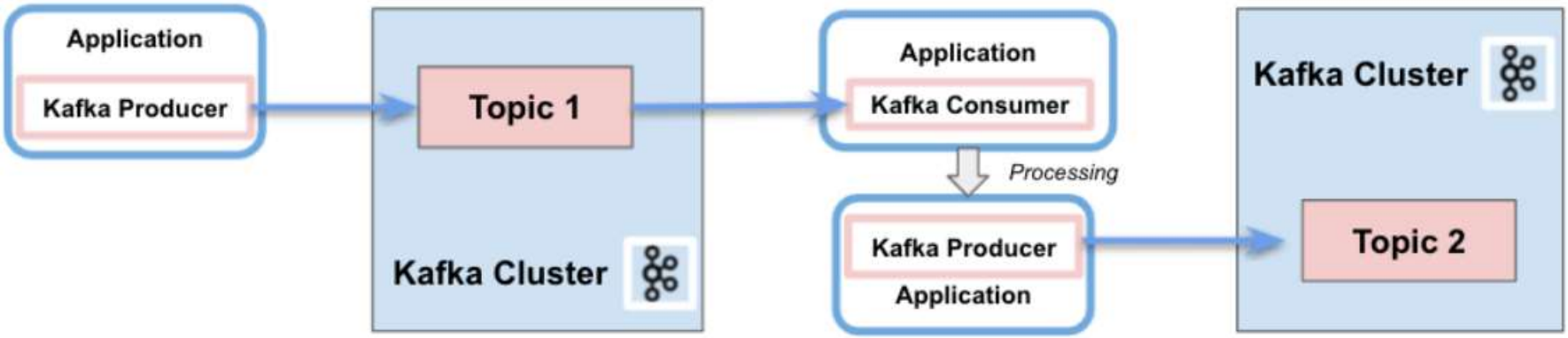
Kafka Streams

Prof. Carlo Ferrari

Michele Stecca, Ph.D.

Developing Kafka Applications

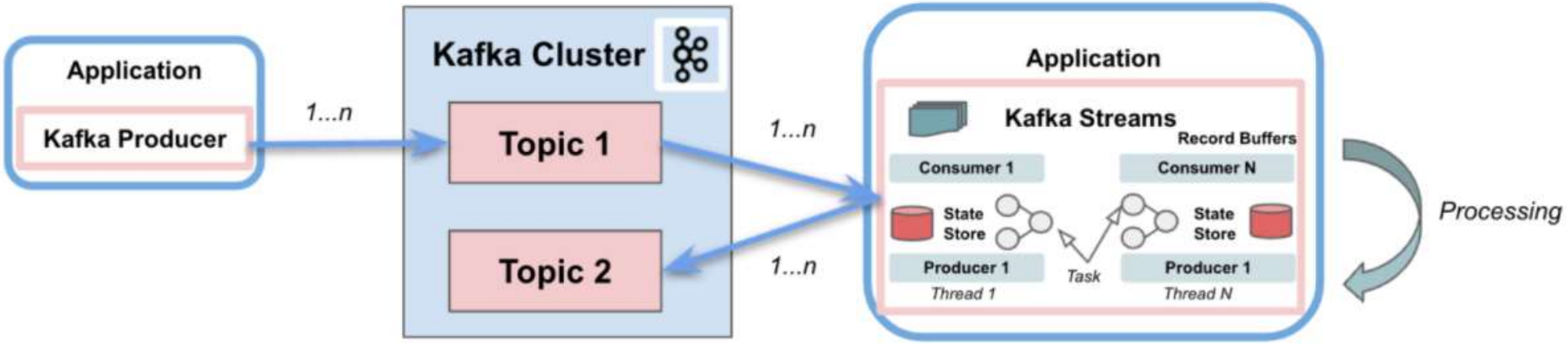
- We can use the JAVA Consumer client
- Repetitive requirements and patterns appear



- Writing an application requires a lot of code
- Threading?
- State management?

Developing Kafka Applications

- Kafka streams



- Write an application requires few lines of code
- Support stateless and stateful operations
- Threading and parallelism

Kafka streams

- Kafka Streams is a **client library** for processing and analyzing data stored in Kafka. The Kafka Stream API interacts with a Kafka cluster but the application does not run directly on Kafka brokers

- Supports fault-tolerant

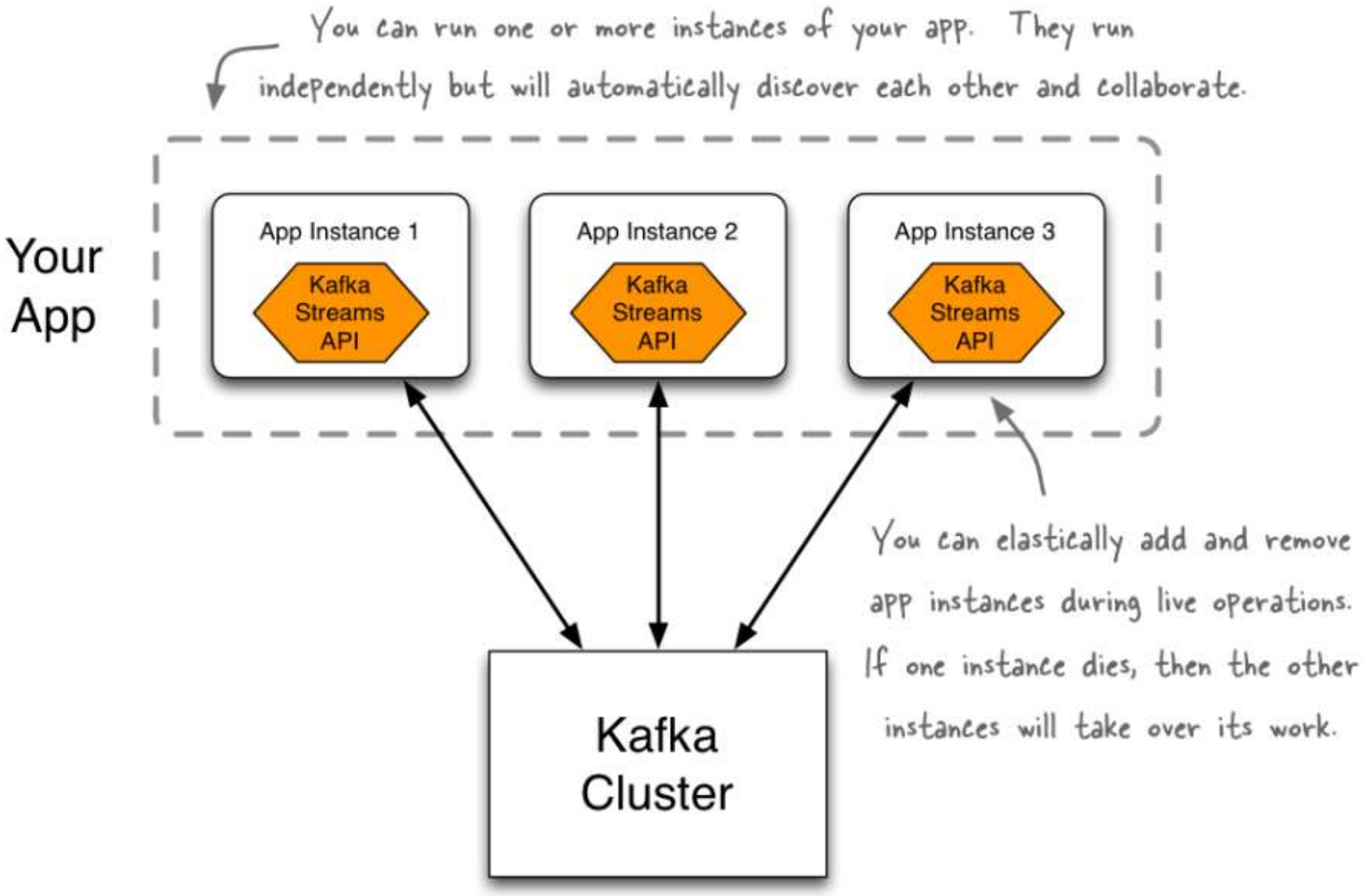
- Offers necessary stream processing primitives:
  - high-level Streams DSL (Domain Specific Language)
  - low-level Processor API

