

Workshop Kafka Streams



A blue gradient polygon shape, resembling a stylized arrow or a corner of a document, pointing towards the bottom right. The color transitions from a lighter cyan on the left to a darker blue on the right. The word "Présentation" is written in white, bold, sans-serif font, slanted upwards to the right, positioned within the lighter part of the shape.

Présentation



Staff Software Engineer @ManoMano

Confluent Certified Trainer

victor.gallet@manomano.com

<https://vgallet.github.io/>

@GalletVictor



<https://github.com/vgallet/workshop-kafka-streams>

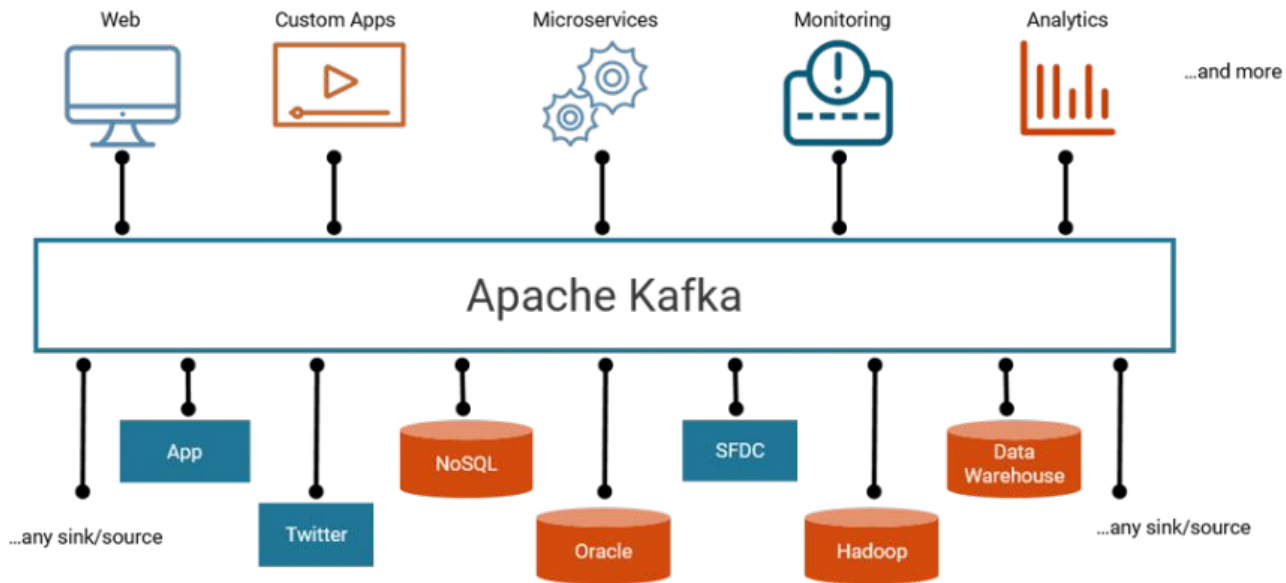
A large, abstract, multi-colored shape in shades of blue and cyan occupies the lower-left portion of the slide. The shape has rounded corners and a gradient from light cyan on the left to dark blue on the right. The word "Workshop" is written in white, bold, sans-serif font, tilted slightly upwards to the right, within the dark blue portion of the shape.

Workshop

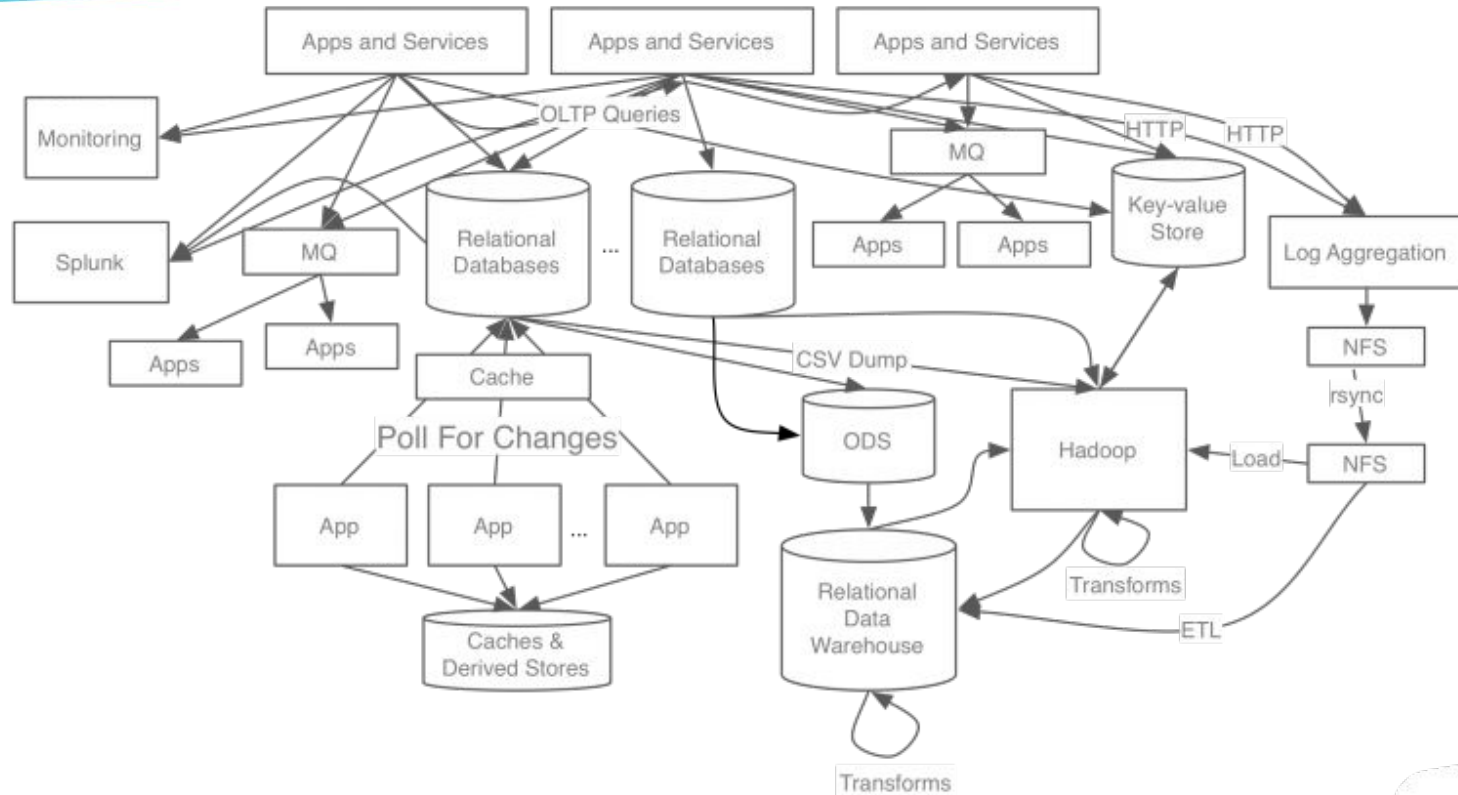
A blue gradient pentagon shape, tilted to the right, with a lighter blue on the left and a darker blue on the right. It contains the text 'Kafka Fundamentals' in white.

