

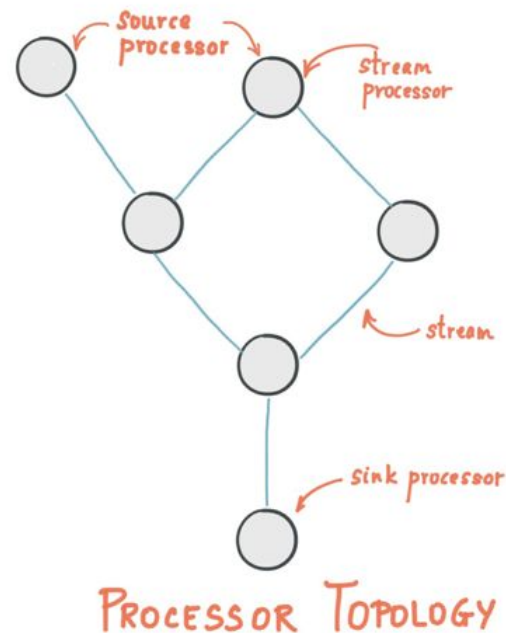
Kafka Streams

Index

- What is Kafka stream? (Recap)
- Joins
- Time concept
- Windowing
- Demo

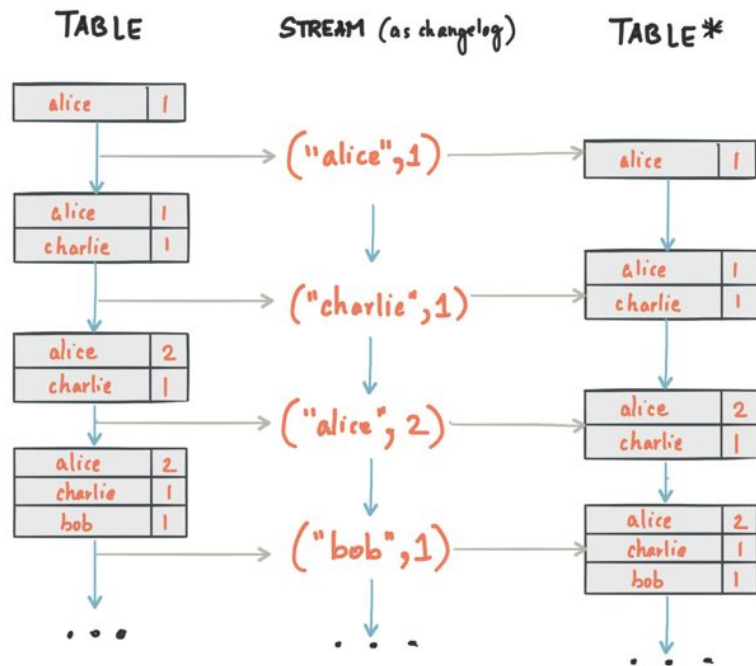
What is Kafka Stream? (Recap)

- Client library
- Processing and analyzing data stored in Kafka
- Builds upon important stream processing concepts:
 - event time/processing time,
 - windowing support
 - management of state



Source: <https://kafka.apache.org>, Dominic Presentation on Kafka Stream

What is Kafka Stream? (Recap)