- Transforms and enriches data

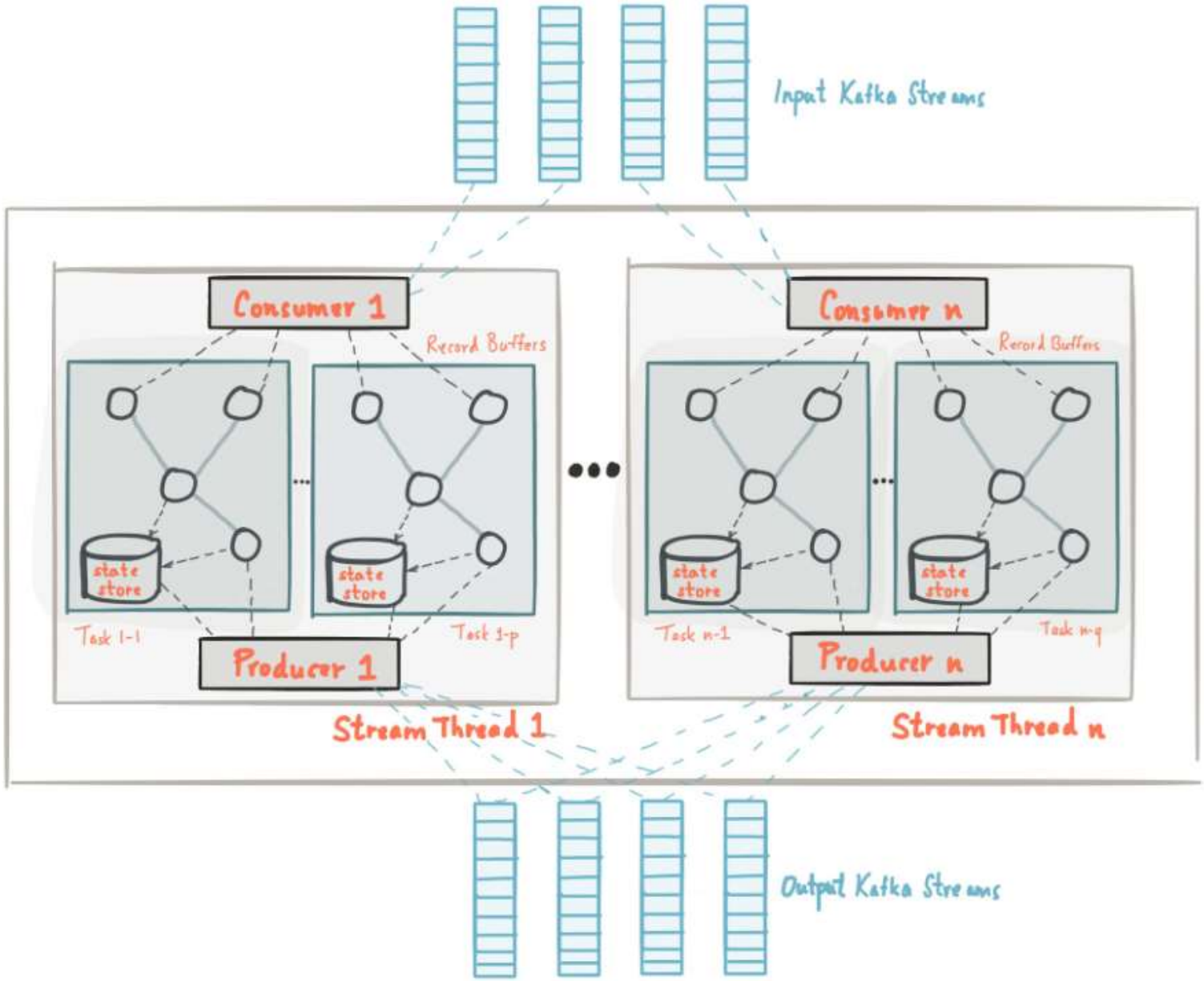
- Supports **per-record stream processing** with millisecond latency (no micro-batching)

- Supports stateless processing, stateful processing, windowing operations

High Level Architecture



Kafka Streams Architecture



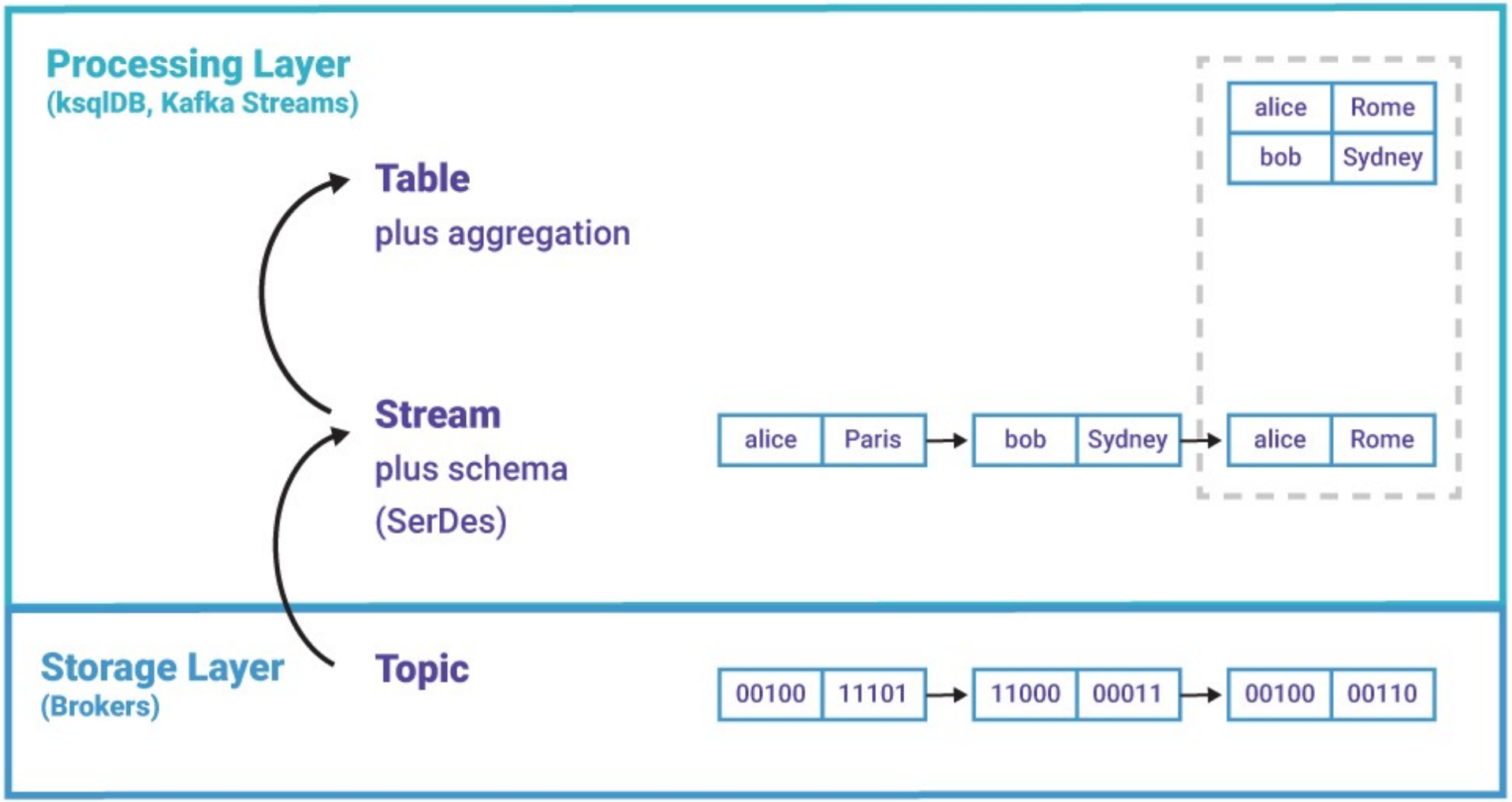
KStreams and KTables

A KStream is an abstraction of a record stream
- Each record represents a self-contained piece of data in the unbounded data set

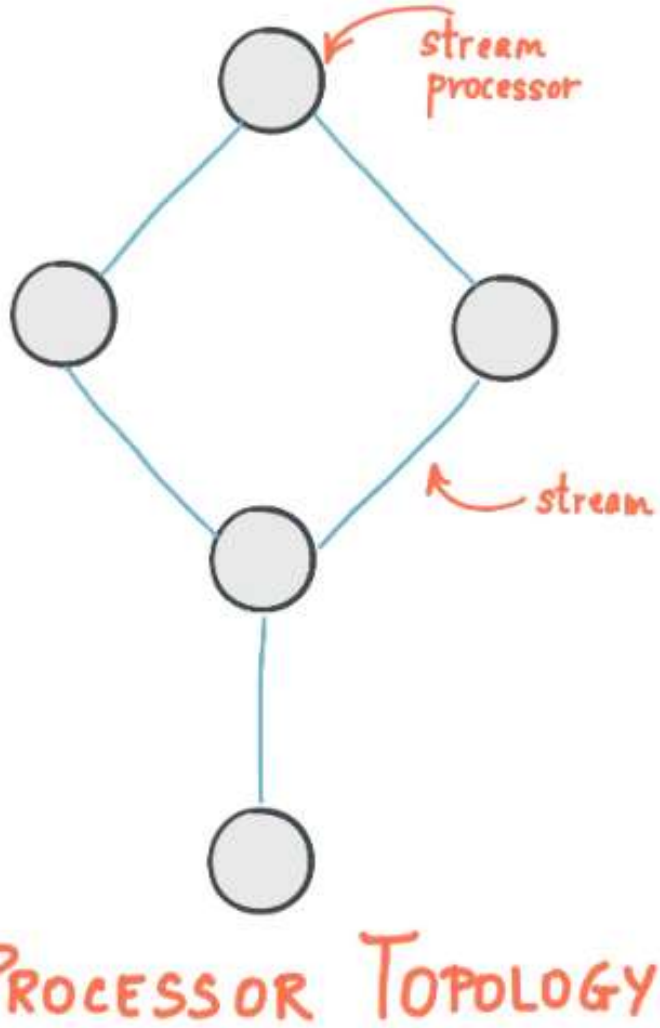
- **A KTable is an abstraction of a changelog stream**
- Each record represents an update

Example	When you need…	Read the Topic into	Data interpreted as	Messages interpreted as
All the places Alice has ever been	All the values of a key	`KStream`	***record*** stream	INSERT (append)
Where Alice is right now	Latest value of a key	`KTable`	***changelog*** stream	UPSERT (overwrite existing)

KStreams and KTables



Process Topology



Process Topology

- A processor topology is a **graph** of stream processors (nodes) that are connected by streams (edges).

- **Stream**: unbounded, continuously updating data set. A stream is an ordered and fault-tolerant sequence of immutable key-value pairs (data records).

- Source Processor produces an input stream to its topology from one Kafka topic by consuming records from these topics and forwarding them to its down-stream processors.

- Sink Processor sends any received records from its up-stream processors **to a Kafka topic**.