Kafka Fundamentals

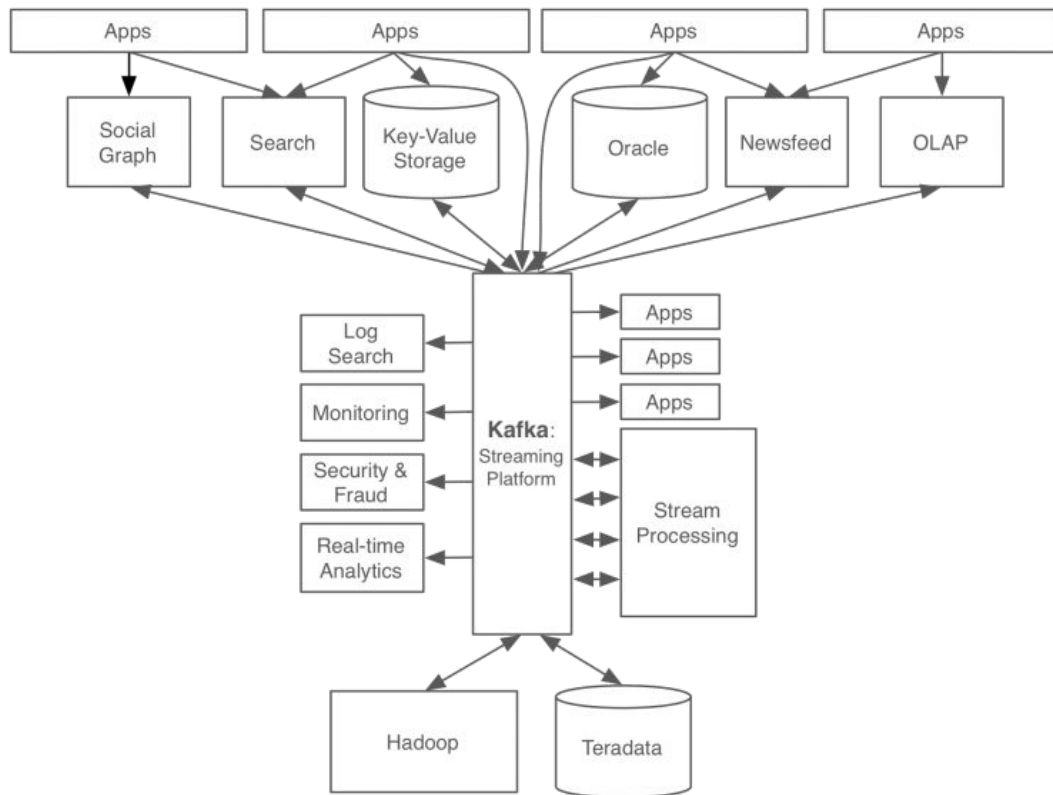
Streaming Platform



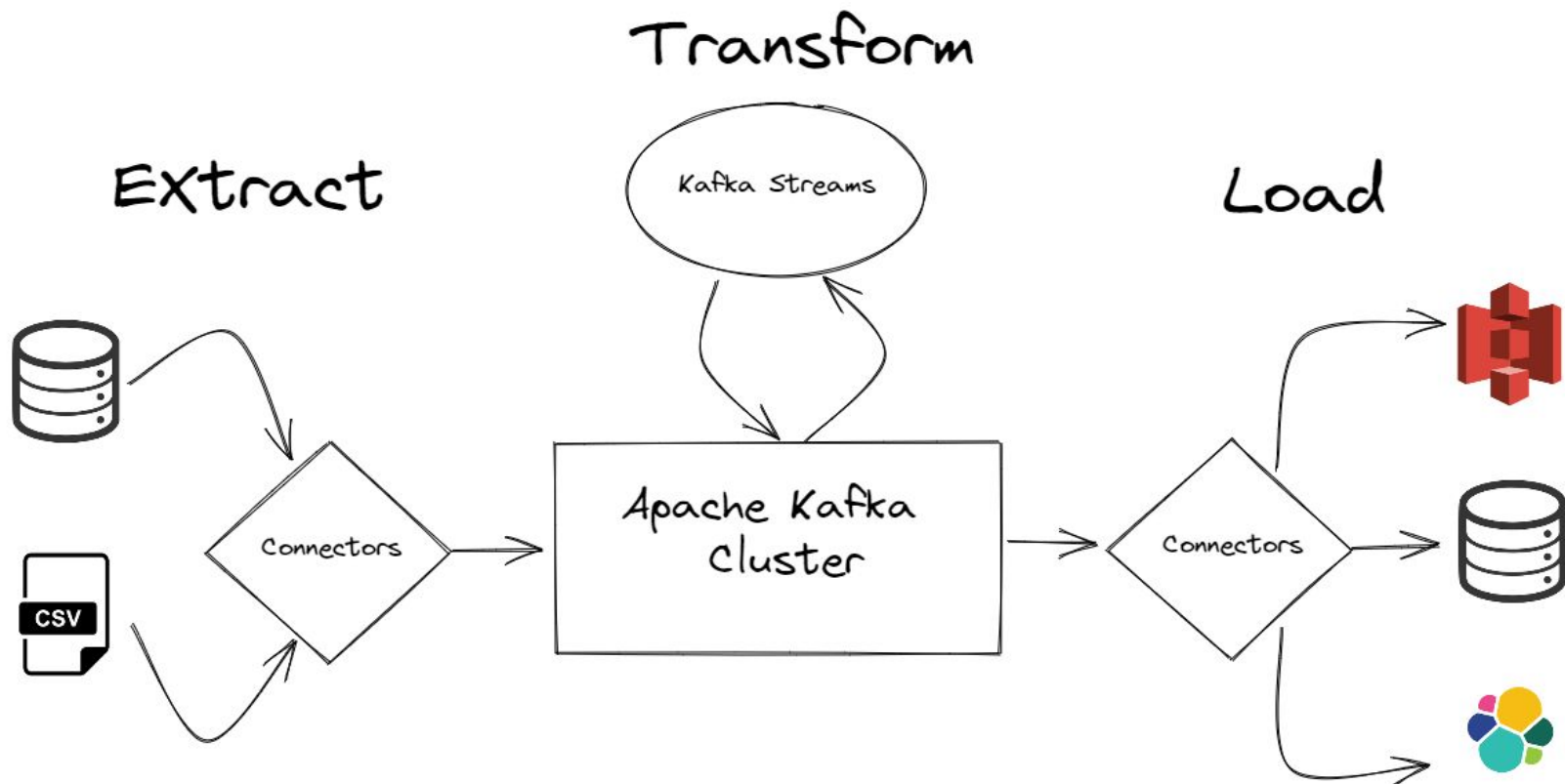
Streaming Platform



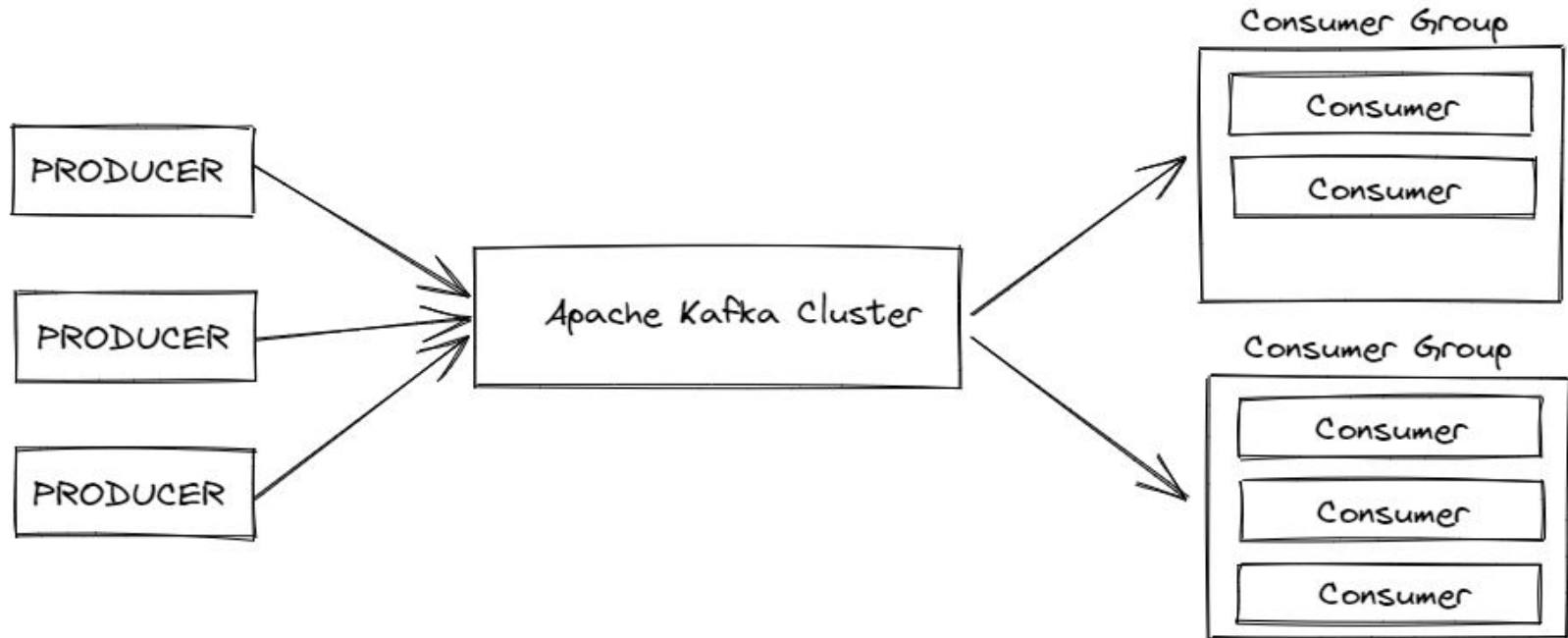
Streaming Platform



Streaming Platform