Source: <https://kafka.apache.org>, Dominic Presentation on Kafka Stream

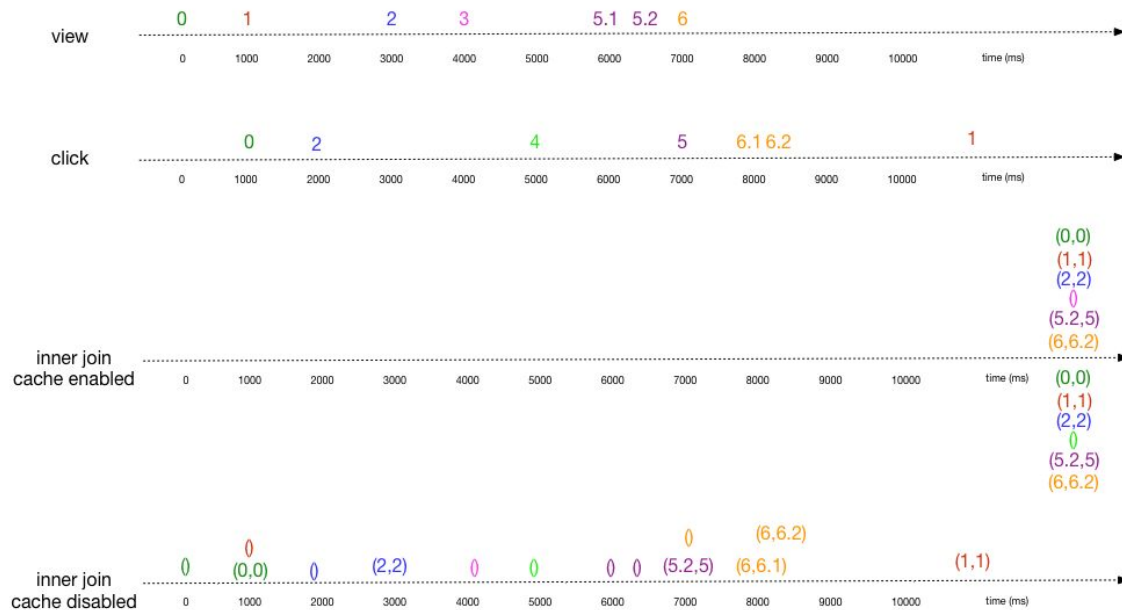
Kafka Joins

- Outer
- Inner
- Left

Kafka Stream/Table Joins

- Stream to Stream Join (always windowed)
- Table to Table Join (always not windowed)
- Stream to Table Join

Kafka Table-Table Outer Joins



Source: <https://blog.codecentric.de/en/2017/02/crossing-streams-joins-apache-kafka/>

Time-Kafka

Every event has an associated notion of time.