Processing data in Kafka Streams

- **Examples of stateless transformation operations:**
  - `filter`
    - Creates a new `KStream` containing only records from the previous `KStream` which meet some specified criteria
  - `map`
    - Creates a new `KStream` by transforming each element in the current stream into a different element in the new stream
  - `mapValues`
    - Creates a new `KStream` by transforming the value of each element in the current stream into a different element in the new stream

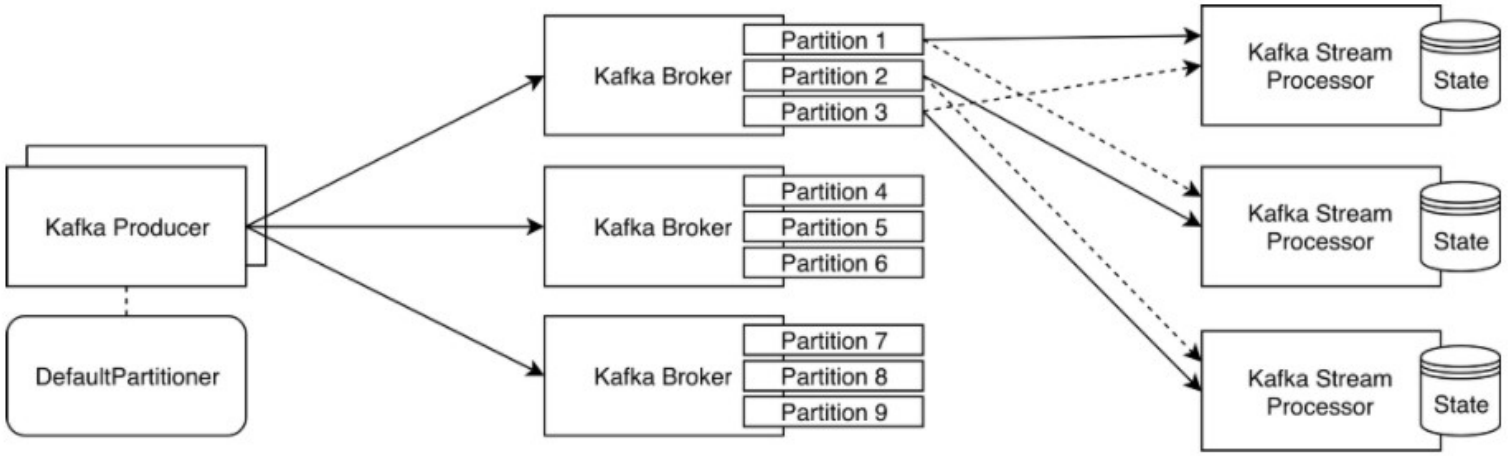
Processing data in Kafka Streams

- **Examples of stateless transformation operations (cont'd):**

- `flatMap`
  - Creates a new `KStream` by transforming each element in the current stream into zero or more different elements in the new stream
- `flatMapValues`
  - Creates a new `KStream` by transforming the value of each element in the current stream into zero or more different elements in the new stream

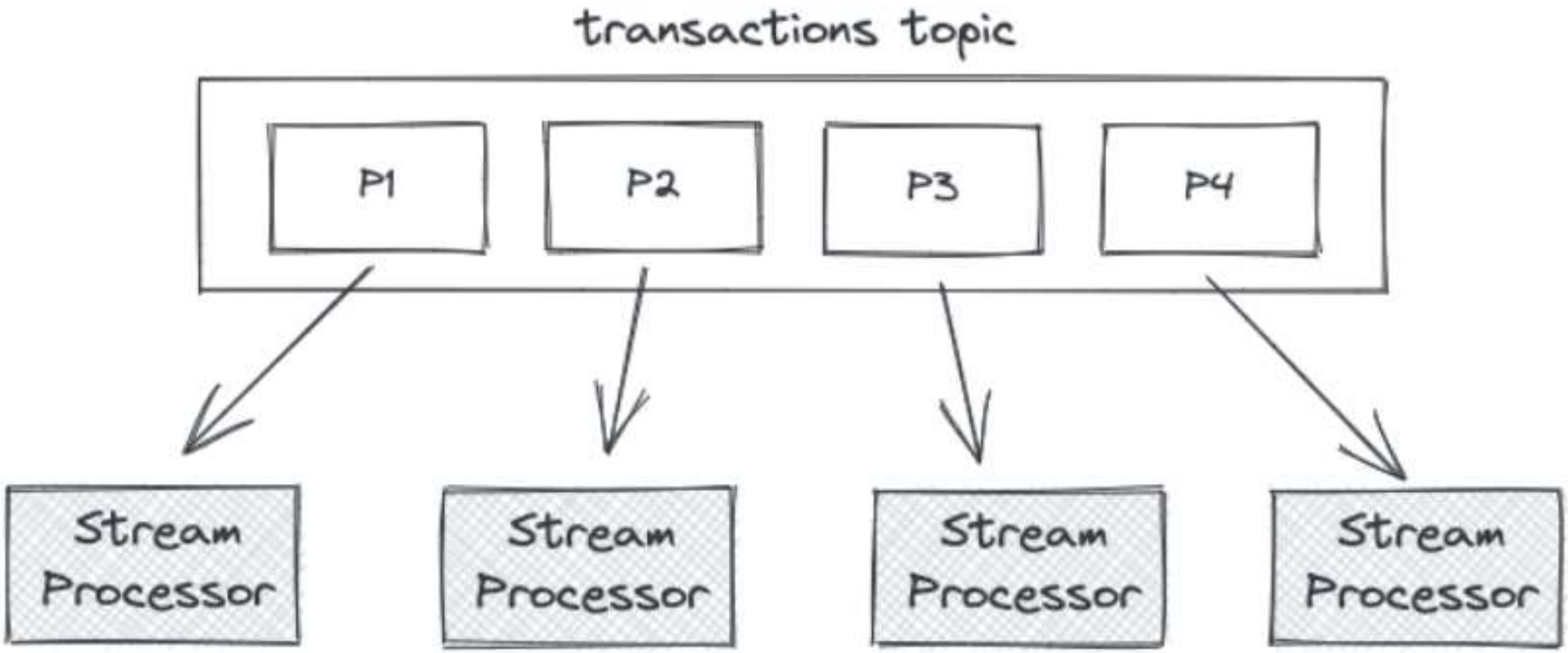
Processing data in Kafka Streams

- **Examples of stateful transformation operations:**
  - `countByKey`
    - Counts the number of instances of each key in the stream; results in a new, ever-updating `KTable`
  - `reduceByKey`
    - Combines values of the stream using a supplied Reducer into a new, ever-updating `KTable`

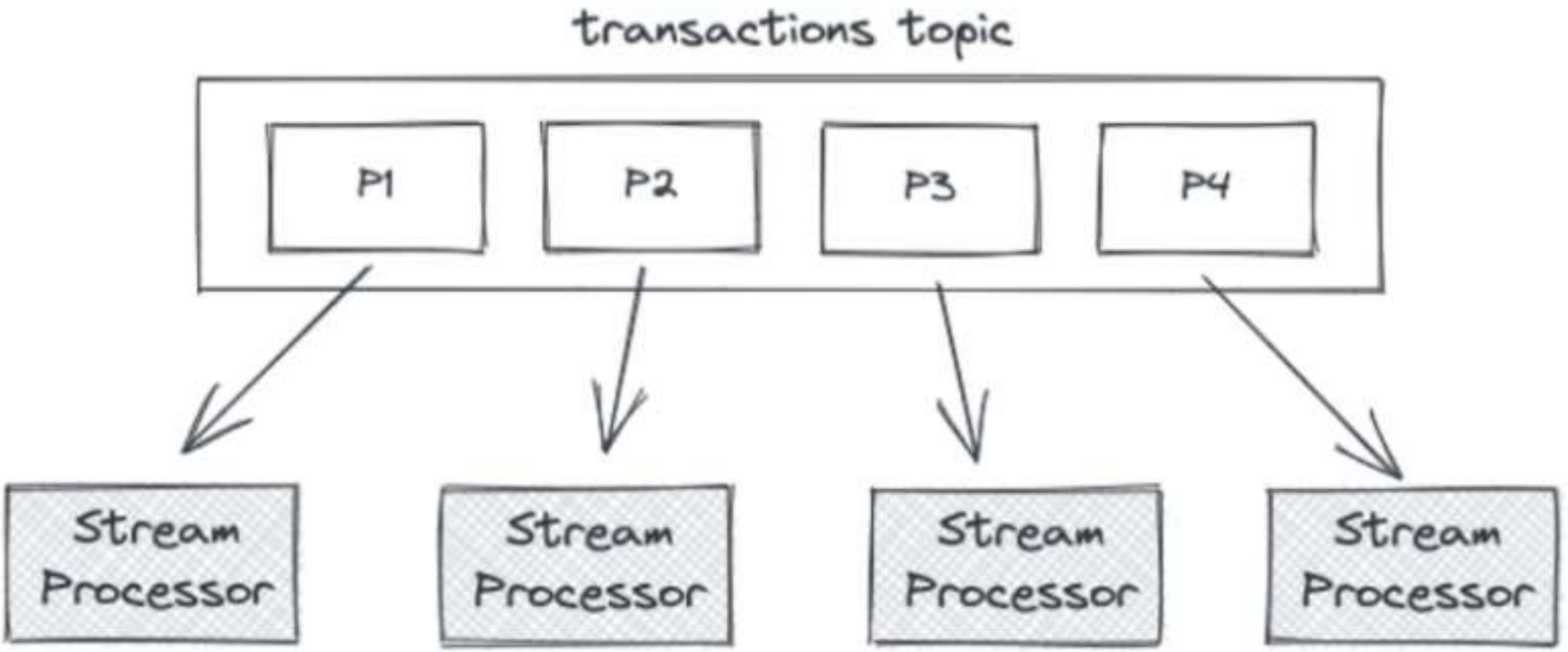


Complete guide here:
https://docs.confluent.io/platform/current/streams/developer-guide/dsl-api.html

