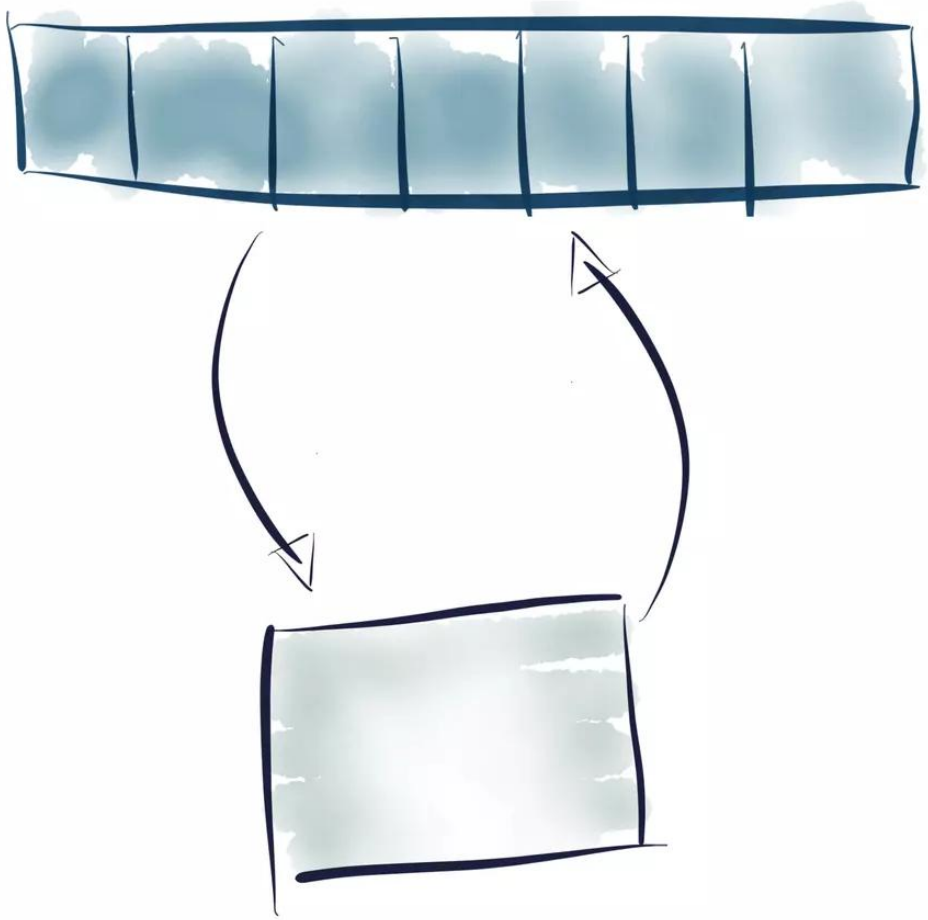
```
stream = builder.stream(Arrays.asList("A", "B"));


 stream.groupByKey()
       .count(Materialized.as("count-store"))
       .toStream()
       .to("output-topic",
         Produced.with(Serdes.String(), Serdes.Long()));
```



Kafka Streams Concepts/Architecture

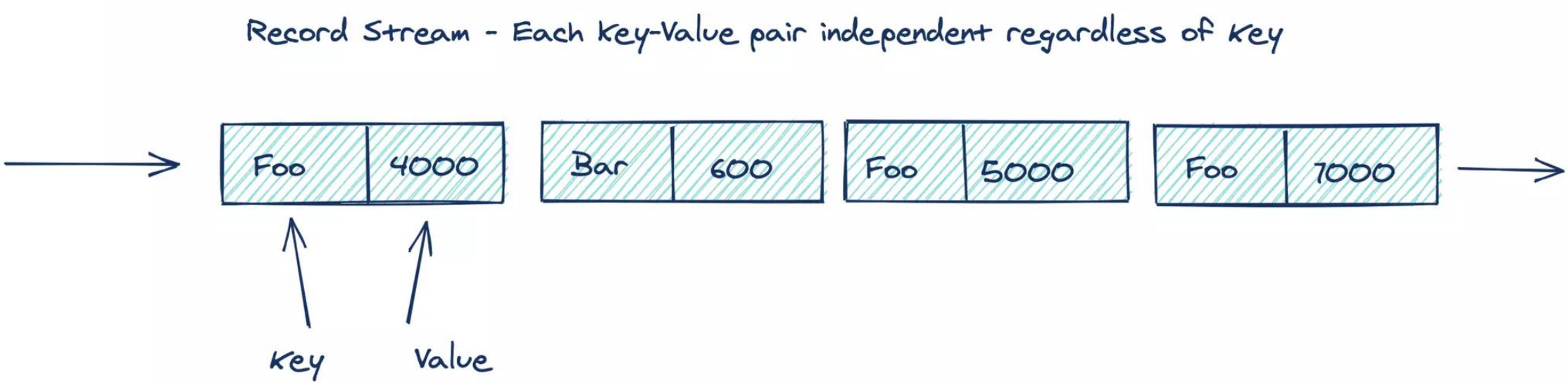
Where does Kafka Streams run?