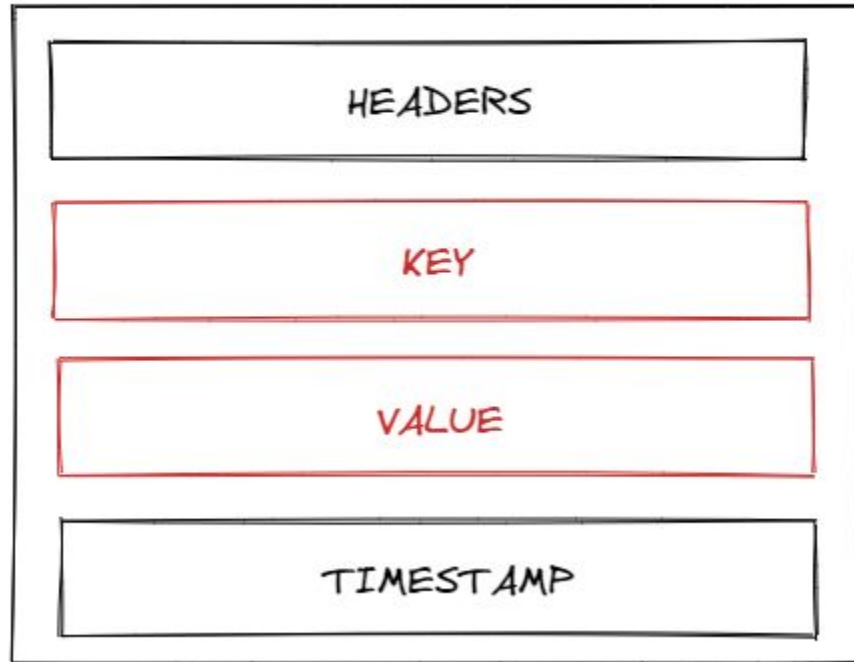


Clients - Producers & Consumers

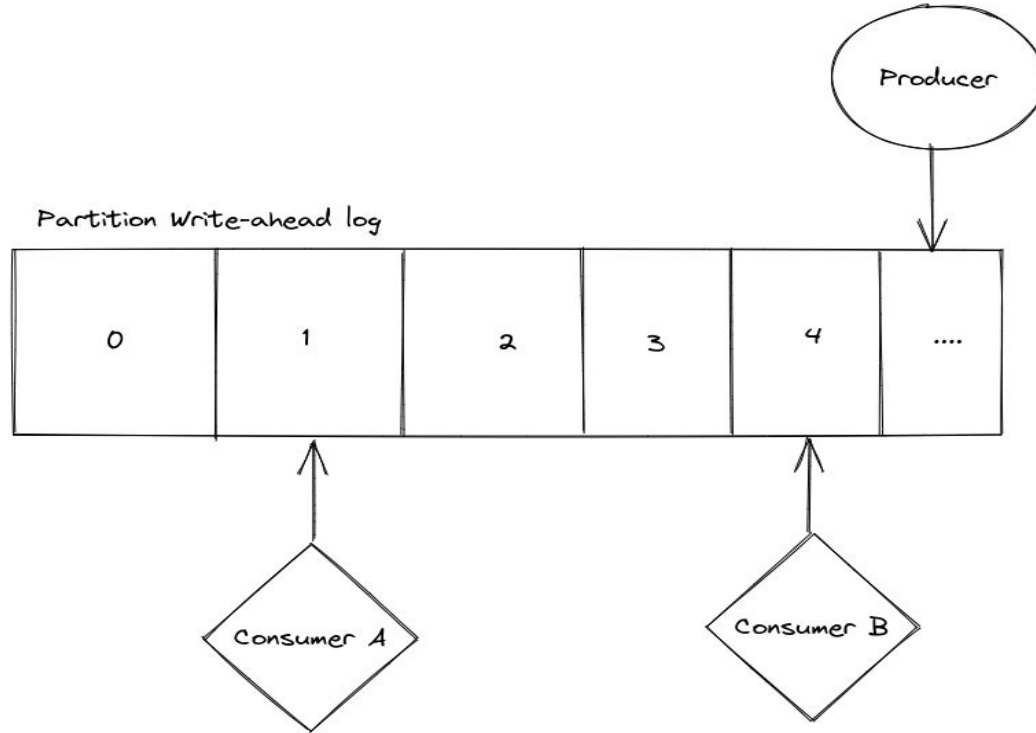


The Record

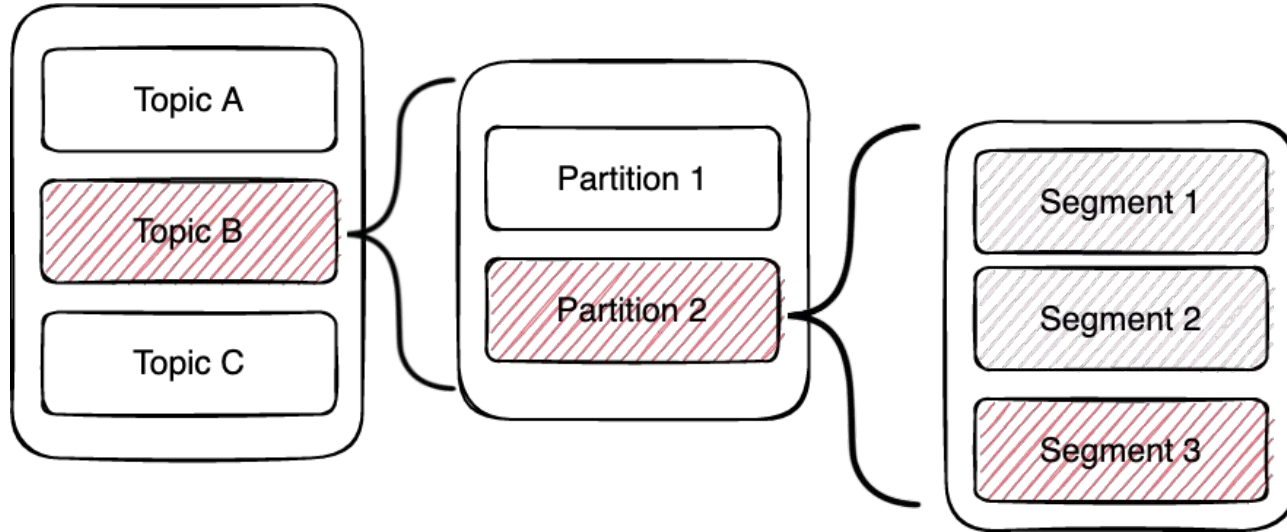
RECORD



The Kafka Commit Log

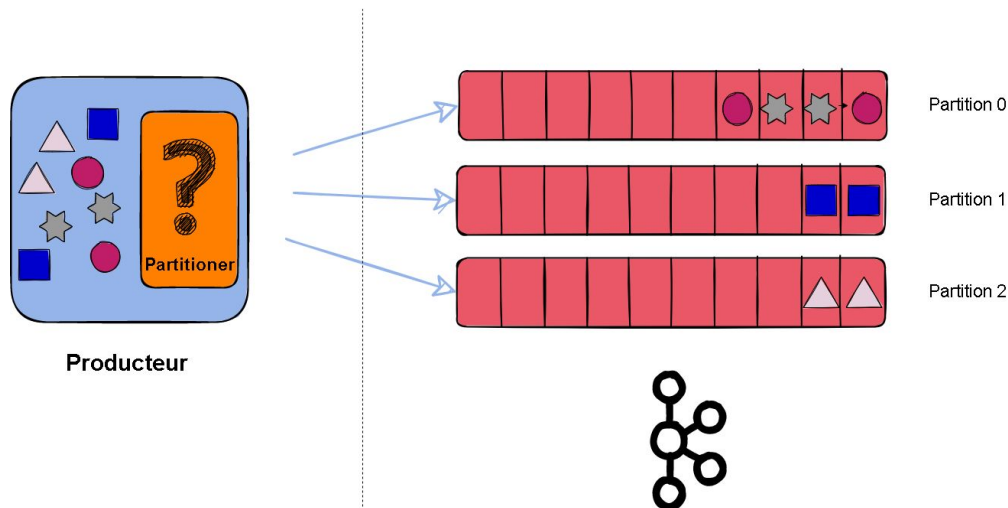