Kafka Streams & Fault Tolerance: an example



Kafka Streams & Fault Tolerance: an example



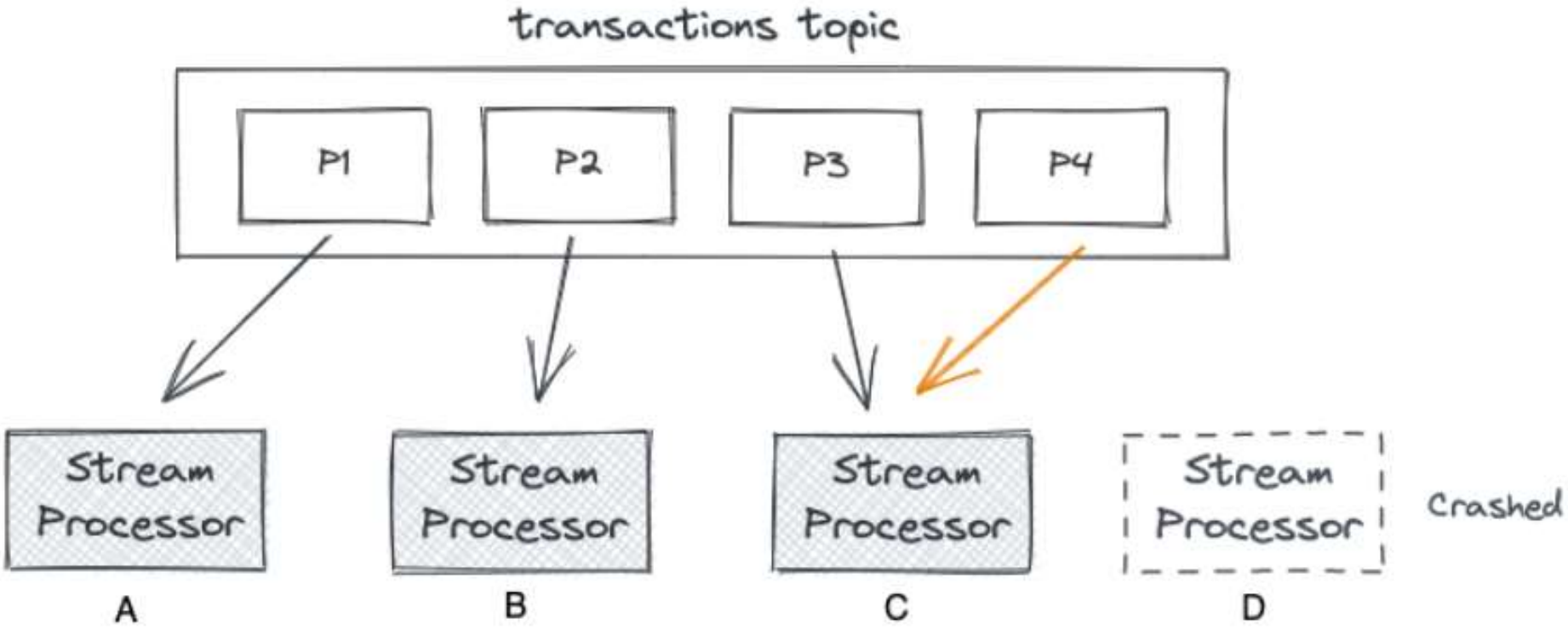
Store	Total
1	870
2	543
...	...
1000	345

Store	Total
1001	123
1002	59
...	...
2000	476

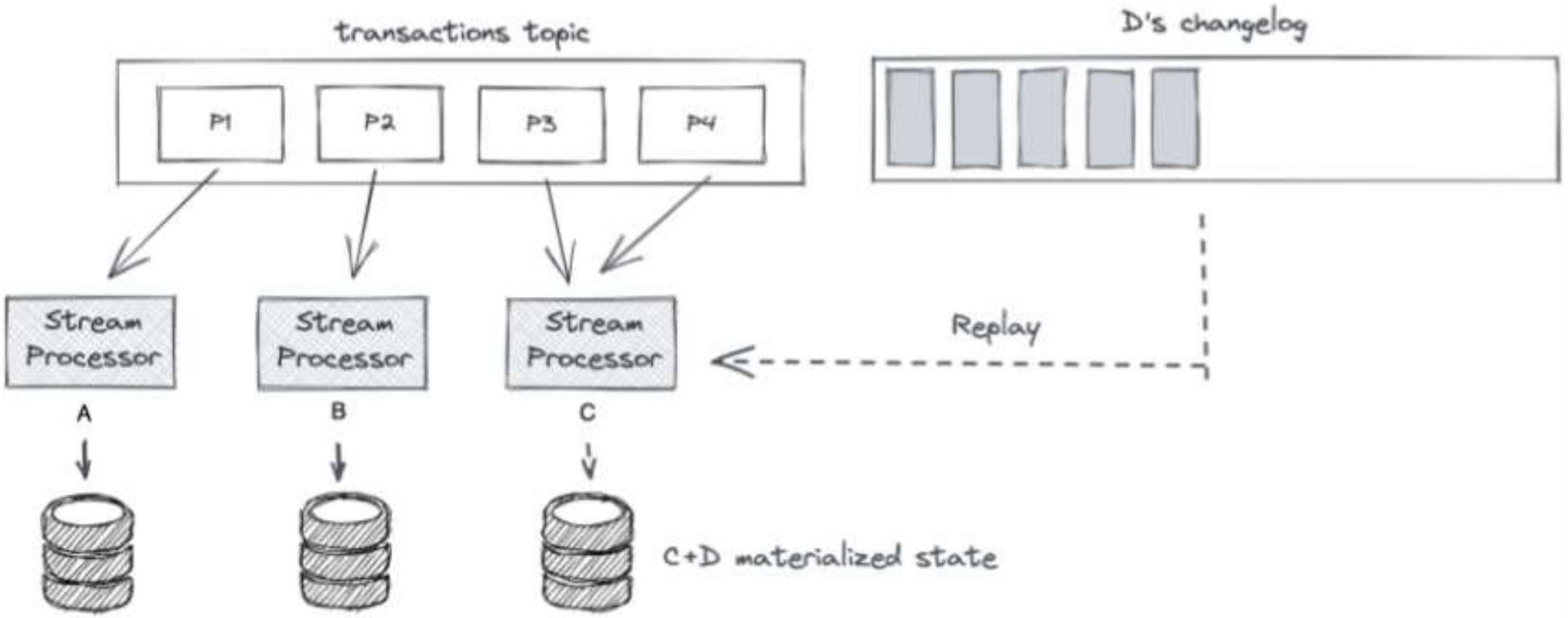
Store	Total
2001	768
2002	678
...	...
3000	900

Store	Total
3001	945
3002	1000
...	...
4000	456

Kafka Streams & Fault Tolerance: an example



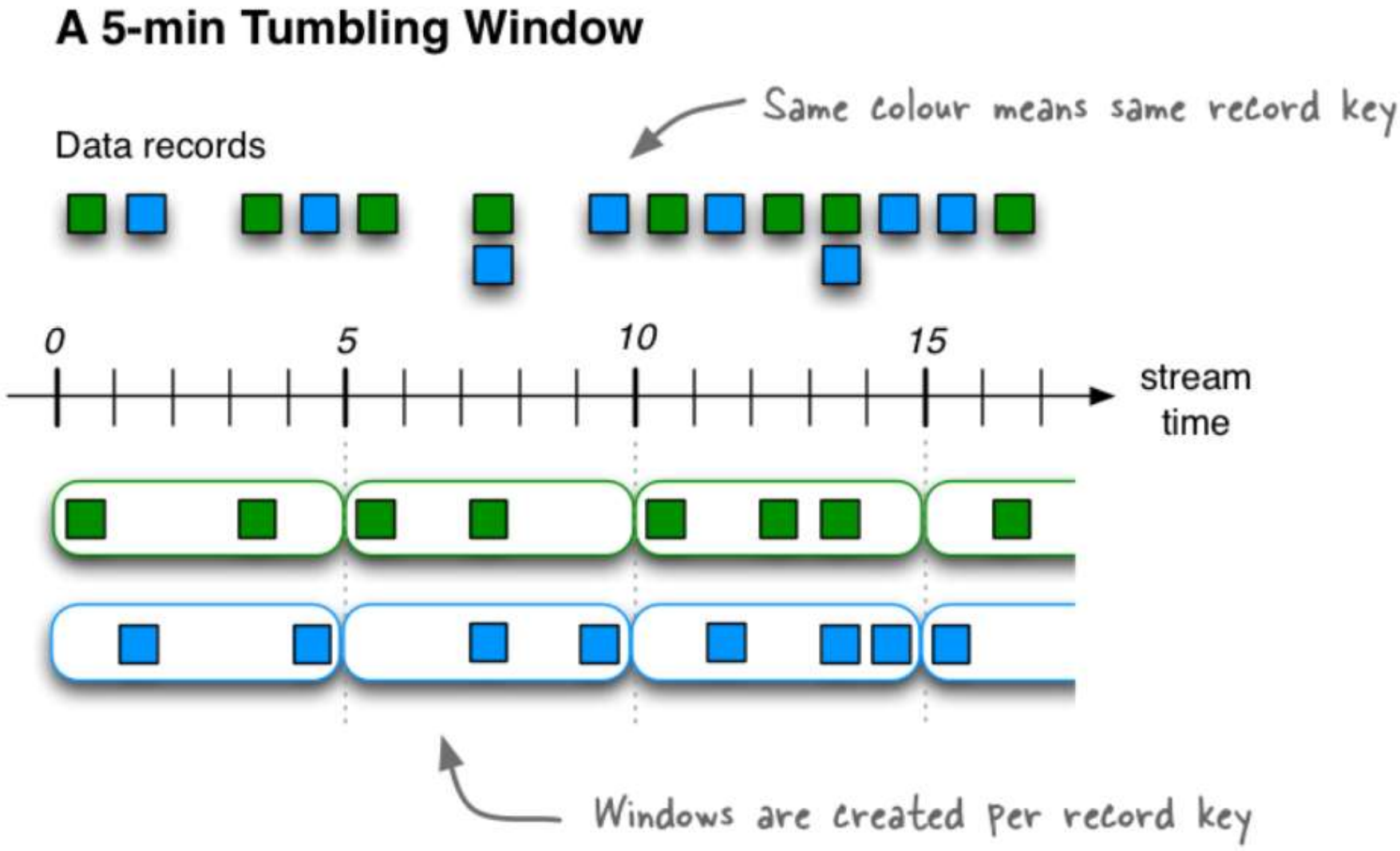
Kafka Streams & Fault Tolerance: an example