- Event time (extending TimestampExtractor)
- Processing time
- Ingestion time

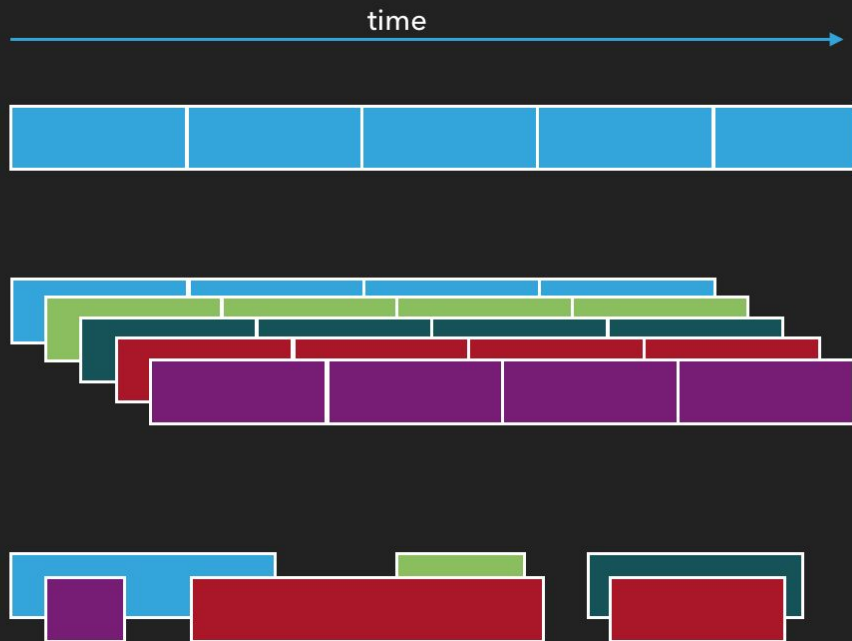
Kafka-Windowing

- ▶ Time-based

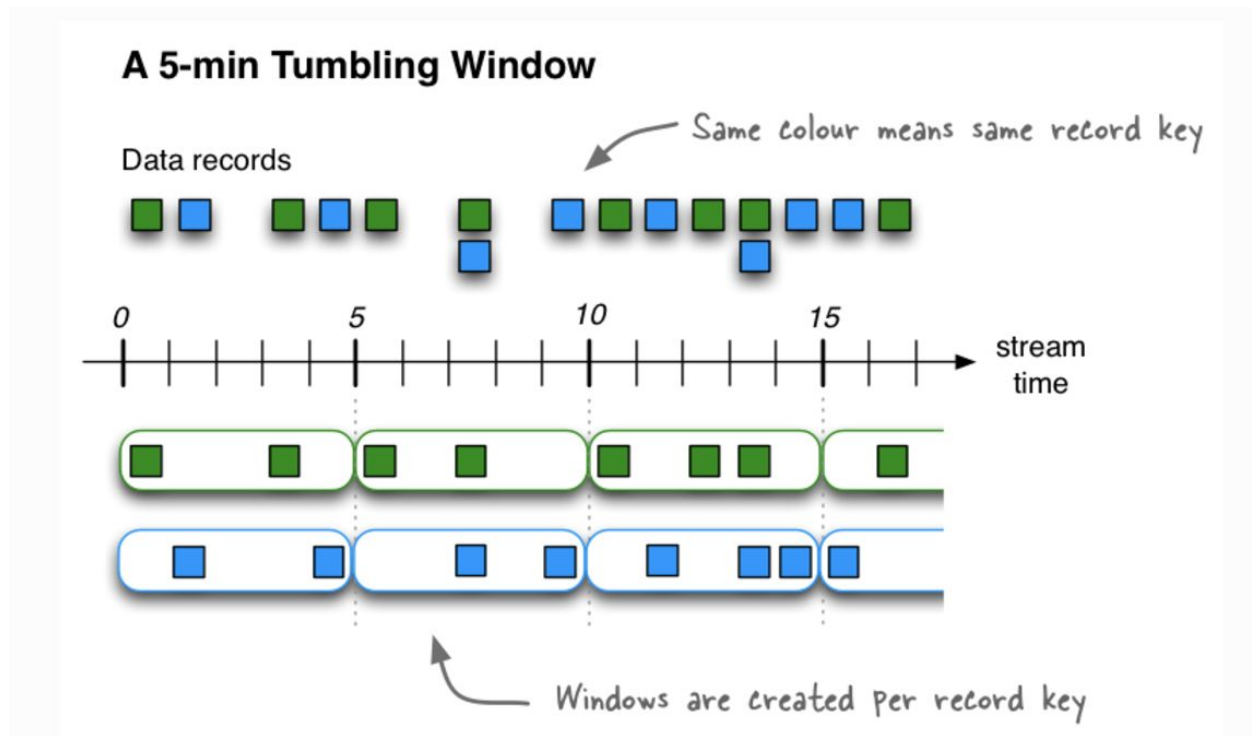
- ▶ fixed/tumbling

- ▶ sliding

- ▶ Session-based