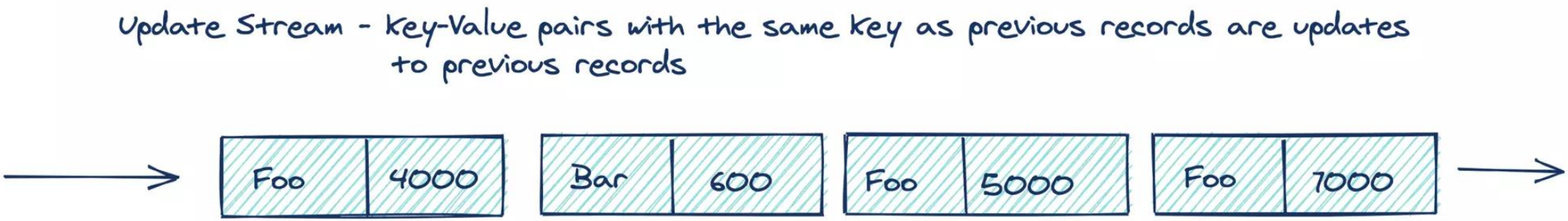


@bbejeck

Record Streams - KStream



Update Stream - KTable

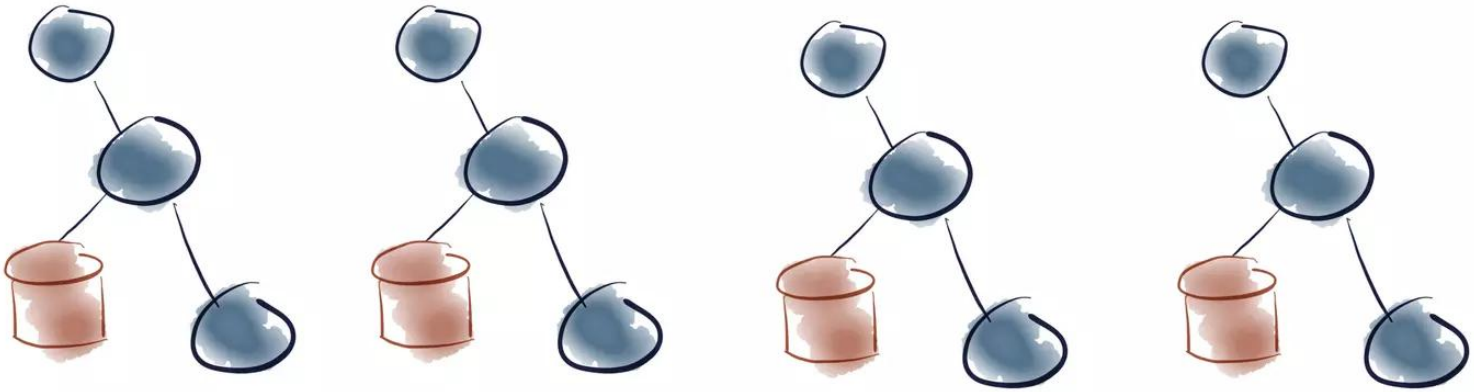


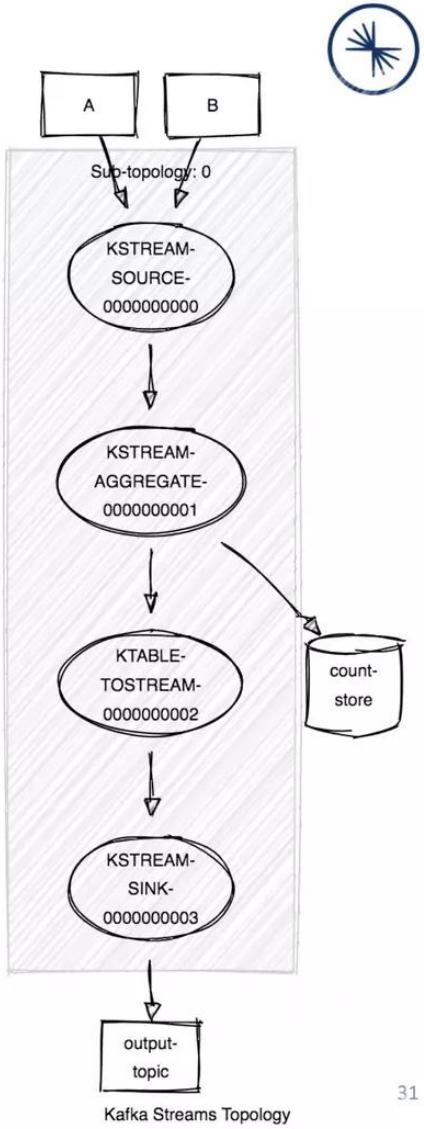
Kafka Streams Tasks

Topic A
A0 A1 A2 A3

Topic B
B0 B1

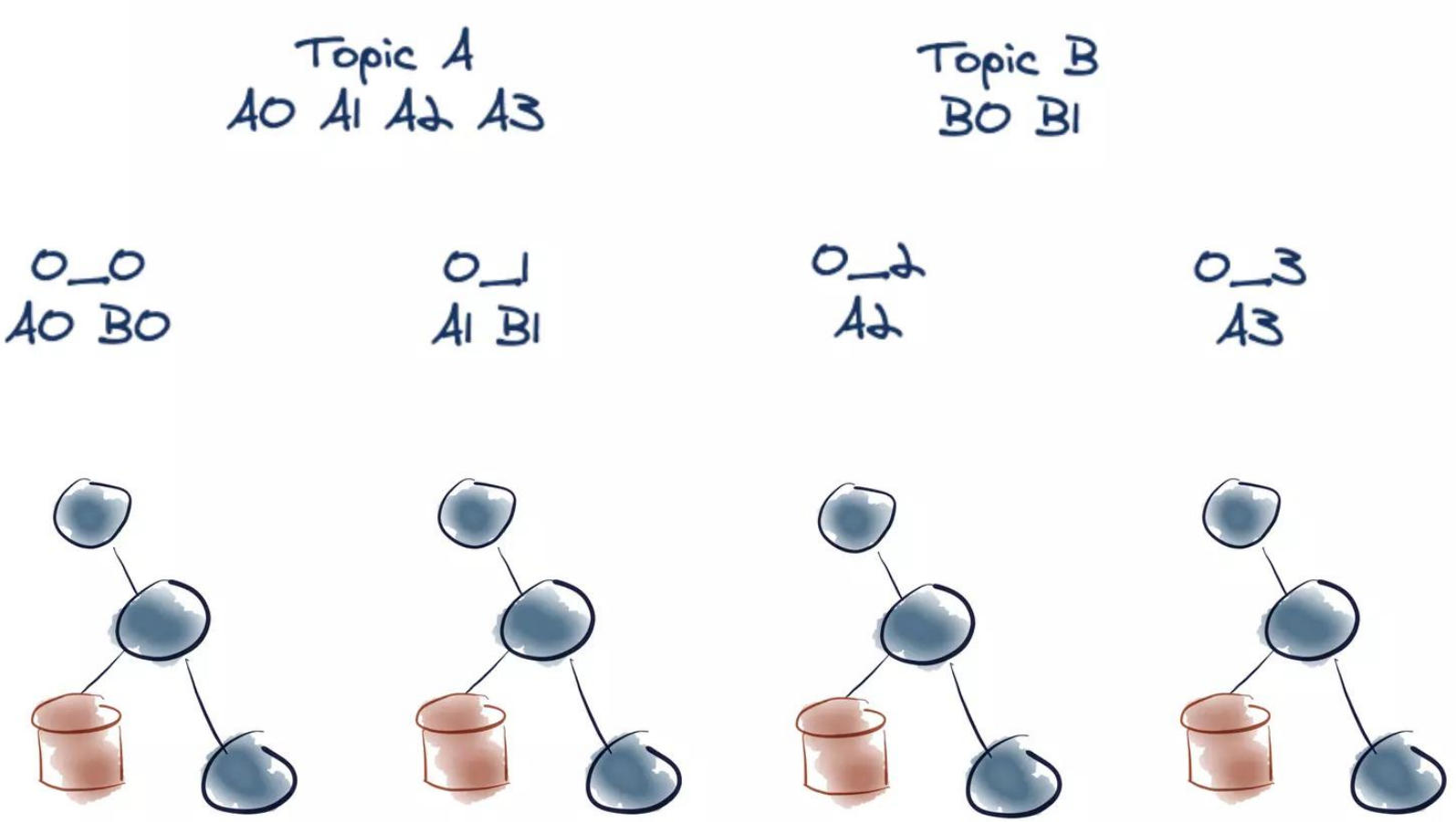
Number tasks = max(4, 2)