Kafka Streams
(Windowing)

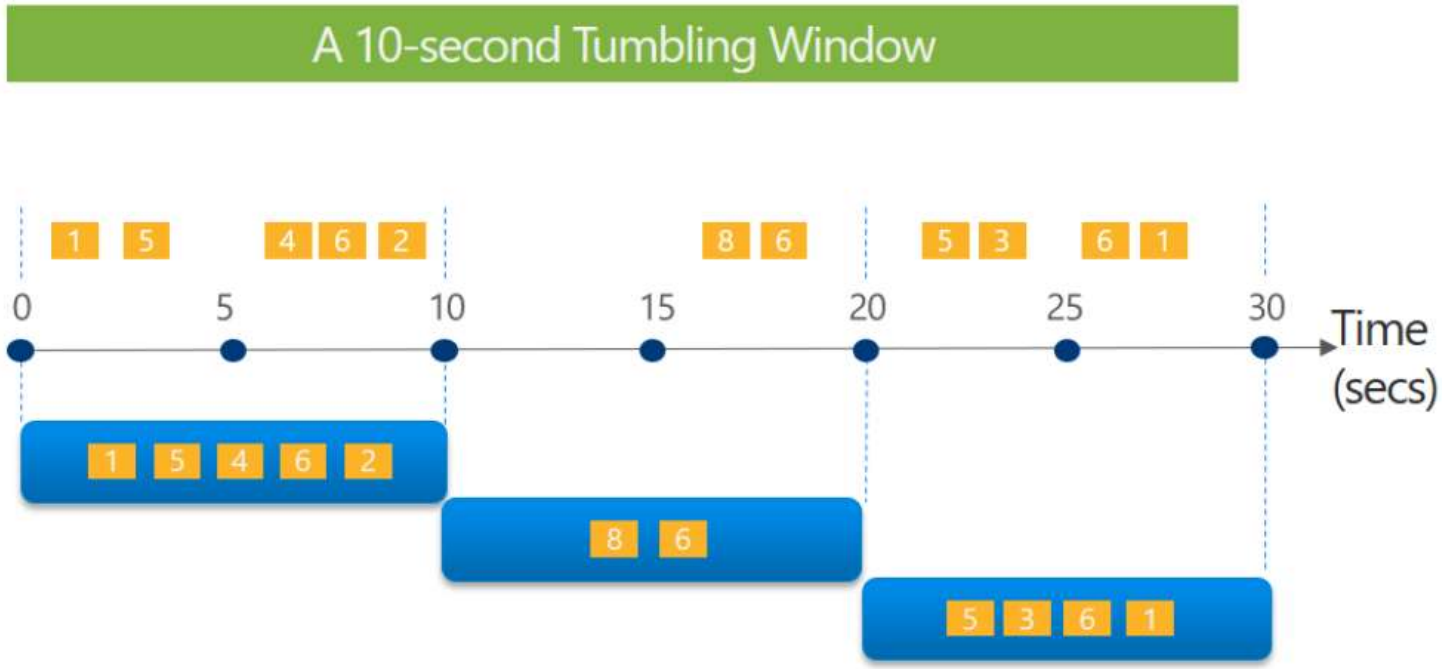
Stateful operations: Tumbling Windows

Tumbling Window: Fixed-size, non-overlapping windows



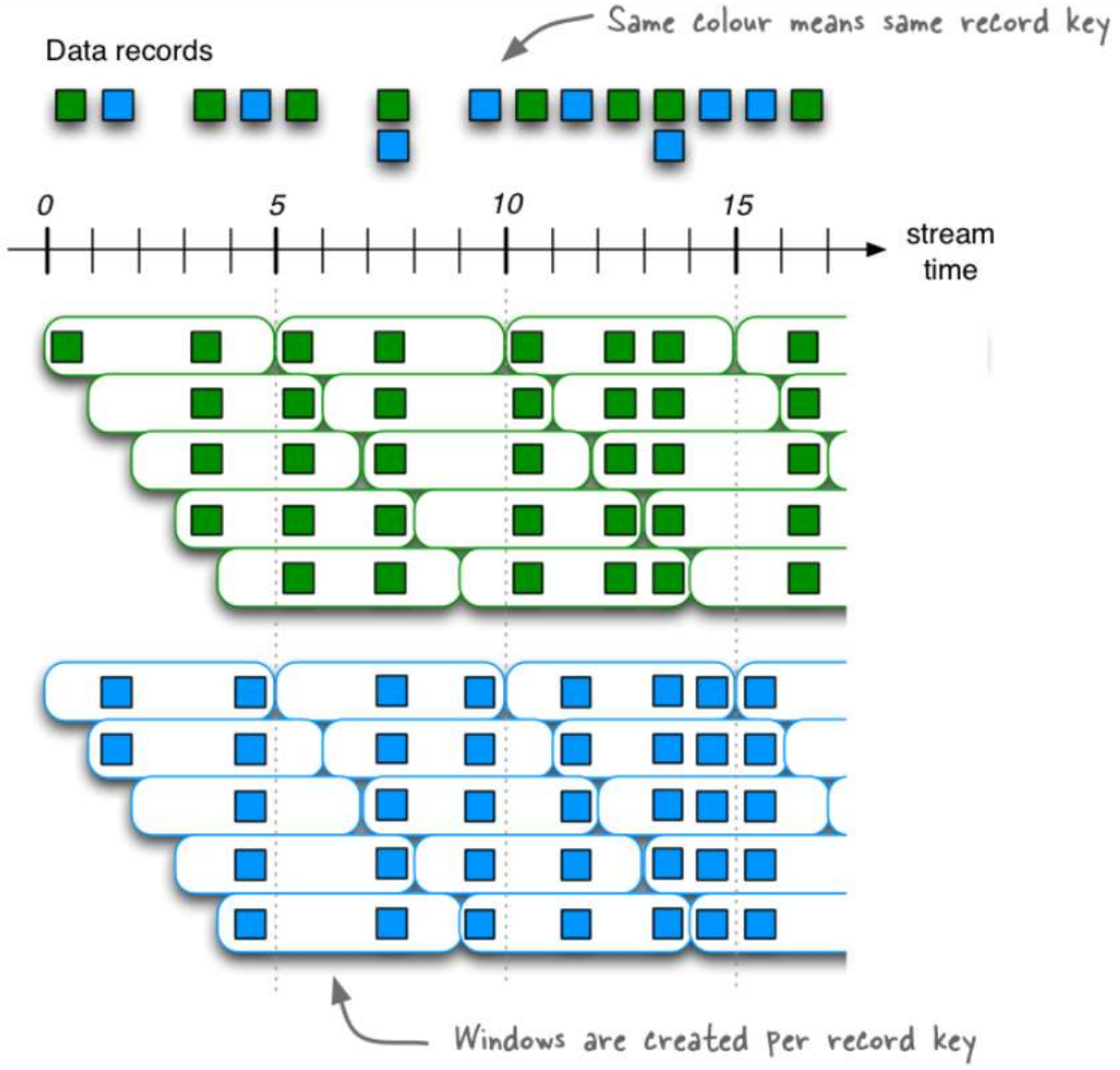
Stateful operations: Tumbling Windows

Tell me the count of Tweets per time zone every 10 seconds



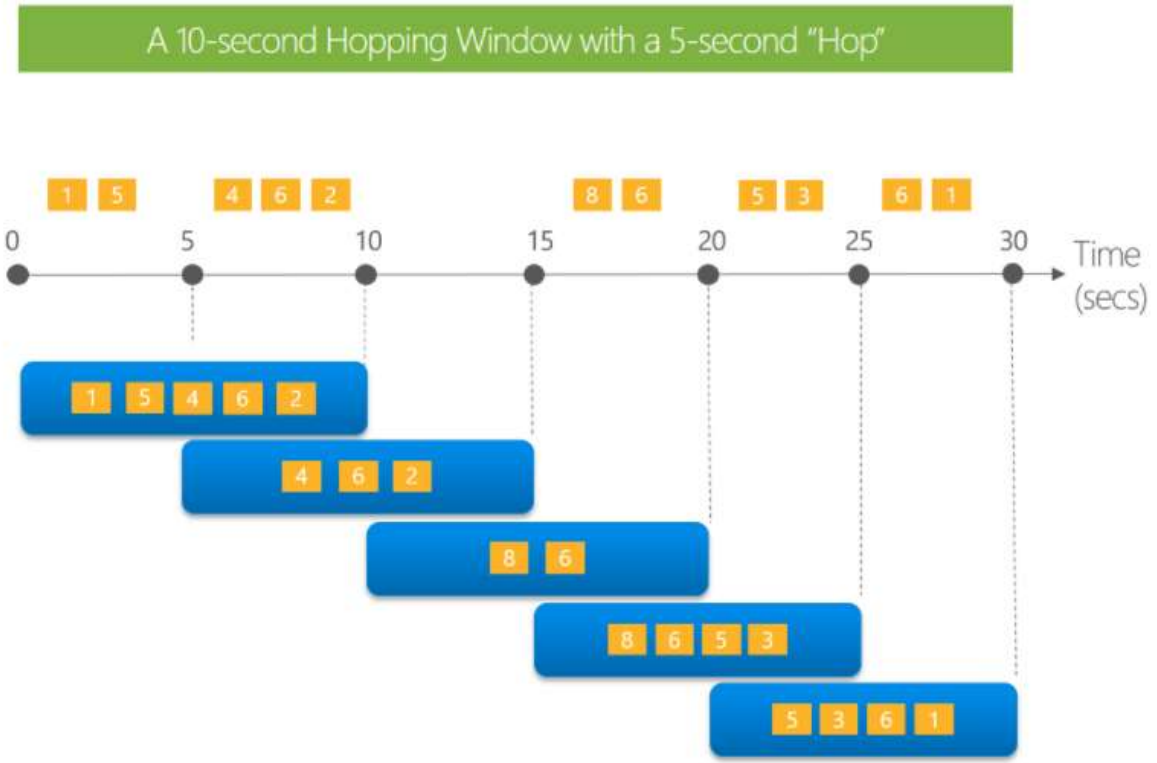
Stateful operations: Hopping Windows

Hopping Window: Fixed-size, overlapping windows



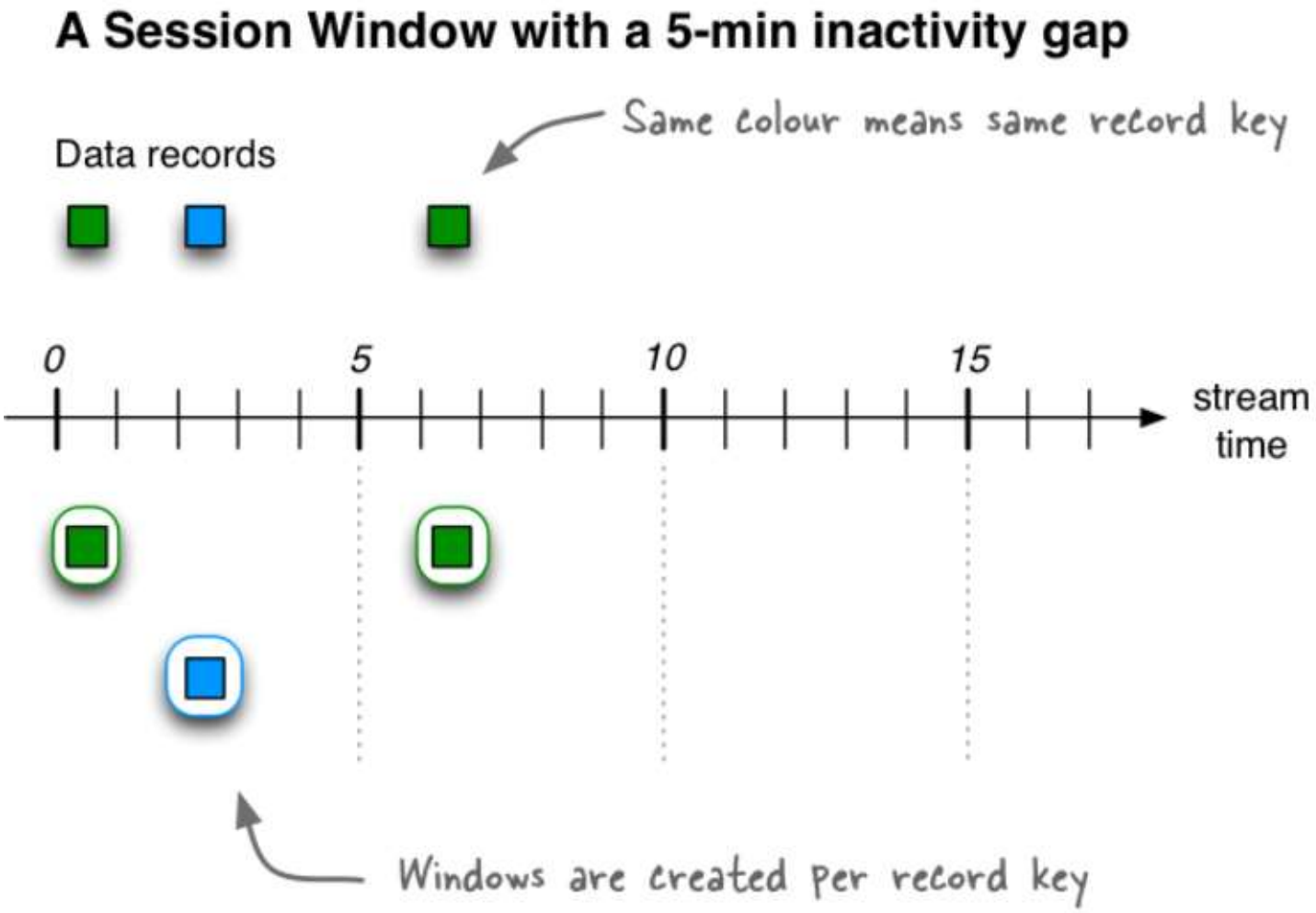
Stateful operations: Hopping Windows