Topic / Partition / Record



Partitioning

$$\text{partition} = \text{hash}(\text{key}) \% \text{nombre de partitions}$$

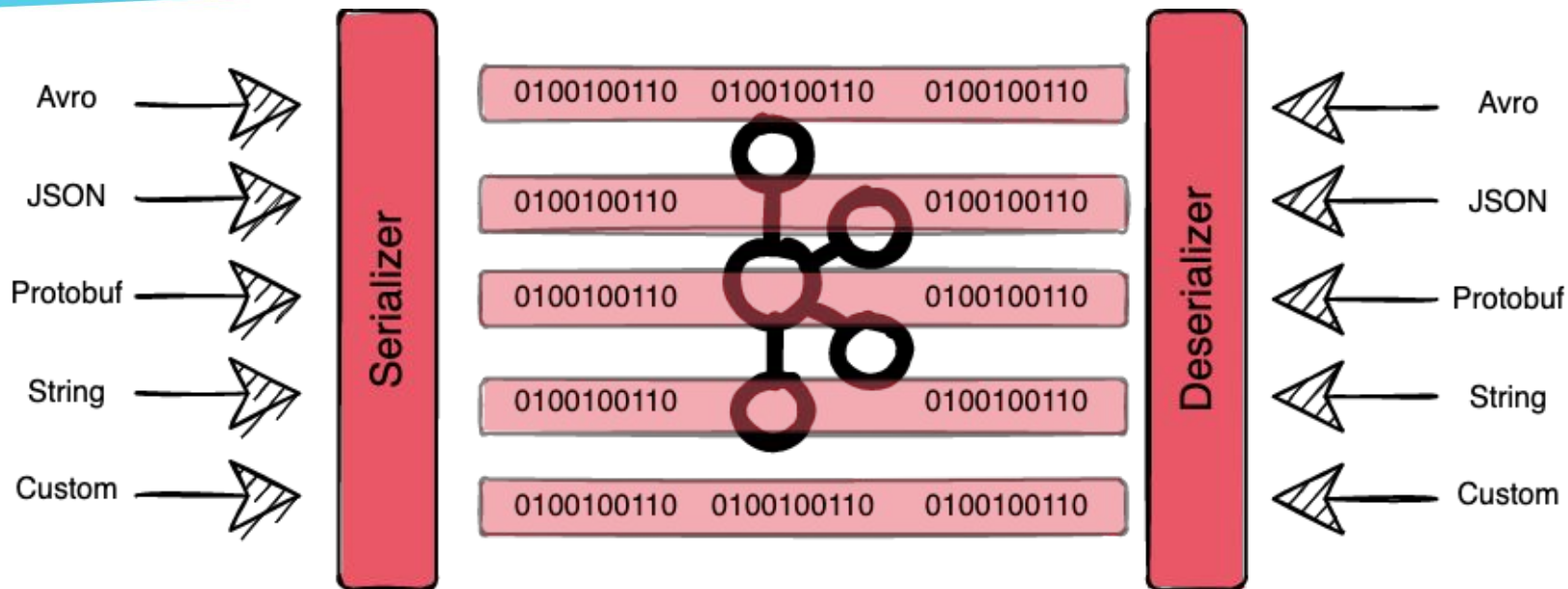


A message is delivered based on the partitioning key