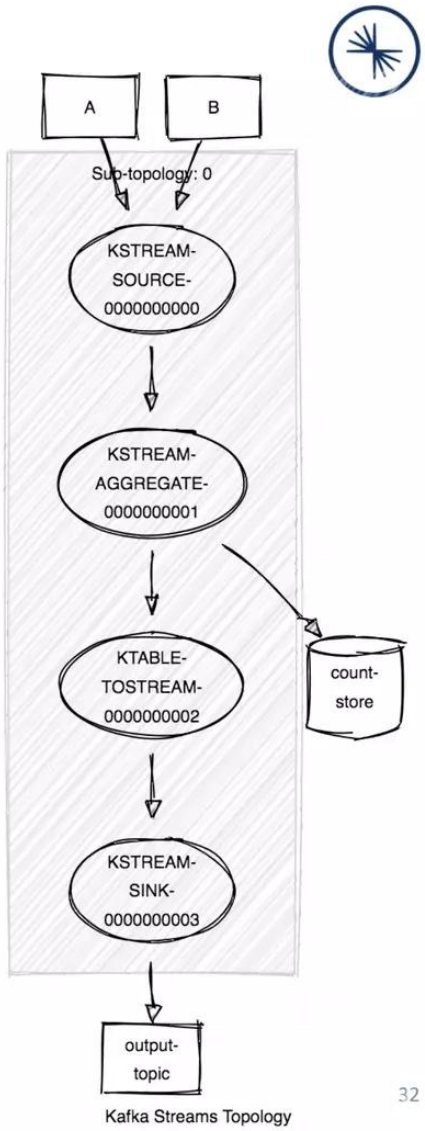




Kafka Streams Topology

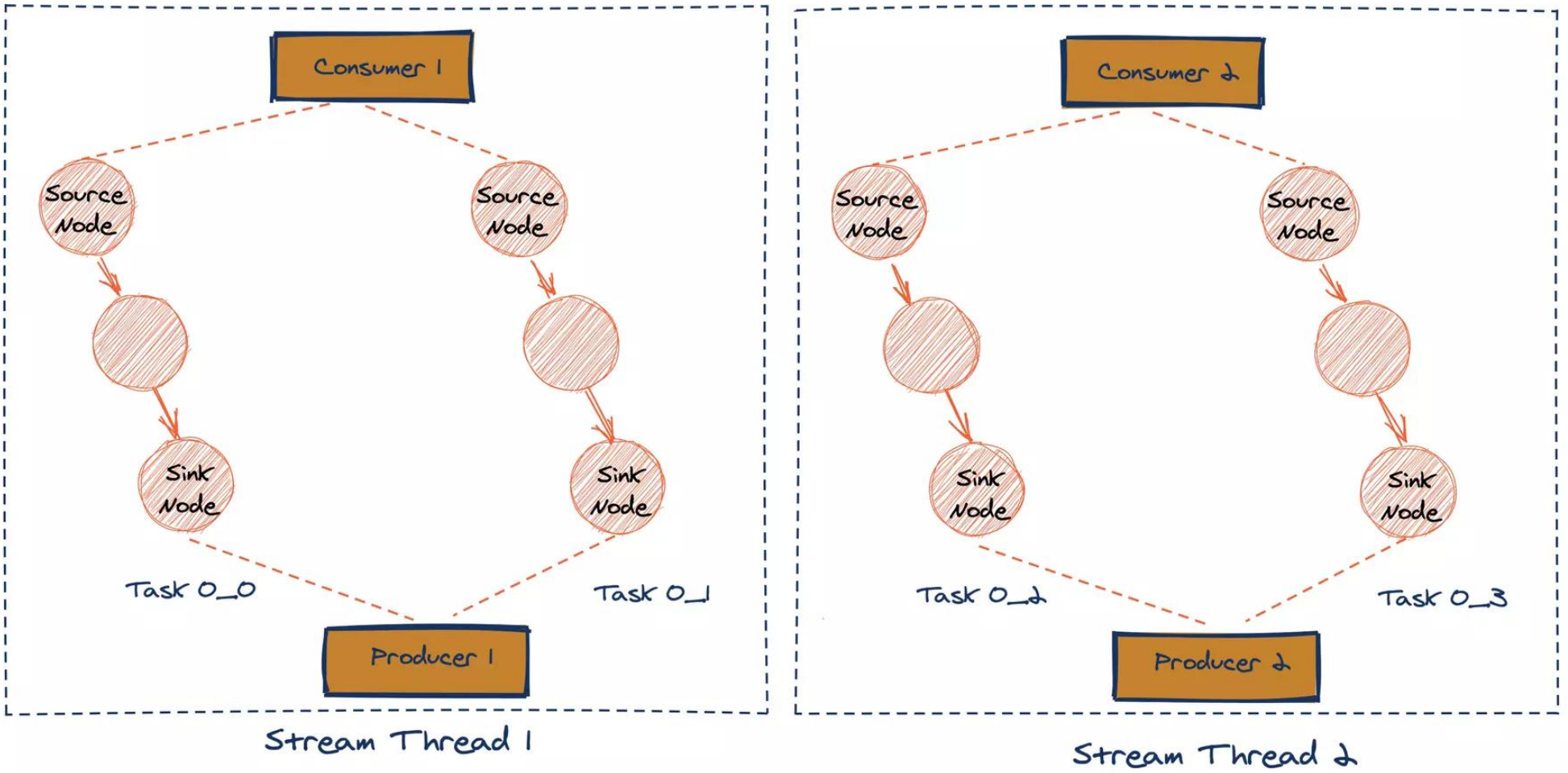
Kafka Streams Tasks





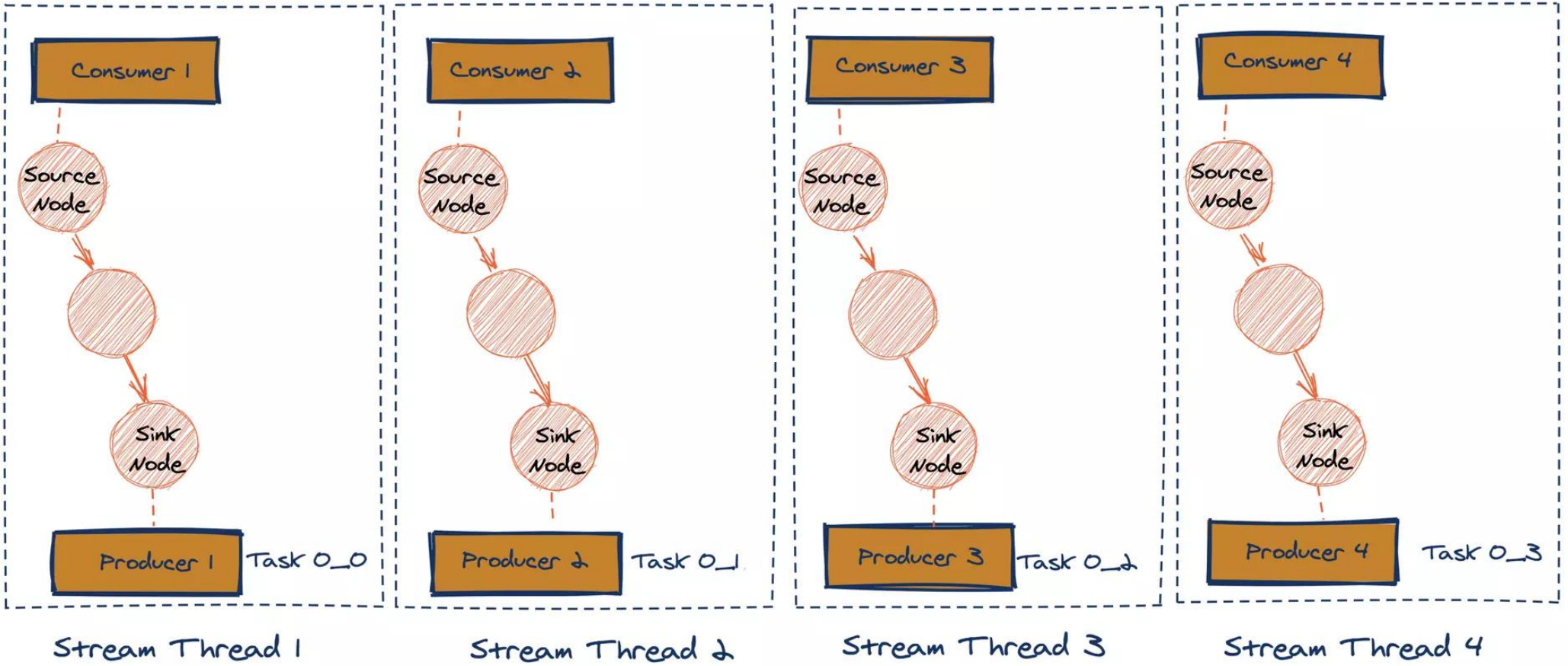
Kafka Streams Topology

Kafka Streams Architecture - Threading



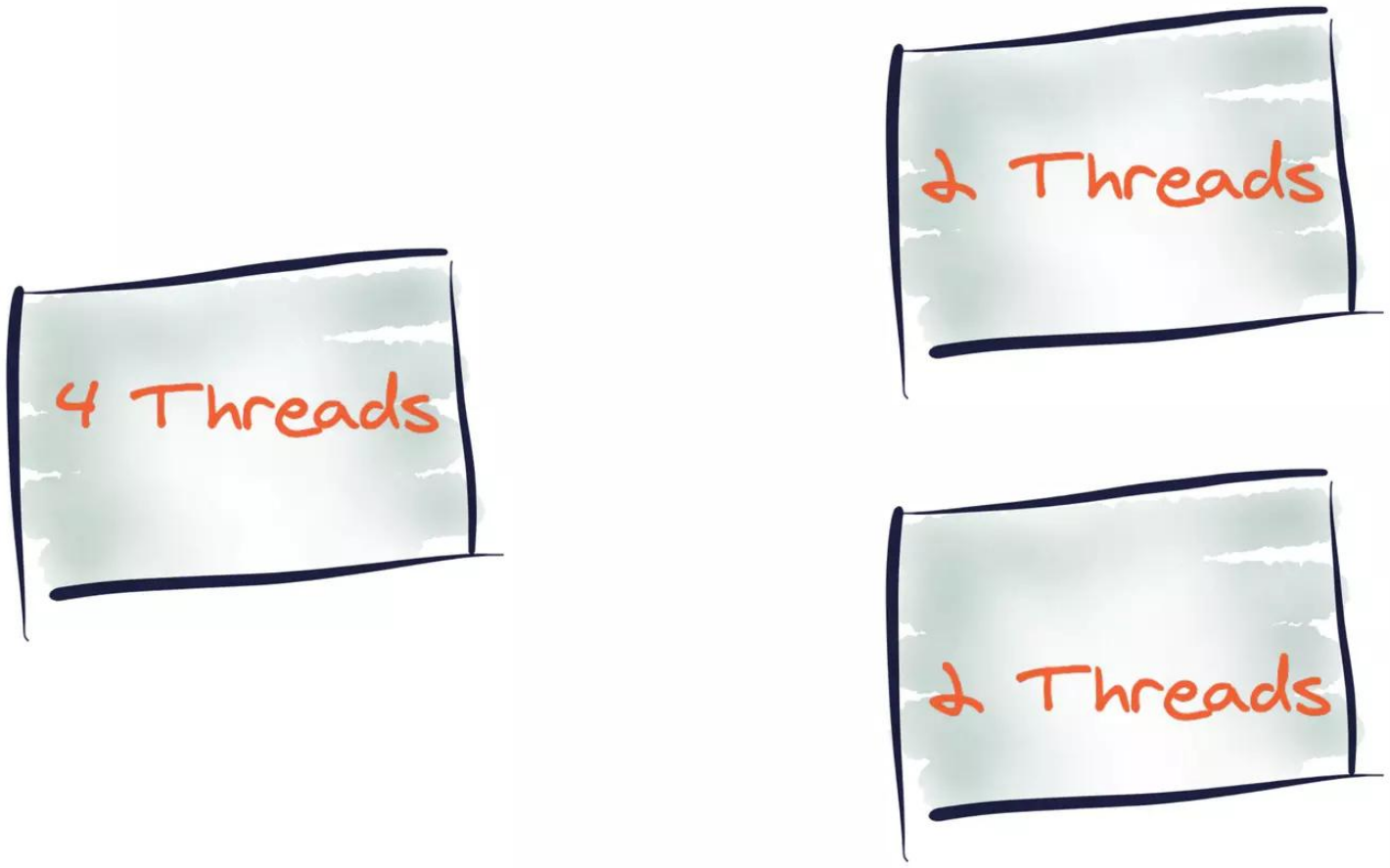
@bbejeck

Kafka Streams Architecture - Threading



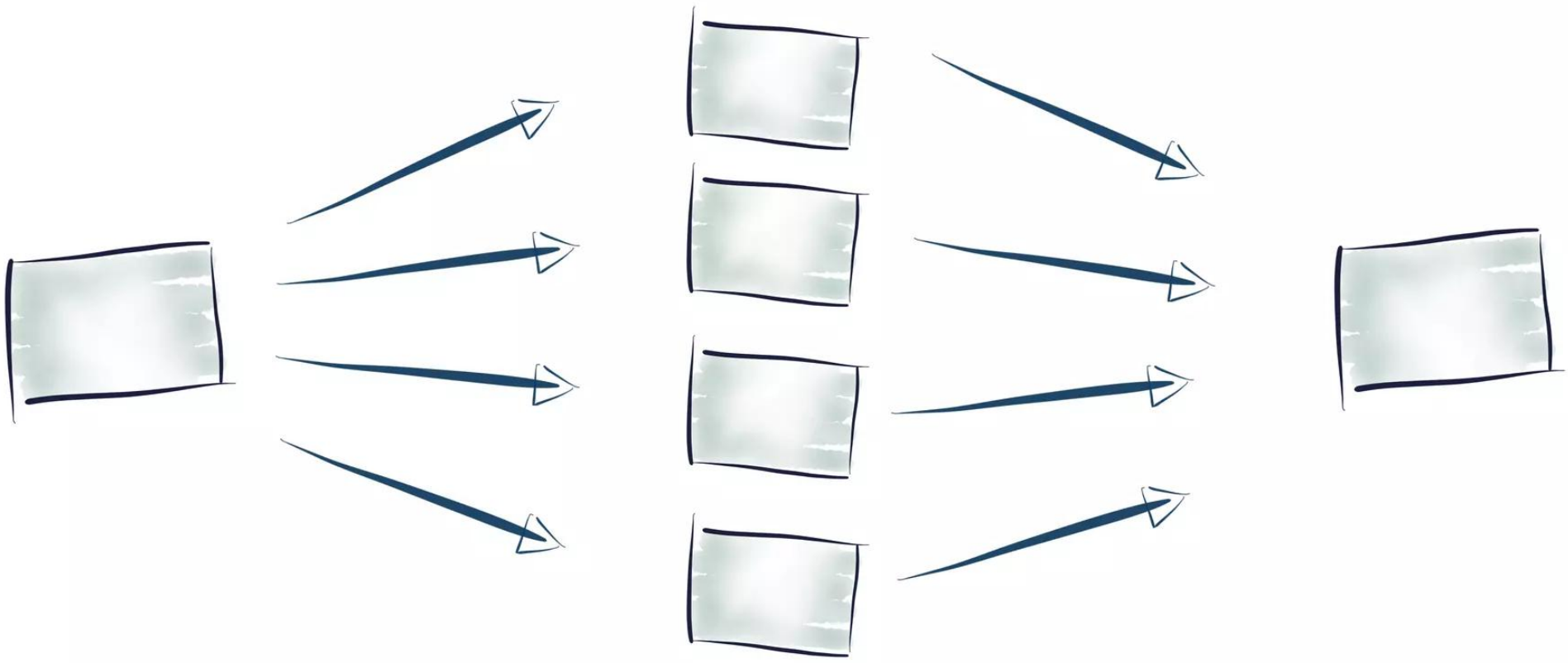
@bbejeck

Kafka Streams – Deployment Consideration



@bbejeck

Kafka Streams Dynamic Expansion/Contraction



@bbejeck

Kafka Steams API/Features

Kafka Streams – DSL API

```
...
stream = streamBuilder.stream(Arrays.asList("A", "B"))

stream.groupByKey()
      .count(Materialized.as("count-store"))
      .toStream()
      .to("output-topic",
          Produced.with(Serdes.String(), Serdes.Long()))
```

Processor API - WordCountProcessorDemo

```
final Topology builder = new Topology();

builder.addSource("Source", "streams-plaintext-input");

builder.addProcessor("Process", new MyProcessorSupplier(), "Source");