Every 5 seconds give me the count of Tweets over the last 10 seconds



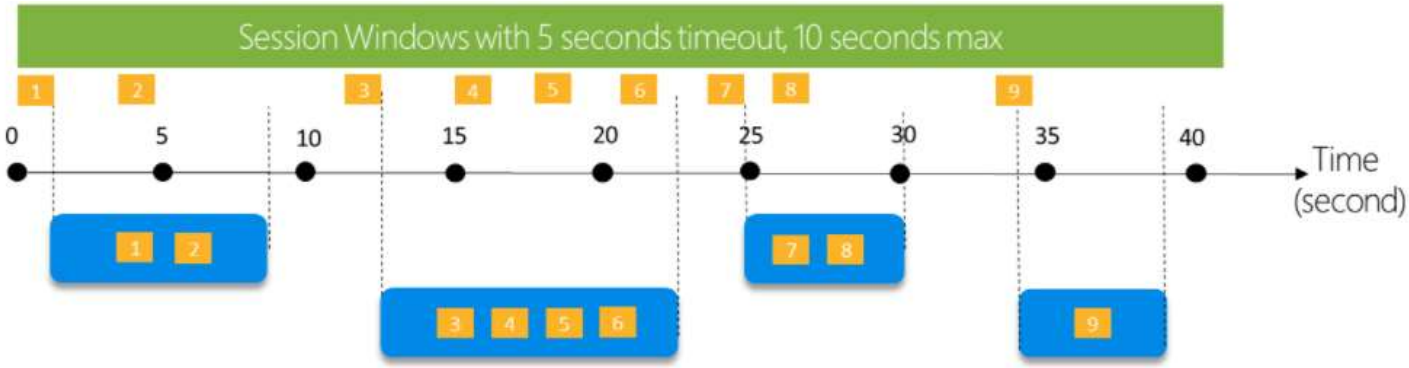
Stateful operations: Session Windows

Session Window: Dynamically-sized, non-overlapping, data-driven windows



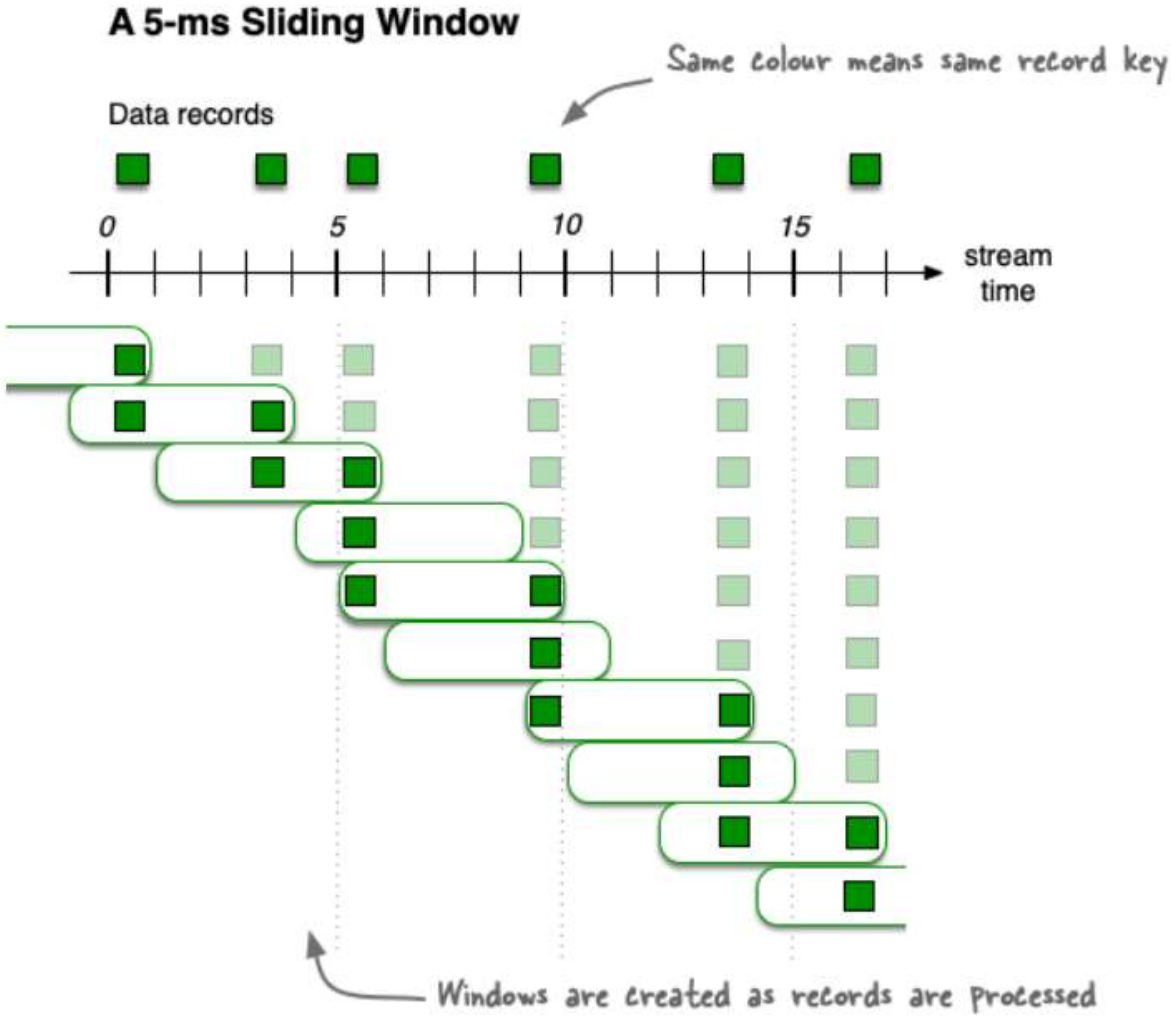
Stateful operations: Session Windows

Tell me the count of Tweets that occur within 5 seconds of each other



Stateful operations: Sliding Windows

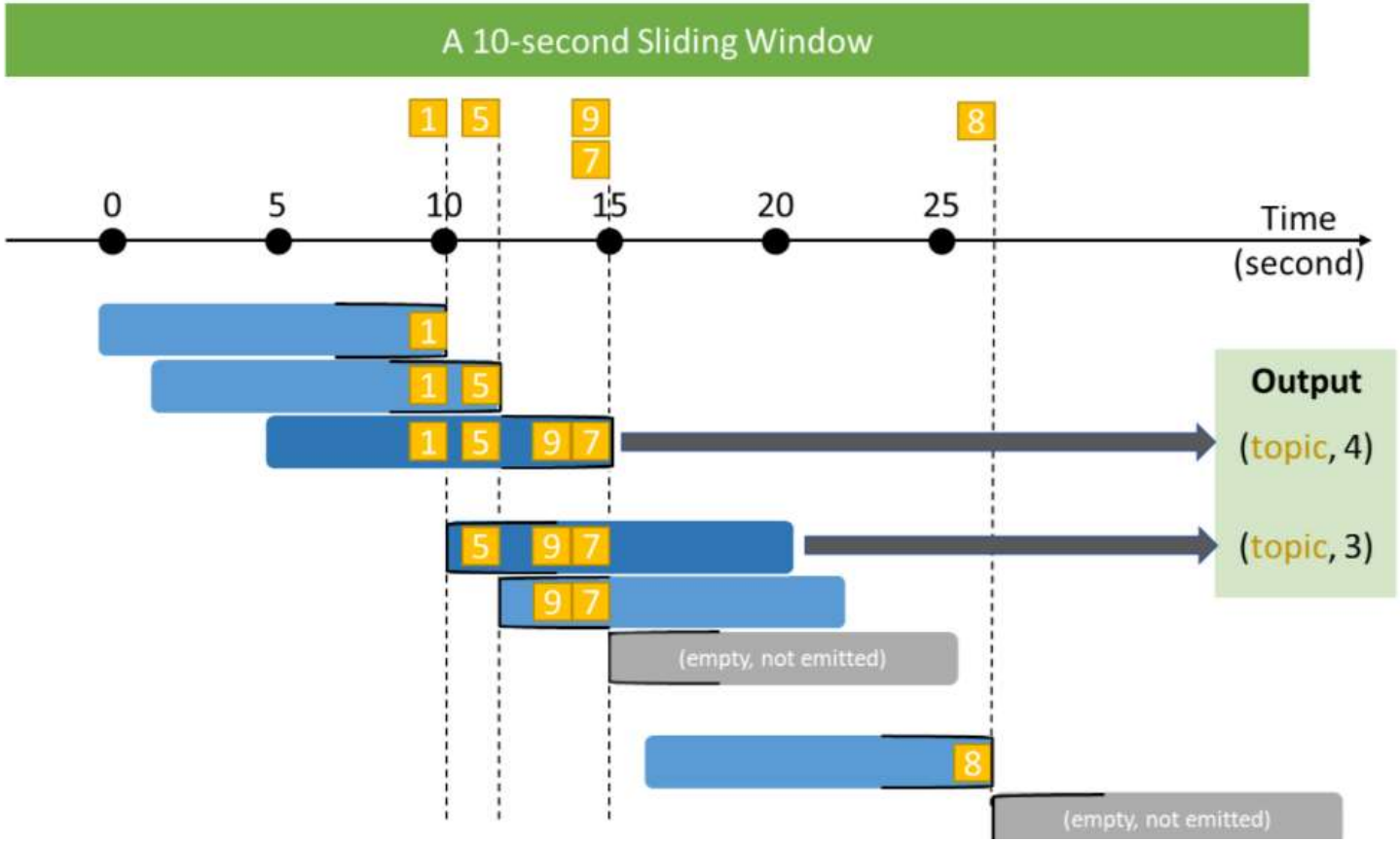
Sliding Window: Fixed-size, overlapping windows that work on differences between record timestamps



