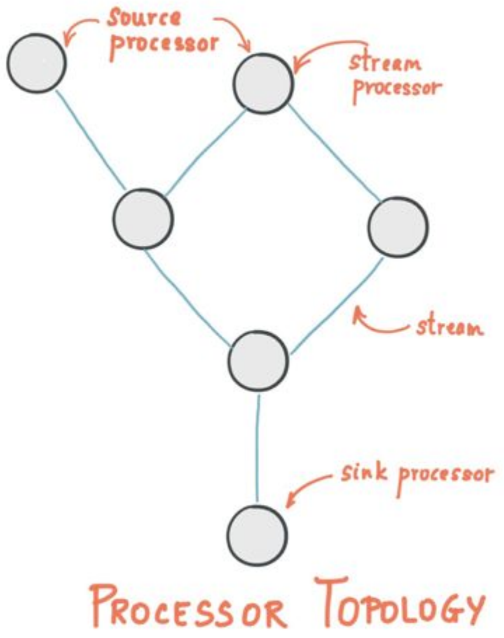
Kafka Streams

Index

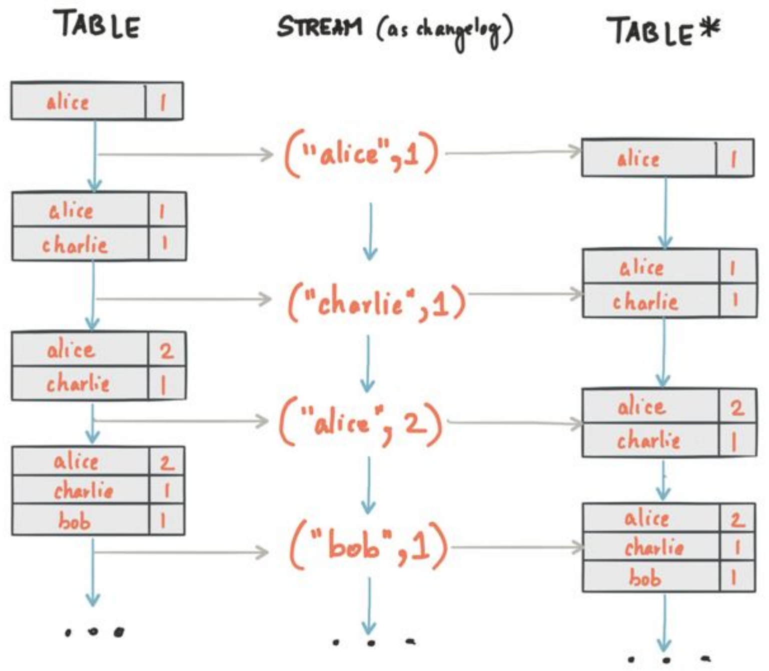
What is Kafka Stream? (Recap)