builder.addStateStore(Stores.keyValueStoreBuilder(
                                Stores.inMemoryKeyValueStore("Counts"),
                                Serdes.String(),
                                Serdes.Integer()),
                                "Process");

builder.addSink("Sink", "streams-wordcount-processor-output", "Process");
```

Processor API - WordCountProcessorDemo

```
final Topology builder = new Topology();

builder.addSource("Source", "streams-plaintext-input");

builder.addProcessor("Process", new MyProcessorSupplier(), "Source");
builder.addStateStore(Stores.keyValueStoreBuilder(
                                  Stores.inMemoryKeyValueStore("Counts"),
                                  Serdes.String(),
                                  Serdes.Integer()),
                                  "Process");

builder.addSink("Sink", "streams-wordcount-processor-output", "Process");
```

@bbejeck

Processor API - WordCountProcessorDemo

```
final Topology builder = new Topology();

builder.addSource("Source", "streams-plaintext-input");

builder.addProcessor("Process", new MyProcessorSupplier(), "Source");
builder.addStateStore(Stores.keyValueStoreBuilder(
                                    Stores.inMemoryKeyValueStore("Counts"),
                                    Serdes.String(),
                                    Serdes.Integer()),
                                    "Process");

builder.addSink("Sink", "streams-wordcount-processor-output", "Process");
```

Processor API - WordCountProcessorDemo

```
final Topology builder = new Topology();

builder.addSource("Source", "streams-plaintext-input");

builder.addProcessor("Process", new MyProcessorSupplier(), "Source");
builder.addStateStore(Stores.keyValueStoreBuilder(
                                Stores.inMemoryKeyValueStore("Counts"),
                                Serdes.String(),
                                Serdes.Integer()),
                                "Process");

builder.addSink("Sink", "streams-wordcount-processor-output", "Process");
```