The target partition is determined by the hash (key) modulo the number of partitions

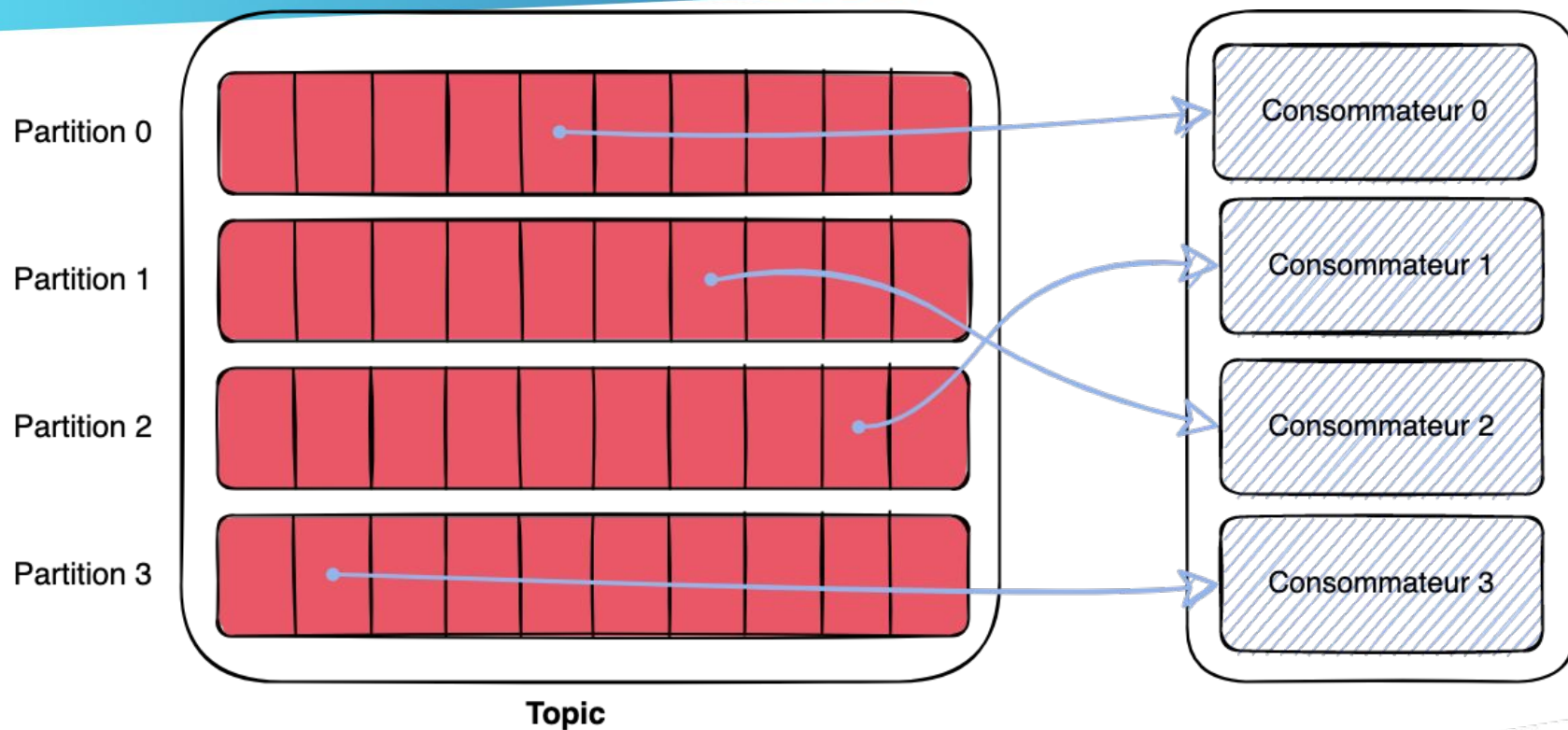
Default - Round Robin if no key is associated with a message

Serialization

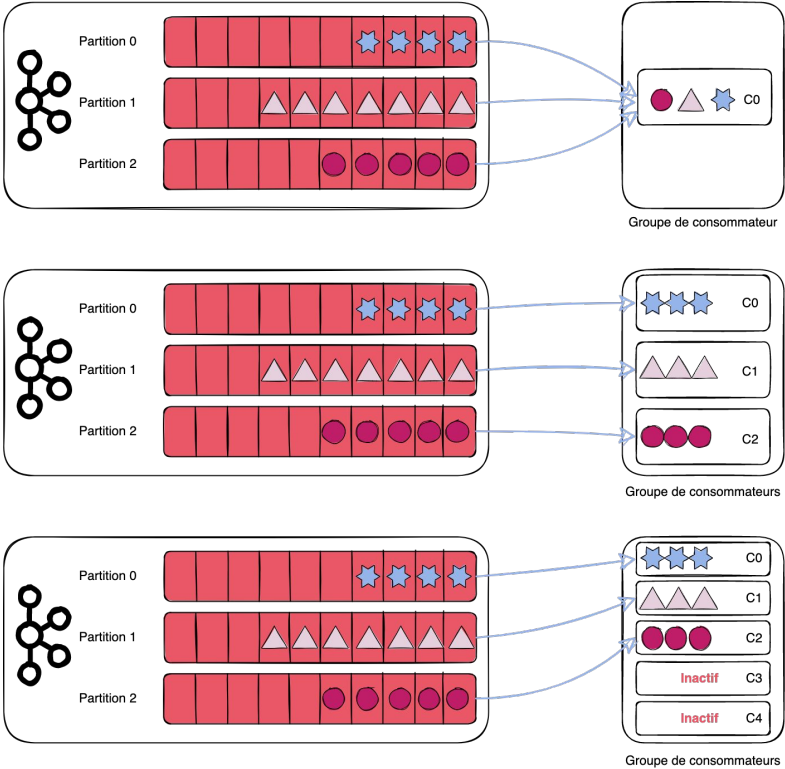


Kafka only knows binary!