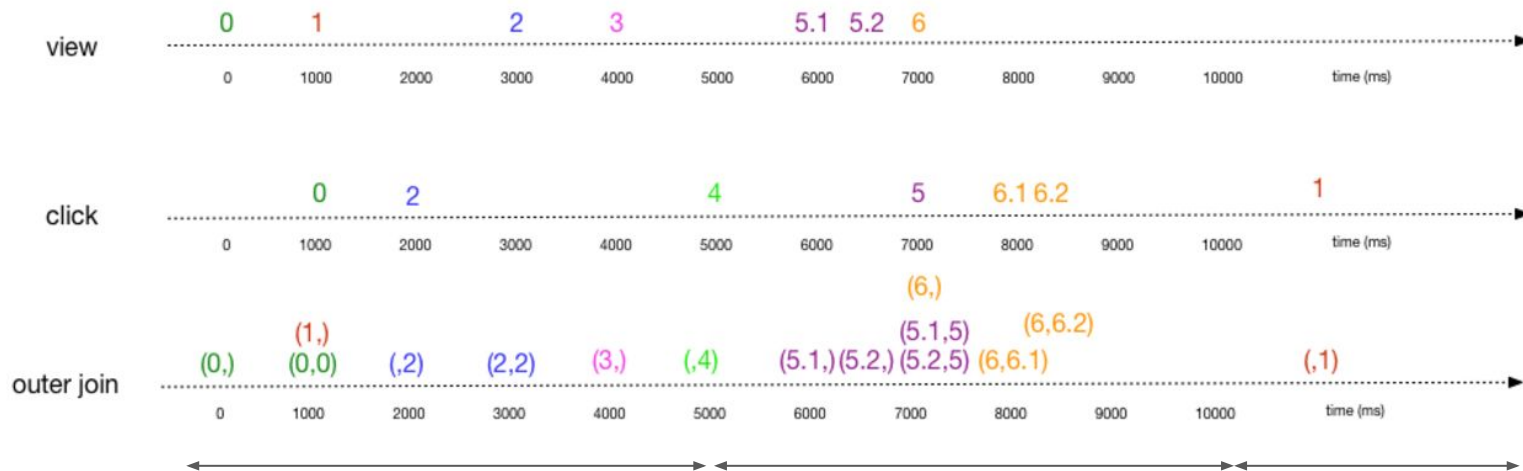


Tumbling window-Grouping



Source: <https://kafka.apache.org>,

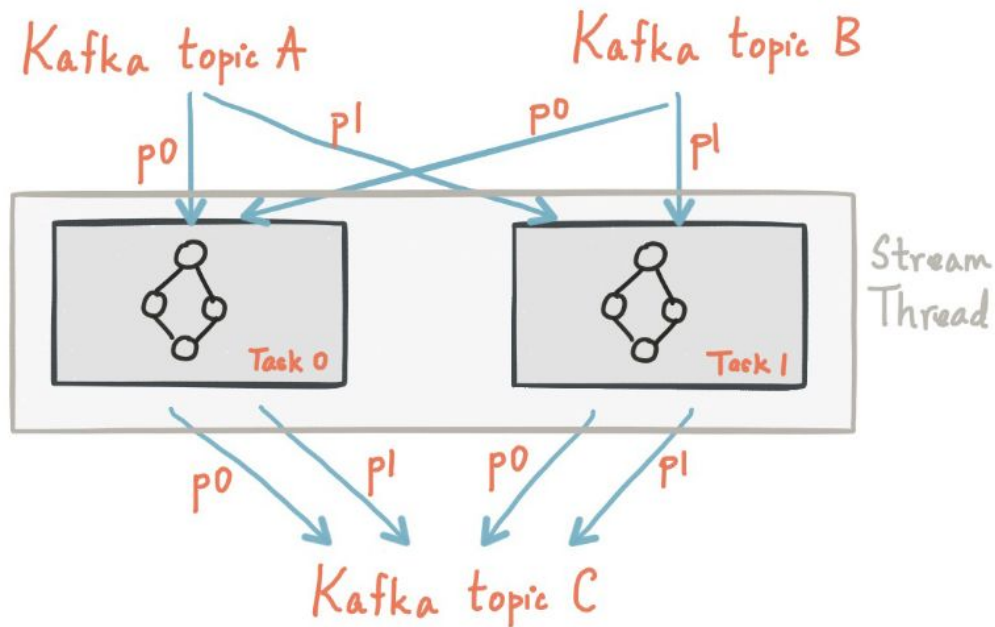
Outer Join Stream-Stream (Windowed over 5000 ms)



Source: <https://blog.codecentric.de/en/2017/02/crossing-streams-joins-apache-kafka/>