Kafka Streams – Combining DSL and Processor API

```
stream.groupByKey()
      .count(Materialized.as("count-store"))
      .toStream()
      .transform(MyTransformSupplier())
      .to("output-topic",
         Produced.with(Serdes.String(), Serdes.Long()))
```

transformValues, flatTransform, process (terminal node)

Kafka Streams – Stateless Operations

```
KStream.filter(..)
KStream.map(..)
KStream.mapValues(..)
KStream.transform(..)
..
KTable.filter(..)
KTable.groupBy(..)
KTable.mapValues(..)
```

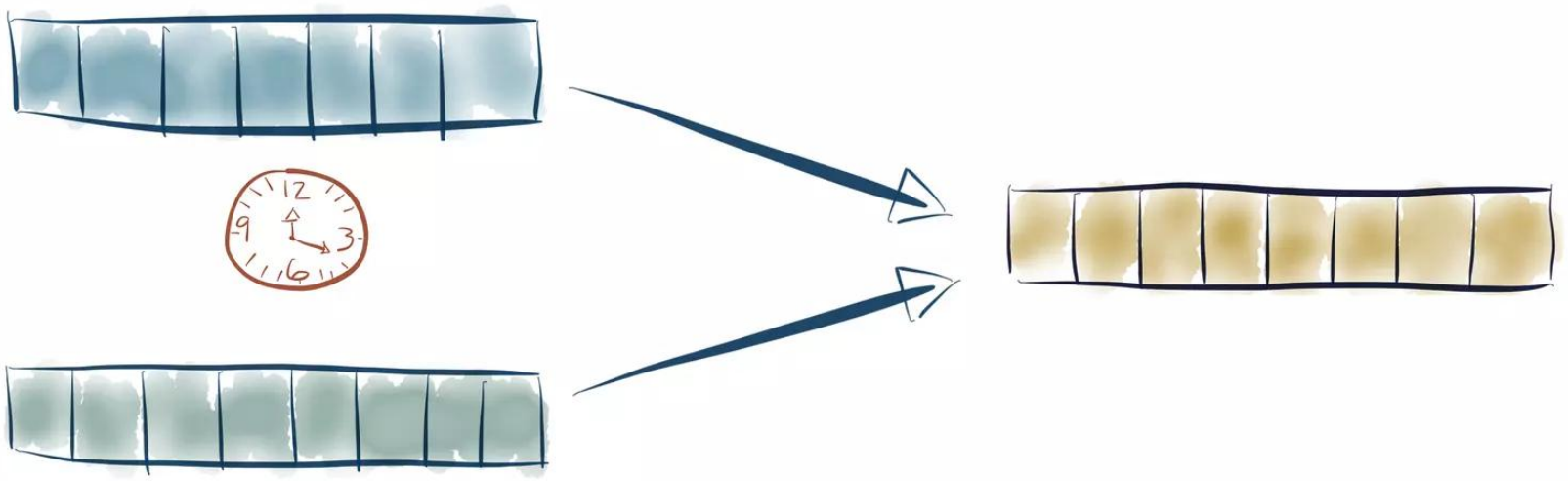
https://docs.confluent.io/current/streams/developer-guide/dsl-api.html#stateless-transformations

Stateful Operations

```
KStream.groupByKey().count()
KStream.groupByKey().reduce(..)
KStream.groupByKey().aggregate(..)
..
KTable.groupBy(..).count(..)
KTable.groupBy(..).reduce(..)
KTable.groupBy(..).aggregate(..)
```

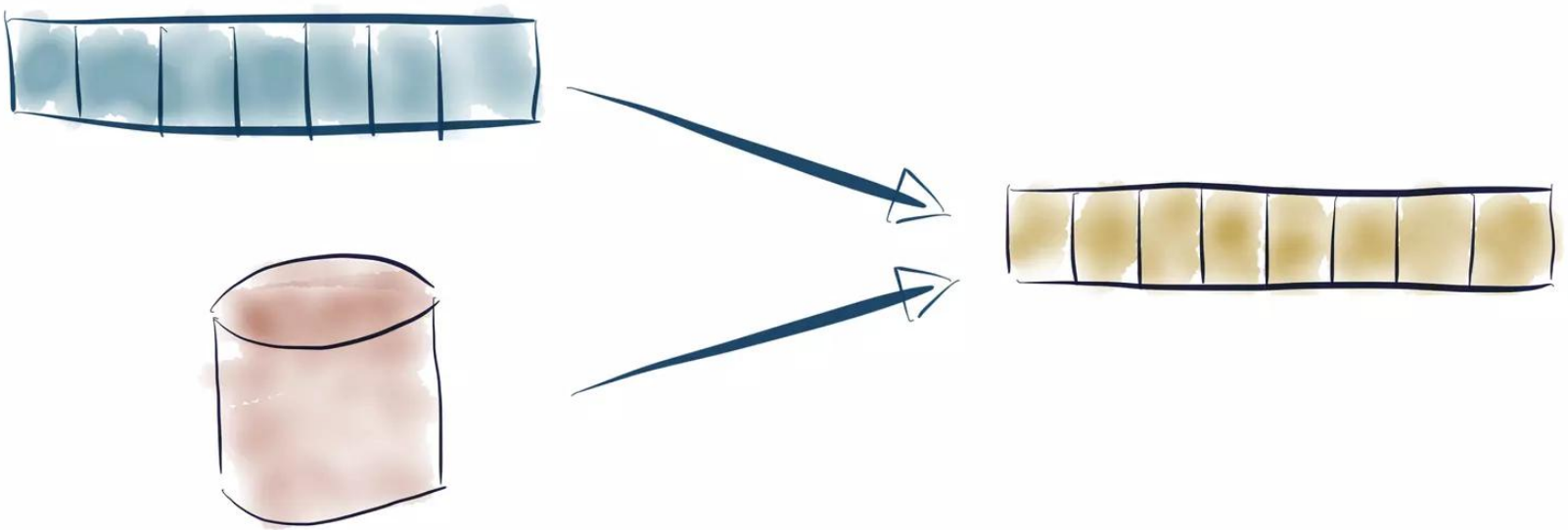
https://docs.confluent.io/current/streams/developer-guide/dsl-api.html#stateful-transformations