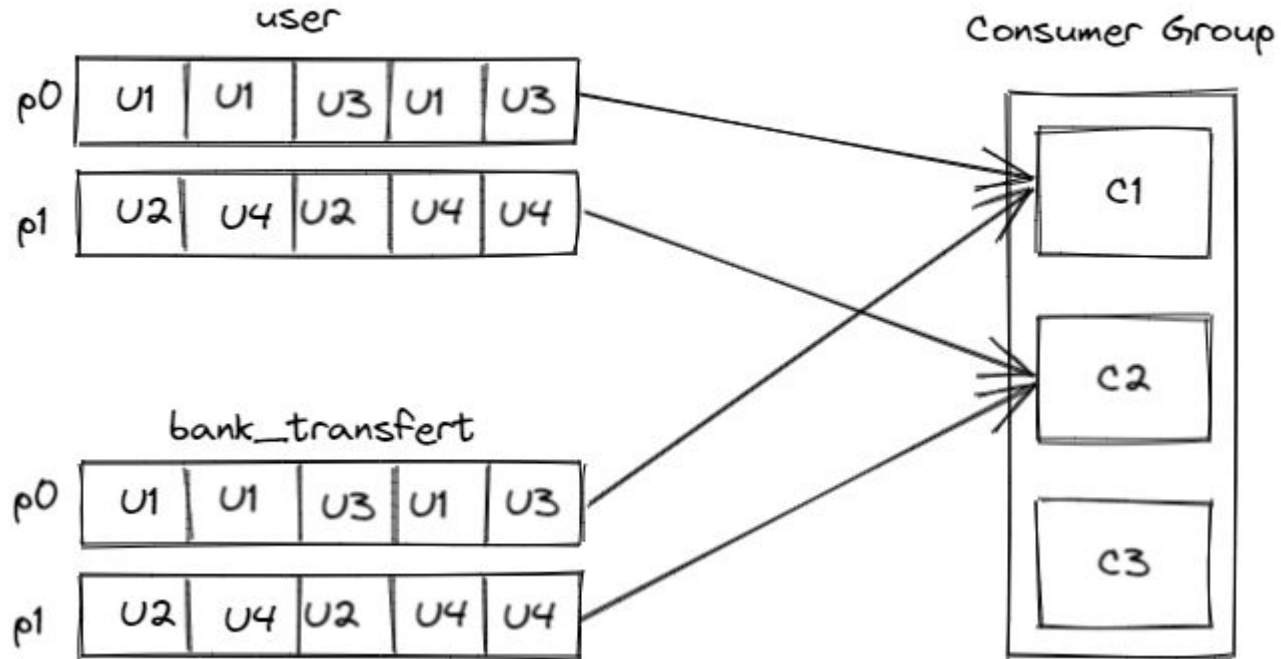
Consumer Group



Consumer Group



Partition Assignment Strategy



Partition Assignment Strategy

- Use **RangeAssignor** when joining data from multiple topics (this is the default):
partition.assignment.strategy=org.apache.kafka.clients.consumer.RangeAssignor
- Use **RoundRobin** when performing stateless operations on records from many topics:
partition.assignment.strategy=org.apache.kafka.clients.consumer.RoundRobinAssignor
- Sticky is **RoundRobin** with a best effort to maintain assignments across rebalances:
partition.assignment.strategy=org.apache.kafka.clients.consumer.StickyAssignor
- **CooperativeSticky** is Sticky but it uses consecutive rebalances rather than the single stop-the-world used by Sticky (the next default):
partition.assignment.strategy=org.apache.kafka.clients.consumer.CooperativeStickyAssignor

Topic Retention Policy

We don't want topics to grow forever!

cleanup.policy

- *delete*
- *compact*
- both: *delete,compact*

Compacted Topic

offset	0	1	2	3	4	5	6	7
key	K1	K2	K3	K1	K1	K3	K2	K2
value	V1	V2	V3	V4	V5	V6	V7	V8



offset	4	5	7
key	K1	K3	K2
value	V5	V6	V8

A blue gradient polygon shape, resembling a stylized arrow or a folded piece of paper, pointing towards the bottom right. It has a light blue top-left corner and a darker blue bottom-right corner, with a smooth gradient in between. The shape is set against a plain white background.

Questions ?

A blue gradient shape, resembling a stylized arrow or a corner of a document, pointing towards the bottom right. The shape has a lighter blue on the left and a darker blue on the right. The text 'Kafka Streams' is written in white, bold, sans-serif font, slanted upwards to the right, and positioned within the lighter blue area of the shape.