Stateful operations: Sliding Windows

Alert me whenever a topic is mentioned more than 3 times in under 10 seconds

Note:
- all tweets on the diagram belong to the same topic



IMPORTANT: A new window is created each time a record enters the sliding window or a record drops out of the sliding window.

Stateful operations: Sliding Windows

For example, if we have a time difference of 5000ms and the following data arrives:

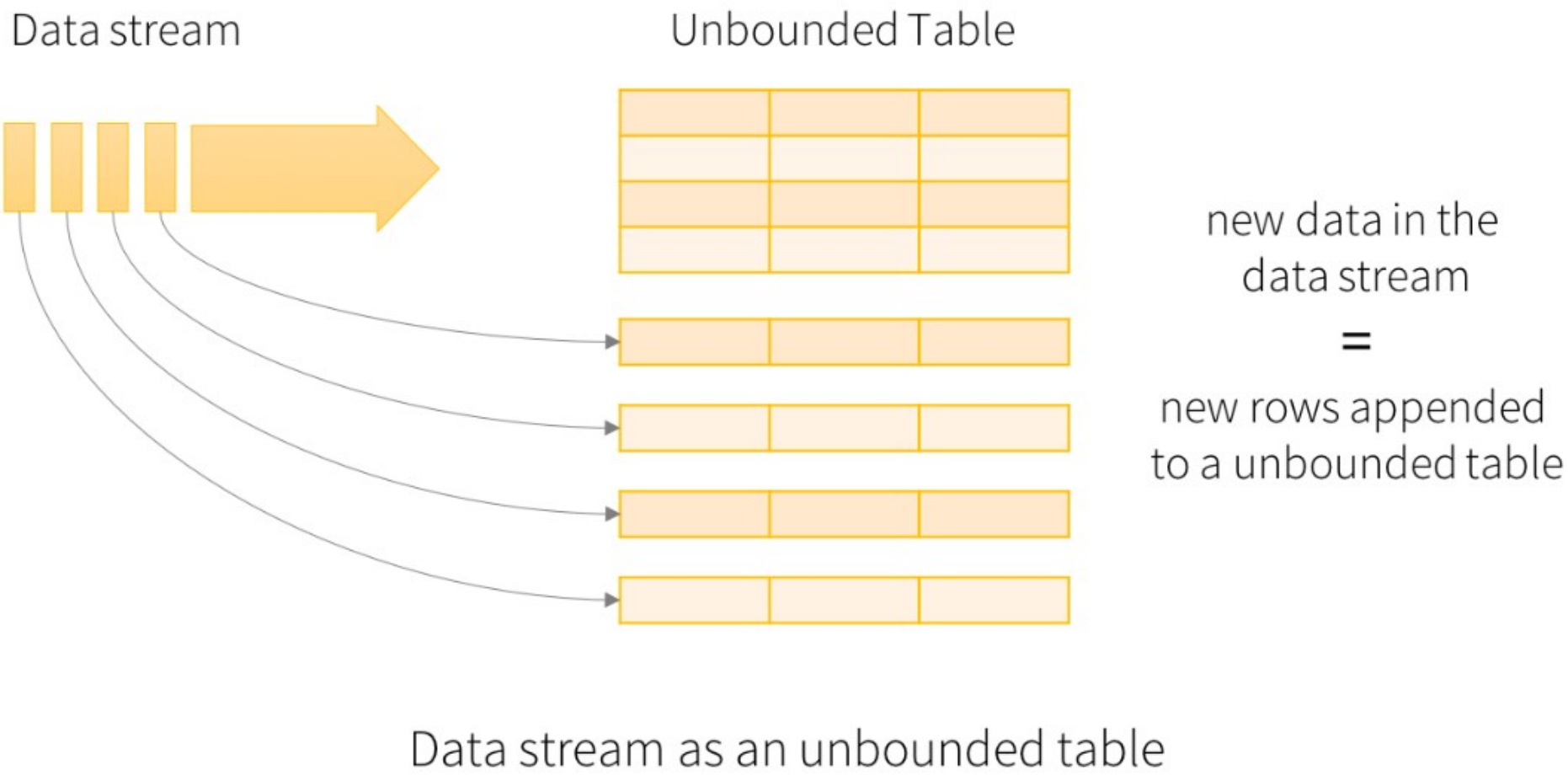
key	value	time
A	1	8000
A	2	9200
A	3	12400