Stateful Operations – Stream-Stream Join



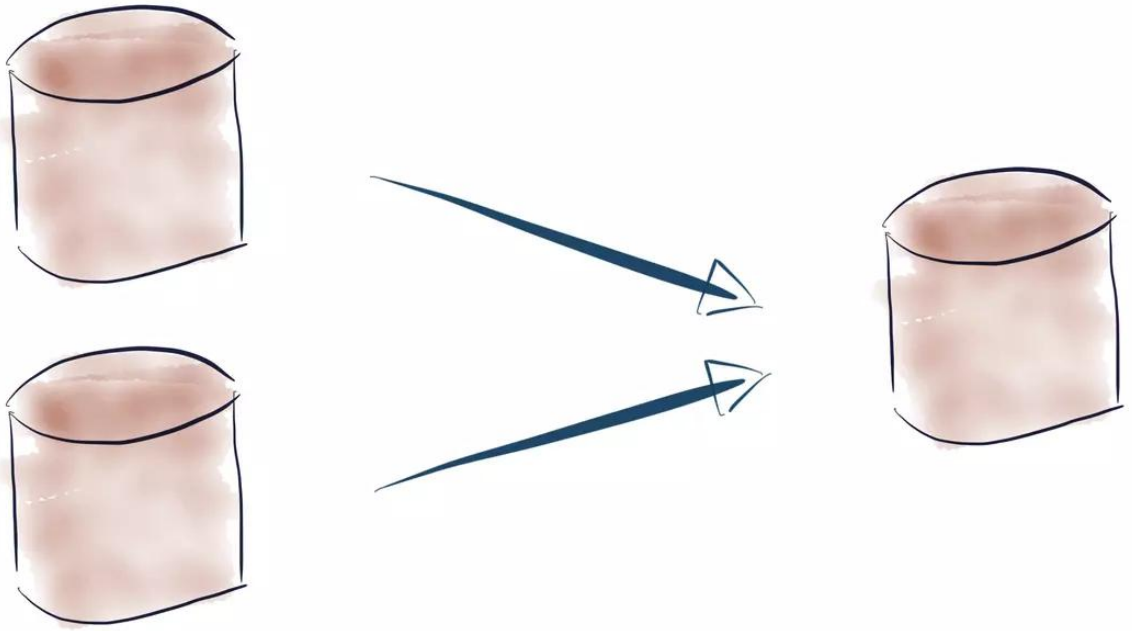
@bbejeck

Stateful Operations – Stream-Table Join



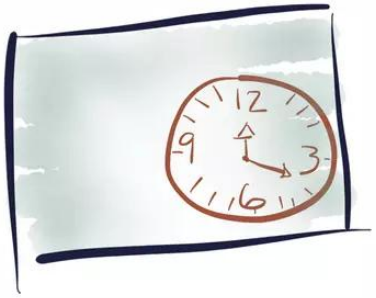
@bbejeck

Stateful Operations – Table -Table Join

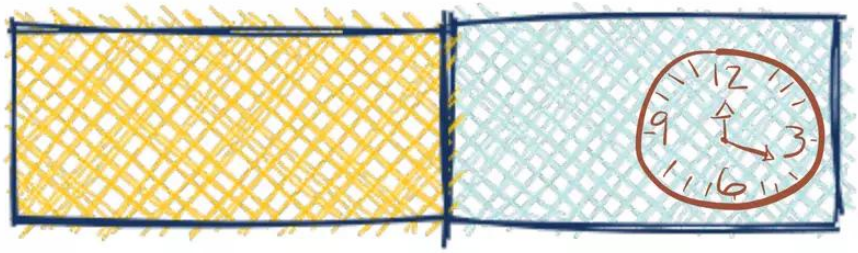


@bbejeck

Time in Kafka Streams



Wall-clock time is the time the application is processing the record



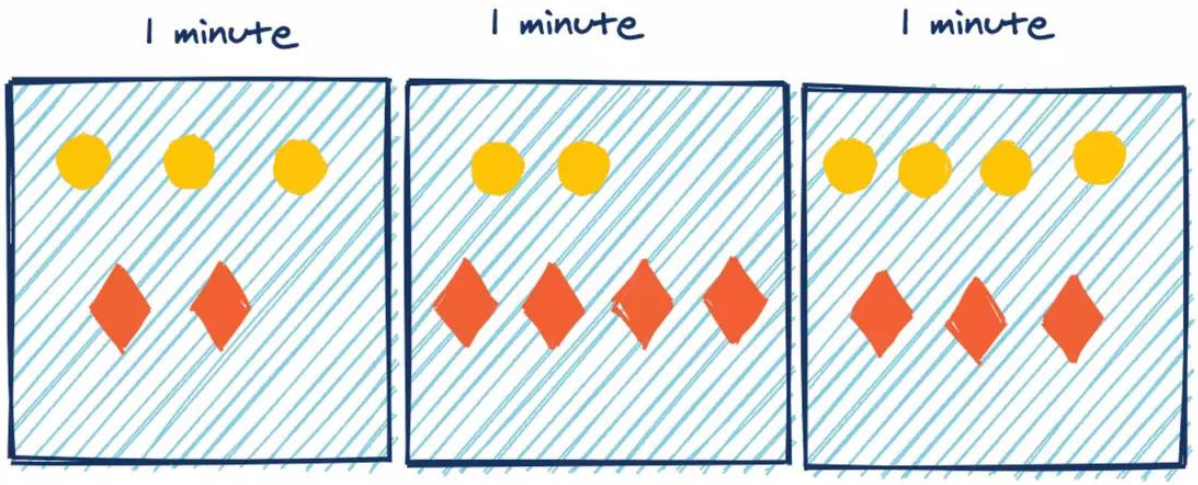
Time used by Kafka Streams is the timestamp of the record – pulled out by the `TimestampExtractor`.

@bbejeck

Windowing – Tumbling Windows

```
Duration windowSize = Duration.ofMinutes(1);
stream.groupByKey().count().windowedBy(TimeWindows.of(windowSize));
```

Tumbling windows - Non-overlapping events bounded by start and end time

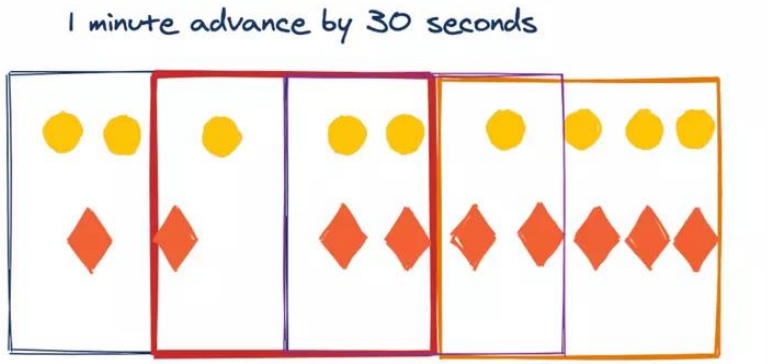


@bbejeck

Windowing – Hopping Windows

```
Duration windowSize = Duration.ofMinutes(1);
Duration advance = Duration.ofSeconds(30);
stream.groupByKey().count().windowedBy(TimeWindows.of(windowSize)).advanceBy(advance));
```