Kafka Streams

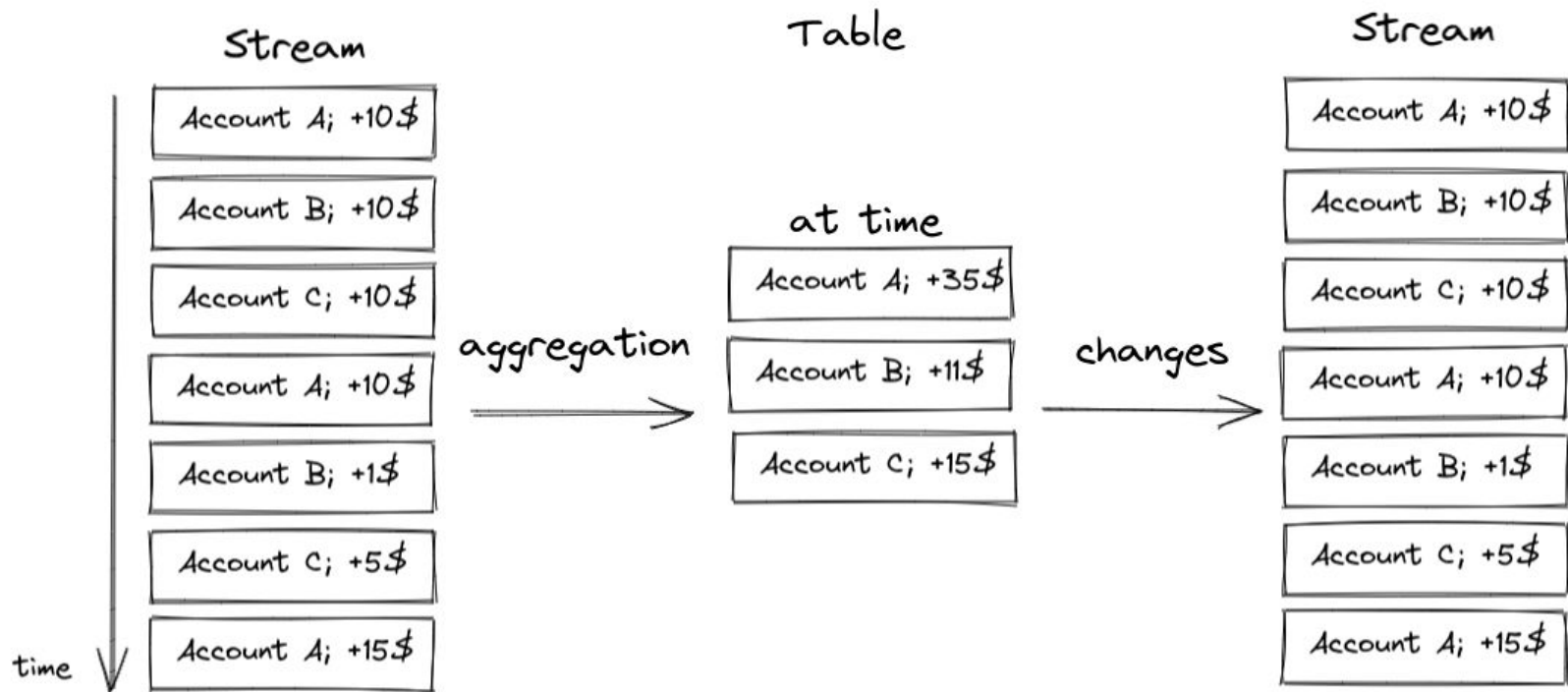
Kafka Streams

- Transforms and enriches data
 - per-record stream processing
 - millisecond latency
 - stateless & stateful processing
 - windowing operations
- Fault-tolerant and distributed processing
- Domain-Specific Language (DSL)
- High level operations: map, filter, count, etc.

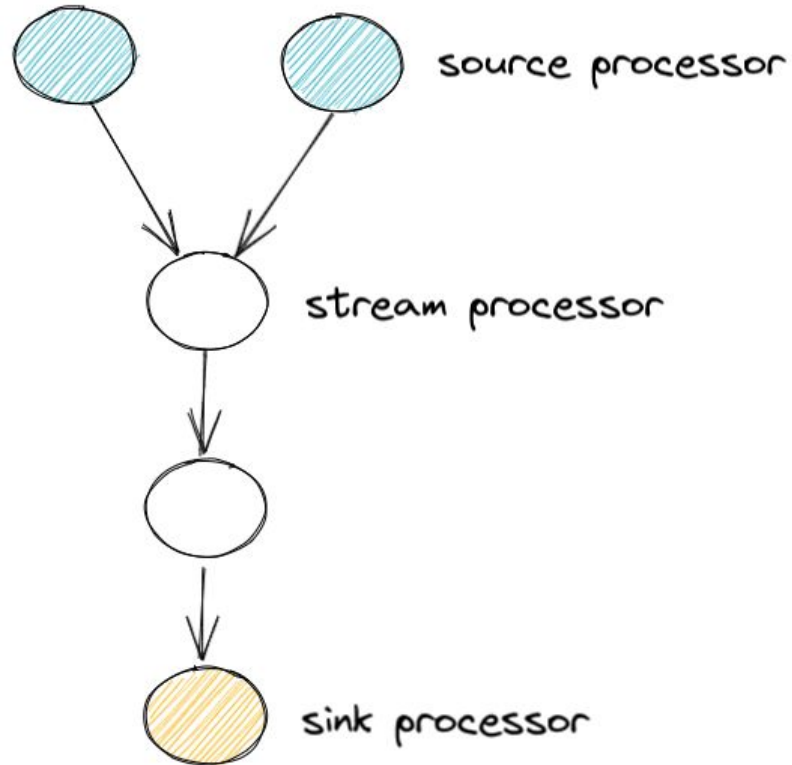
Kafka Streams

- not a framework, just a Java client library for building applications and microservices
- No separate resource management technology required
- Deploy to containers, VMs, bare metal, cloud
- Powered by Kafka: elastic, scalable, distributed, battle-tested
- Perfect for small, medium, large use cases
- Exactly-once processing semantics
- Part of the Apache Kafka project

Stream - Table Duality



Stream Topology



Stateless Processor

filter	<code>KStream<K,V> smallPurchases = purchases.filter((key,value) -> value.amount < 50.0)</code>
mapValues	<code>KStream<K,V> upper = words.mapValues(value -> value.toUpperCase());</code>
flatMapValues	<code>KStream<byte[], String> words = textLines.flatMapValues(sentence -> Arrays.asList(pattern.split(sentence)));</code>

Stateful Processor

count	<p>Counts the number of instances of each key in the stream;</p> <pre>KTable table = stream .groupByKey() .count()</pre>
reduce	<p>Combines values of the stream using a supplied Reducer</p> <pre>KTable table = stream .groupByKey() .reduce(...)</pre>

Windows, Aggregations, and Joins

Windows:

- Divide stream into "time buckets"
- Tumbling, Hopping, and Session windows

Aggregations

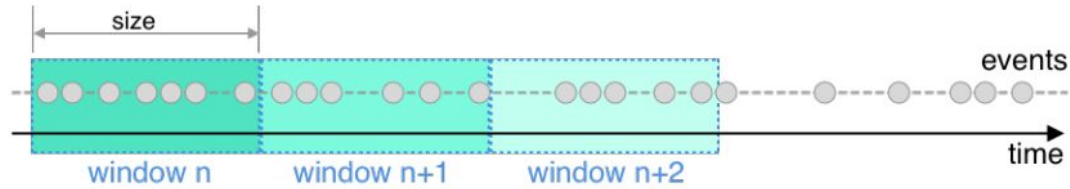
- Accumulate some value as new records come in
- Usually windowed
- Examples: sum, count, max, min

Joins:

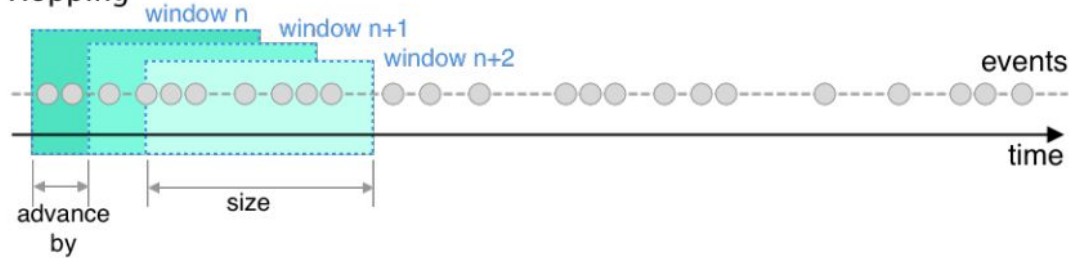
- Combine different streams/tables together on a key
- Can be windowed with a "sliding window"