We'd have the following 5 windows:

- window `[3000;8000]` contains [1] (created when first record enters the window)
- window `[4200;9200]` contains [1,2] (created when second record enters the window)
- window `[7400;12400]` contains [1,2,3] (created when third record enters the window)
- window `[8001;13001]` contains [2,3] (created when the first record drops out of the window)
- window `[9201;14201]` contains [3] (created when the second record drops out of the window)

Source: https://kafka.apache.org/27/javadoc/org/apache/kafka/streams/kstream/SlidingWindows.html

Kafka Streams vs. Spark Streaming



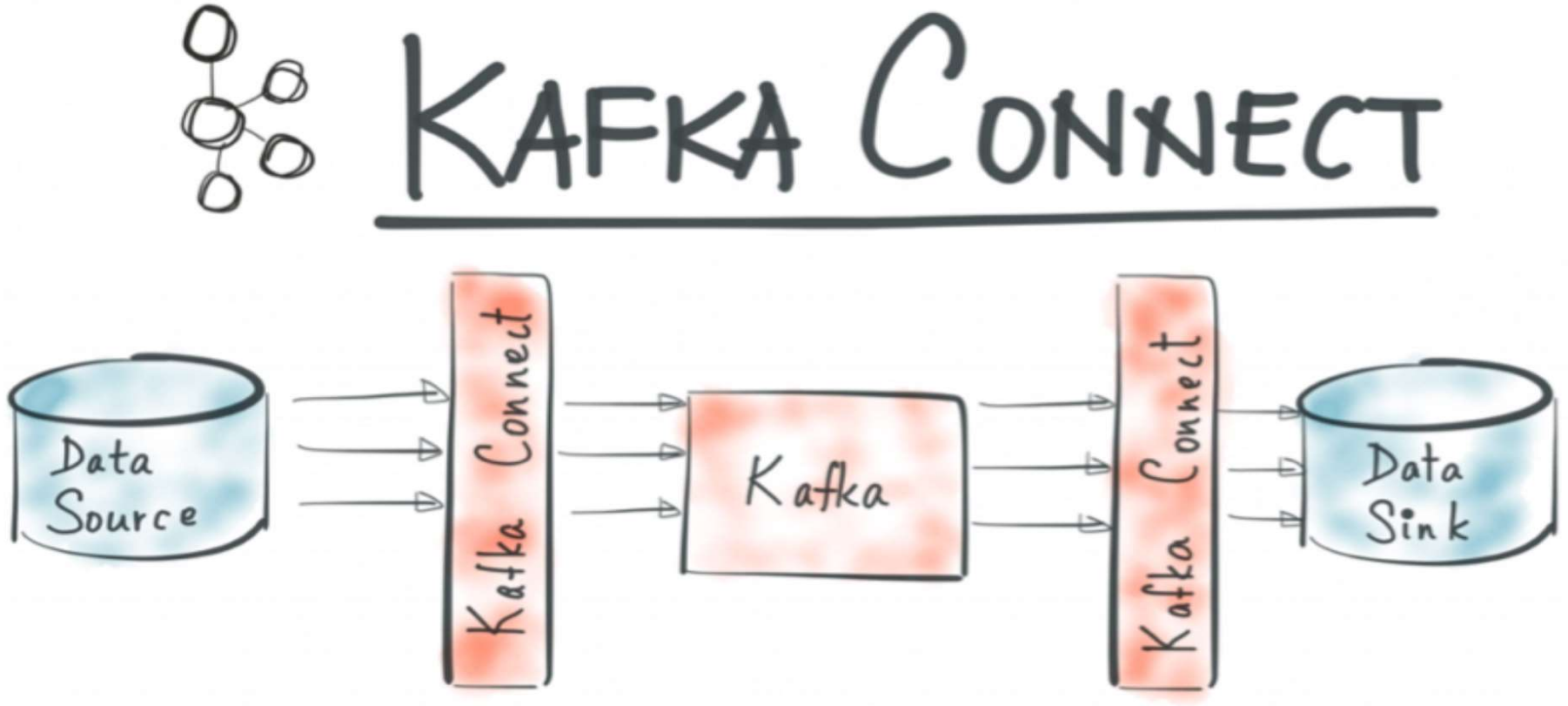
Data stream as an unbounded table

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html

https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

Kafka Connect

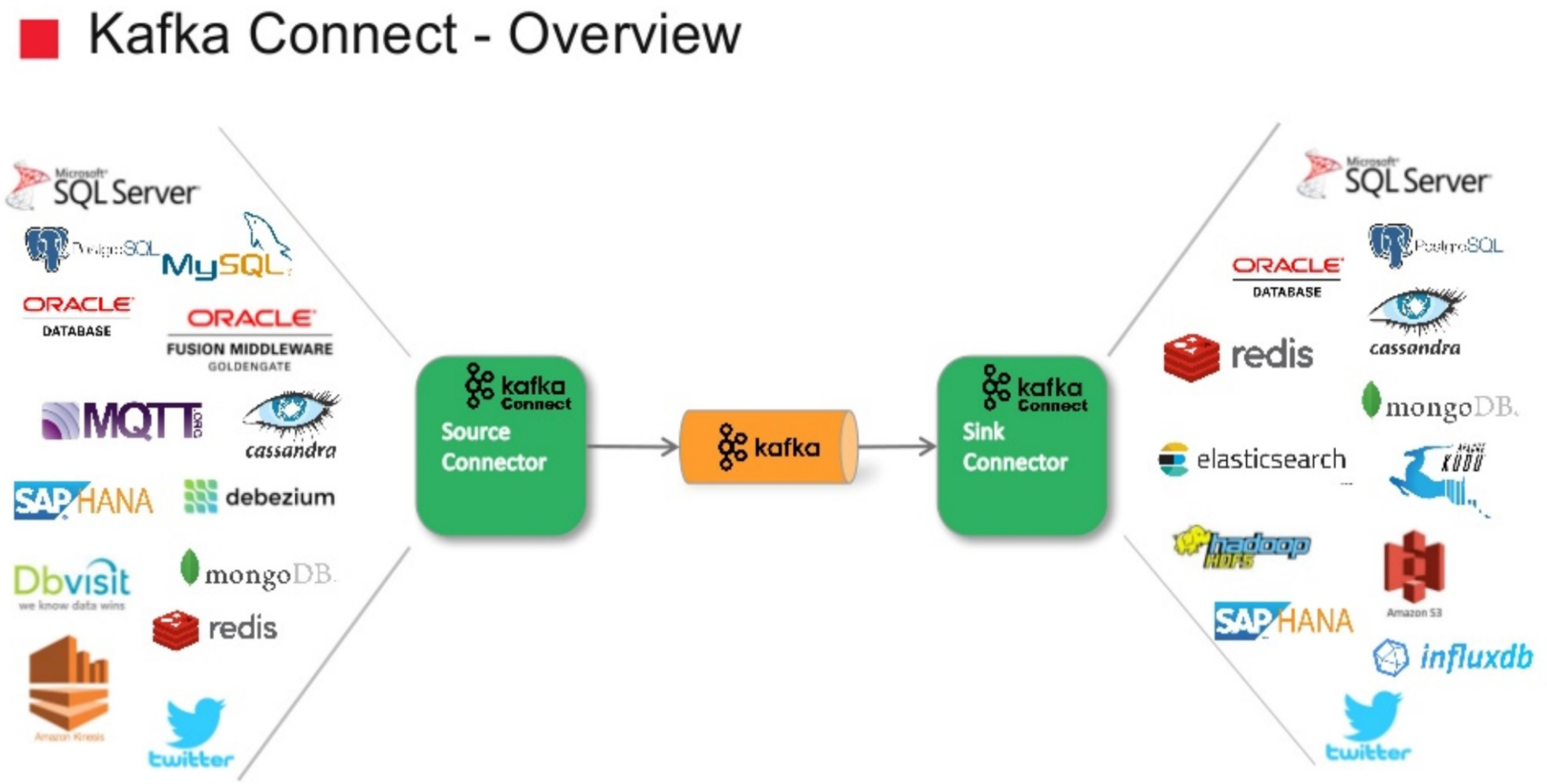
- **JDBC connector** for **Kafka connect**
- Use **CDC (Change data capture)** tool which integrates with **kafka connect.**



Kafka Connect

Kafka Connect is a tool for scalably and reliably streaming data between Apache **Kafka** and other data systems.

Runs separately from Kafka brokers.



References

https://www.baeldung.com/java-kafka-streams-vs-kafka-consumer

https://docs.confluent.io/platform/current/streams/

https://kafka.apache.org/documentation/streams/developer-guide/dsl-api.html#streams-developer-guide-dsl-windowing

https://github.com/sj666/df_stream_kafka/blob/master/README.md

https://medium.com/event-driven-utopia/understanding-materialized-views-part-2-ae957d40a403

Matteo Nardelli, Kafka Streams: Hands-on Session