Hopping windows - overlapping fixed sized window bounded by start end time with an incremental update

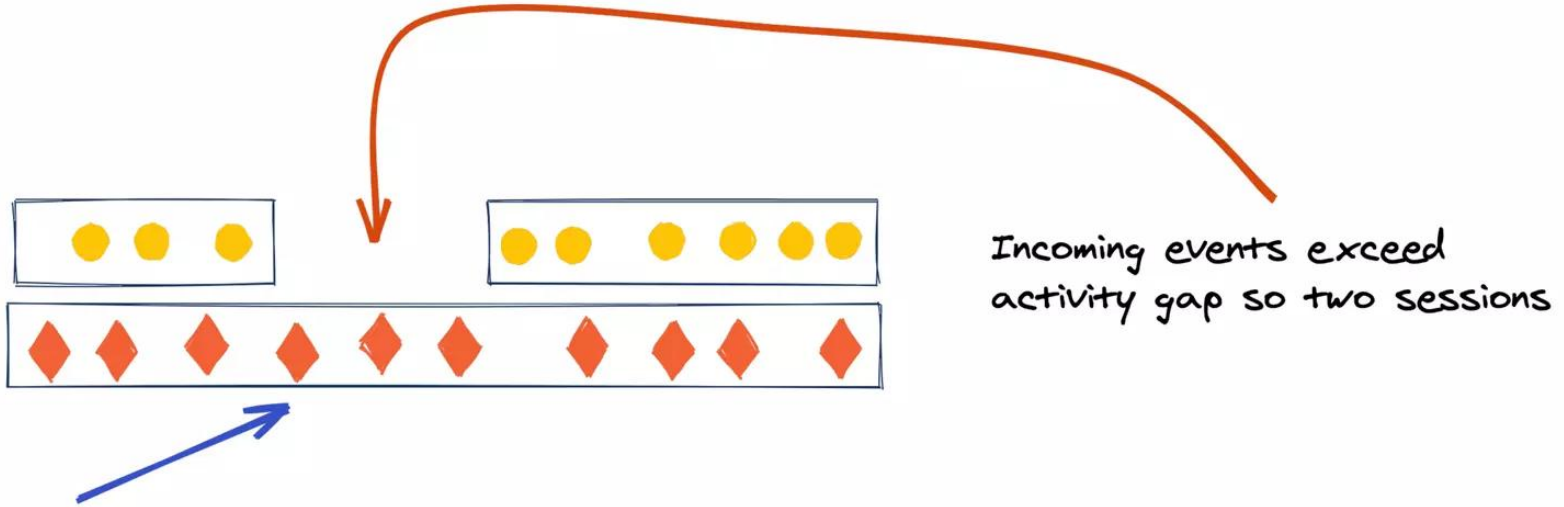


@bbejeck

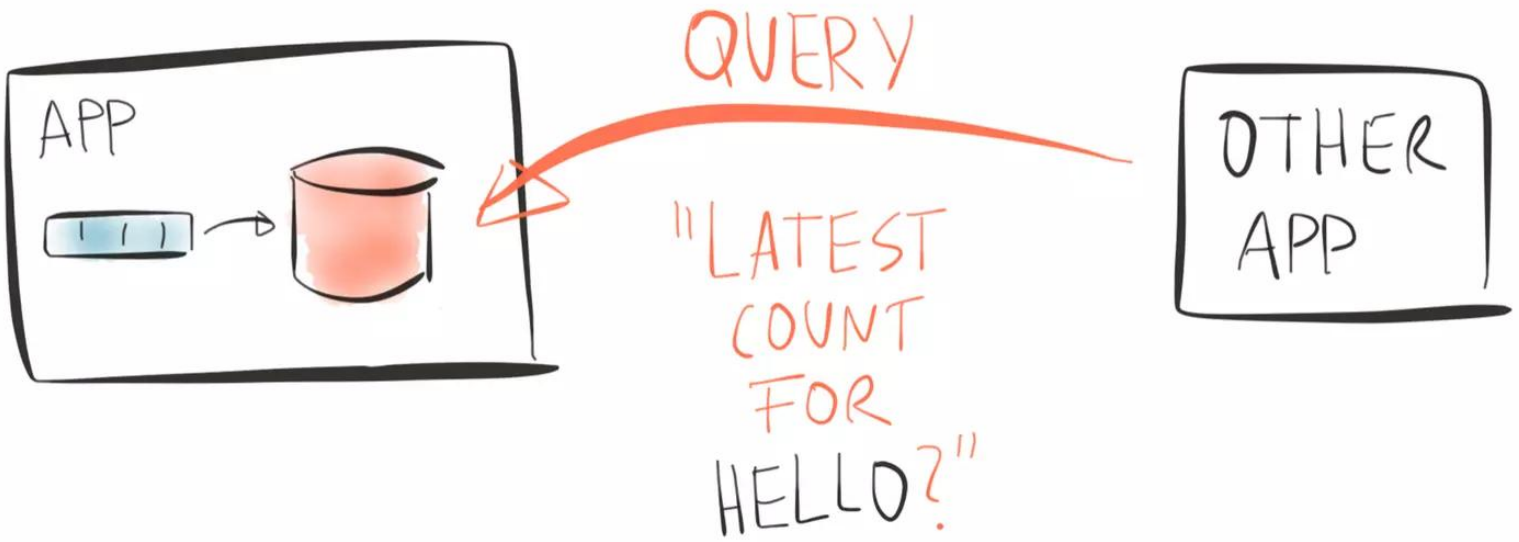
Windowing – Session Windows

```
Duration inactivityGap = Duration.ofMinutes(3);

stream.groupByKey().count().windowedBy(SessionWindows.of(inactivityGap));
```

Session Windows - Dynamic sized windows based on behavior inactivity defines a session



Interactive Queries



Resources

- **Kafka Tutorials - https://kafka-tutorials.confluent.io/**
- **Confluent Developer - developer.confluent.io**
- **Examples - https://github.com/confluentinc/kafka-streams-examples/**
- **Documentation - https://docs.confluent.io/current/streams**

Thank you!

@bbejeck

bill@confluent.io

 cnfl.io/meetups

 cnfl.io/blog

 cnfl.io/slack

Learn Kafka!

Confluent Developer

developer.confluent.io