Kafka stream-Features

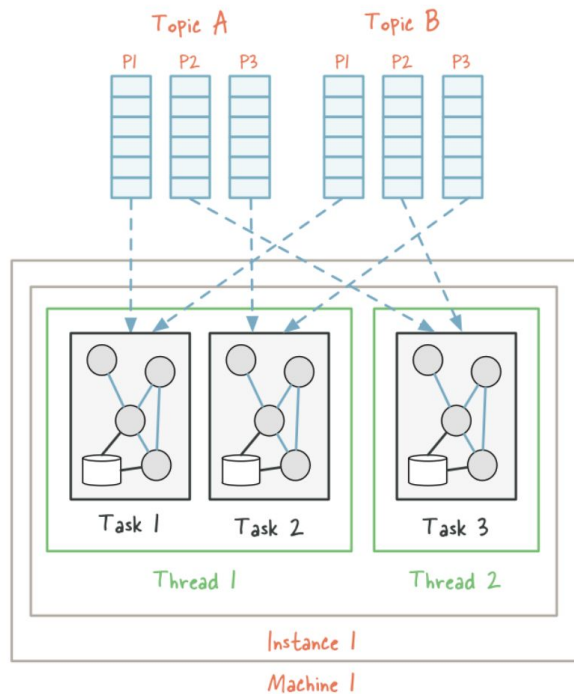
Stream tasks



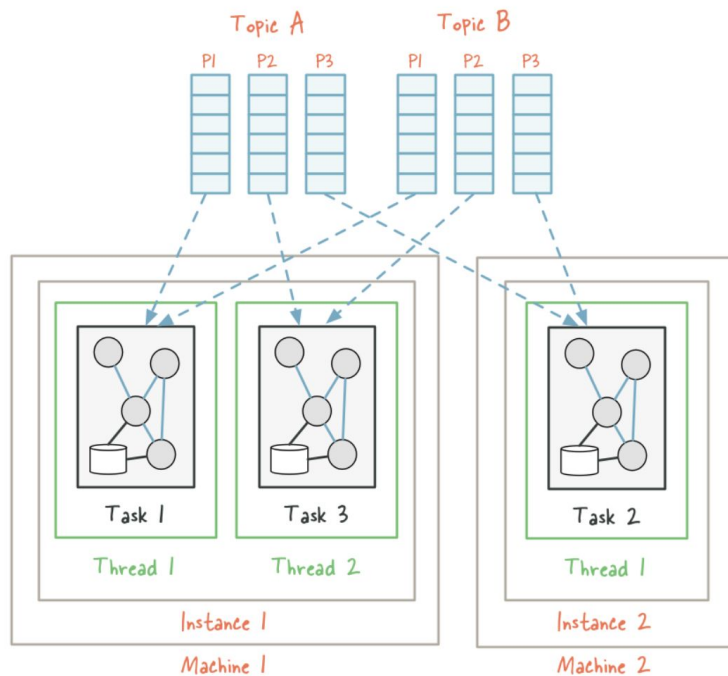
One stream thread running two stream tasks.

Threading model

- Configure number of threads
- Allows parallelizing in a single instance
- No state shared within threads
- Same as having multiple instance
- Scalability handled by Kafka cluster



Threading model



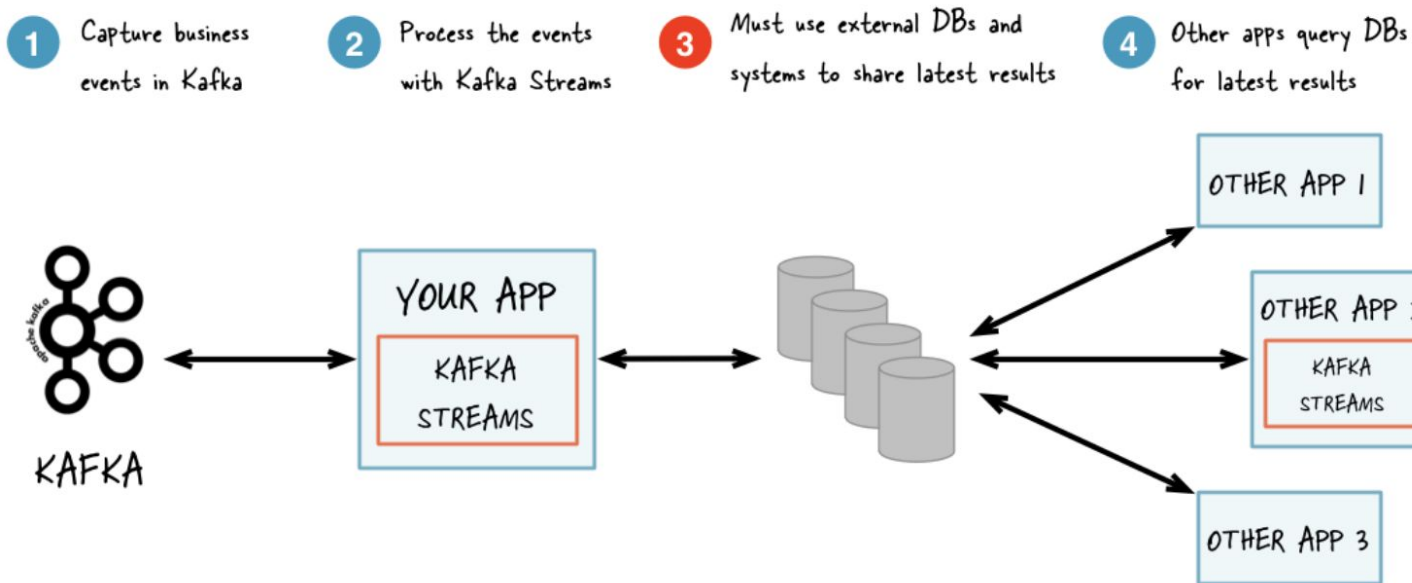
Joins

- The join topics should have same partition count
- Joins based on keys
- Partitions are assigned to instances
- Realtime joins => Key are distributed across partition via
 - $\text{Hash}(\text{Key}) \% \text{partition_count}$

Global KTable

- Act as a broadcast variable
- Global KTable: Data in all the partitions is available to all instances
- Benefits:
 - More convenient and effective joins
 - No need to co-partition data
- Drawbacks:
 - Increase local storage
 - Increased network load
- Ideally, should be done for less data size