Windowing Types

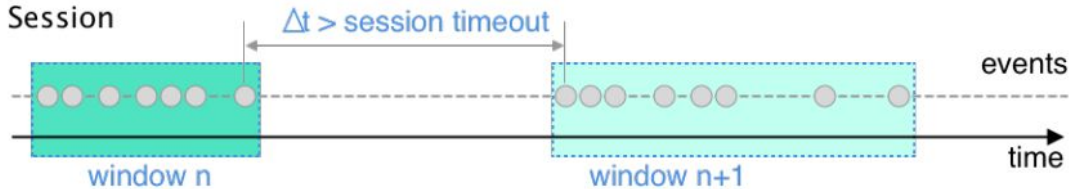
Tumbling



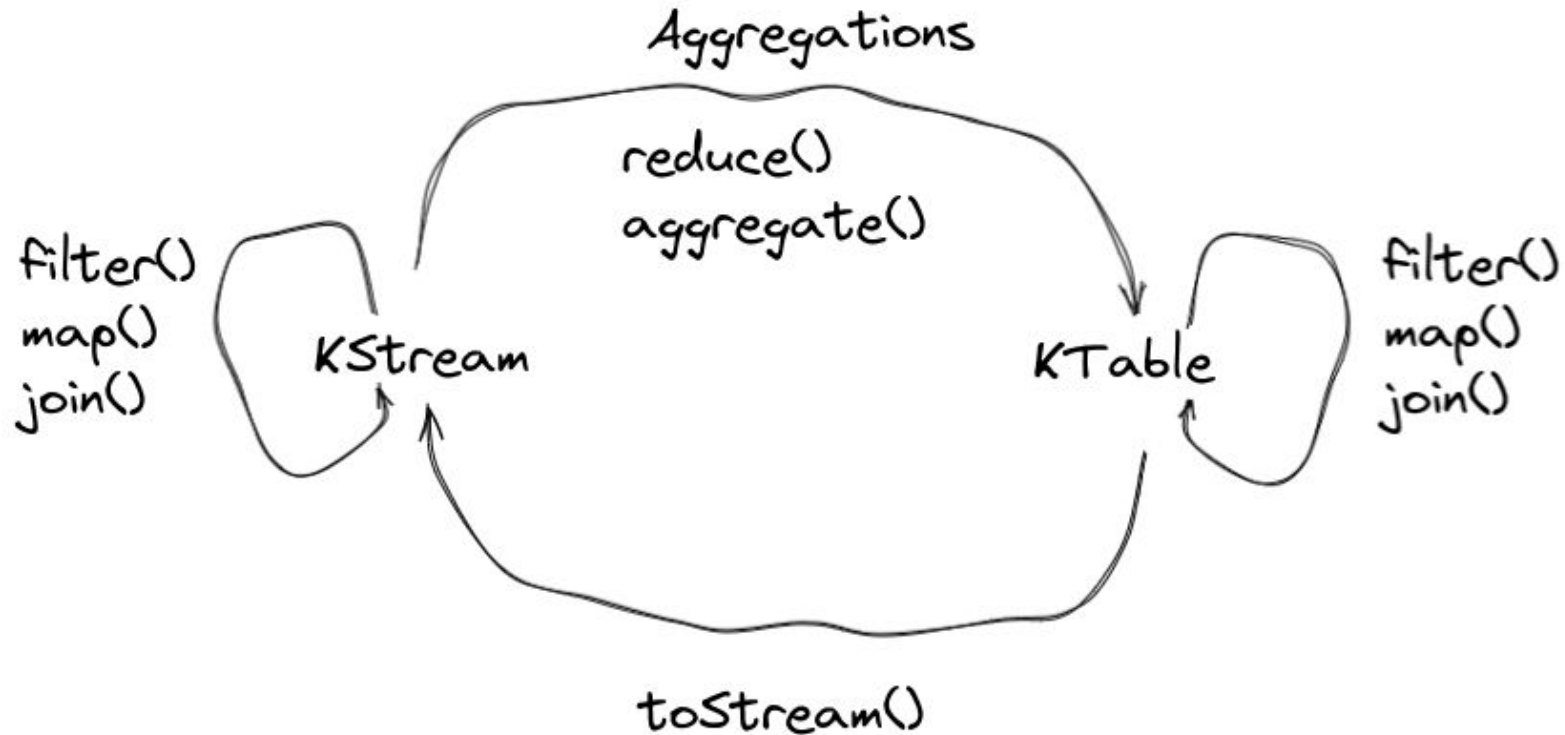
Hopping



Session



Mixed Processing



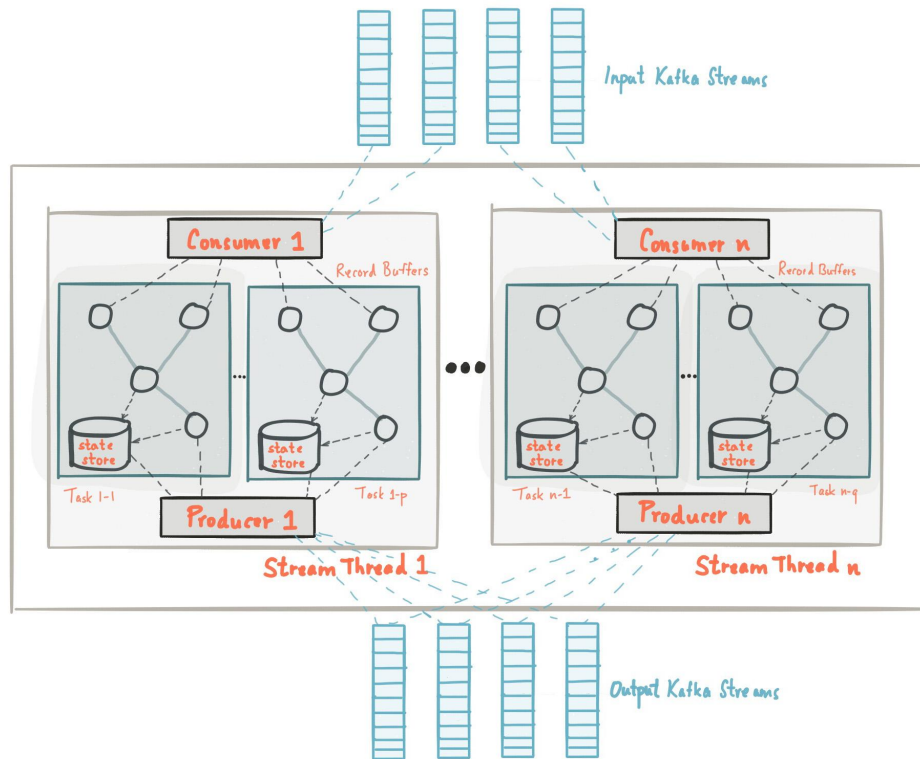
Configuring a Streams Application



```
import java.util.Properties;
import org.apache.kafka.streams.StreamsConfig;

Properties props = new Properties();
// Set a few key parameters
props.put(StreamsConfig.APPLICATION_ID_CONFIG, "my-first-streams-application");
props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
// Any further settings
props.put(... , ...);
```

Kafka Streams Architecture



Time To Code!

<https://github.com/vgallet/workshop-kafka-streams>



Thank you