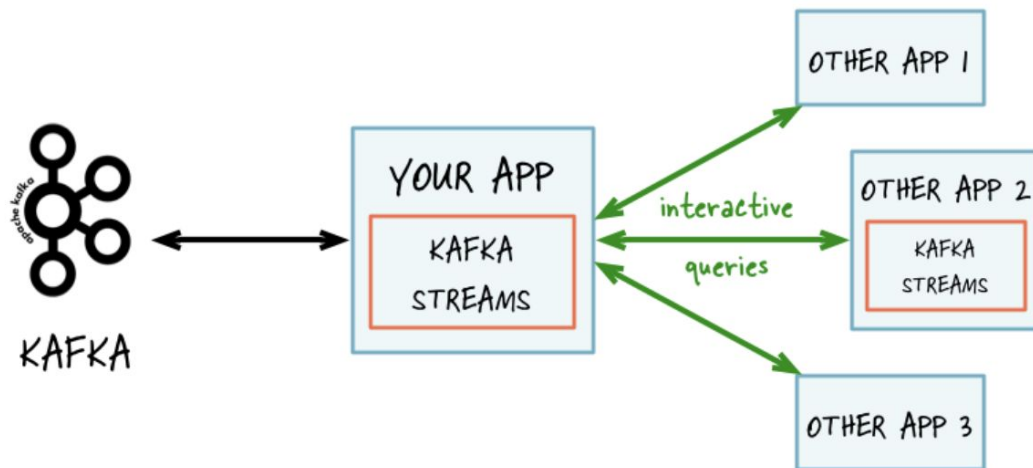
Interactive queries



Without Interactive Queries: increased complexity and heavier footprint of architecture.

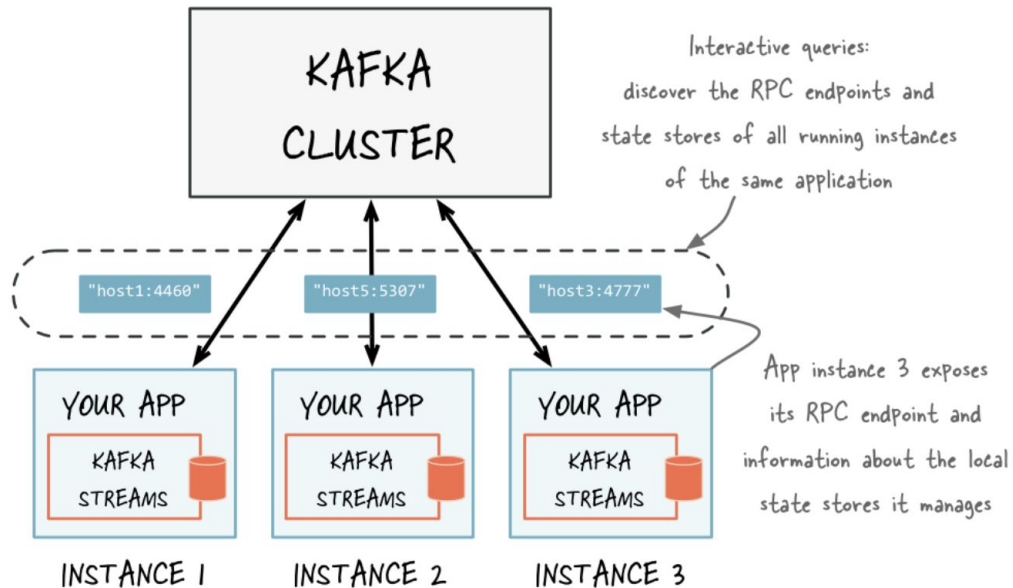
Interactive queries

- 1 Capture business events in Kafka
- 2 Process the events with Kafka Streams
- 3 With interactive queries, other apps can directly query the latest results



With Interactive Queries: simplified, more application-centric architecture.

Interactive queries-RPC



*Discover any running instances of the same application as well as the respective RPC endpoints they expose for
Interactive Queries*

Interactive queries-RPC

- `KafkaStreams#allMetadata()`
- `KafkaStreams#allMetadataForStore(String storeName)`
- `KafkaStreams#metadataForKey(String storeName, K key, Serializer<K> keySerializer)`
- `KafkaStreams#metadataForKey(String storeName, K key, StreamPartitioner<K, ?> partitioner)`

Processing guarantees

- At-Least once
 - `processing.guarantee="at_least_once"`
- Exactly once
 - `processing.guarantee="exactly_once_v2"`