- 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

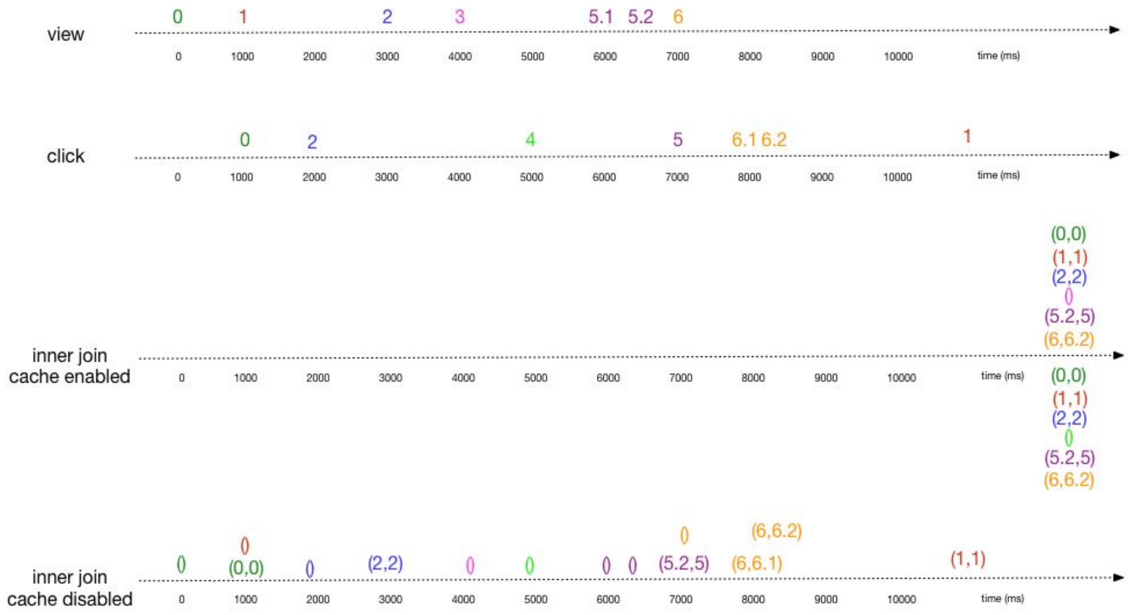
Kafka Joins

- 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



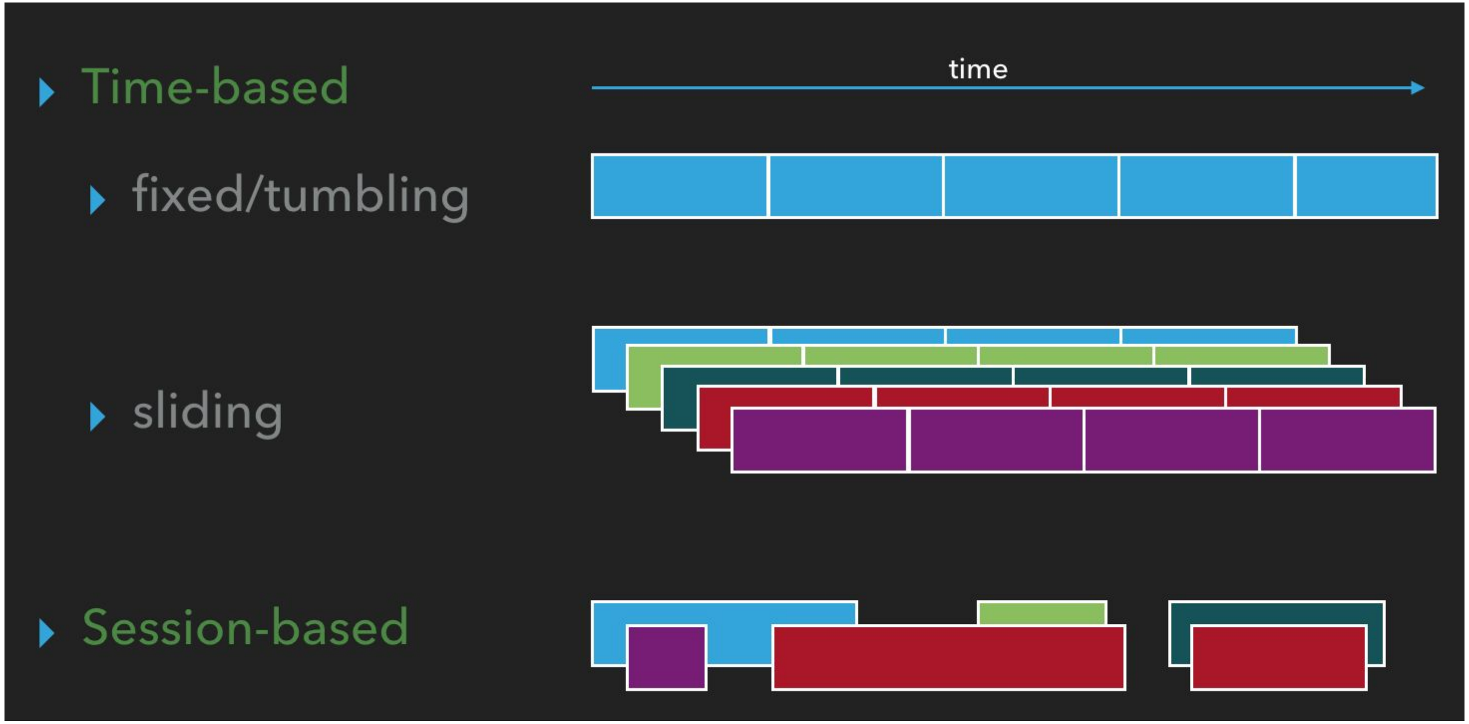
Source: https://blog.codecentric.de/en/2017/02/crossing-streams-joins-apache-kafka/

Time-Kafka

Every event has an associated notion of time.

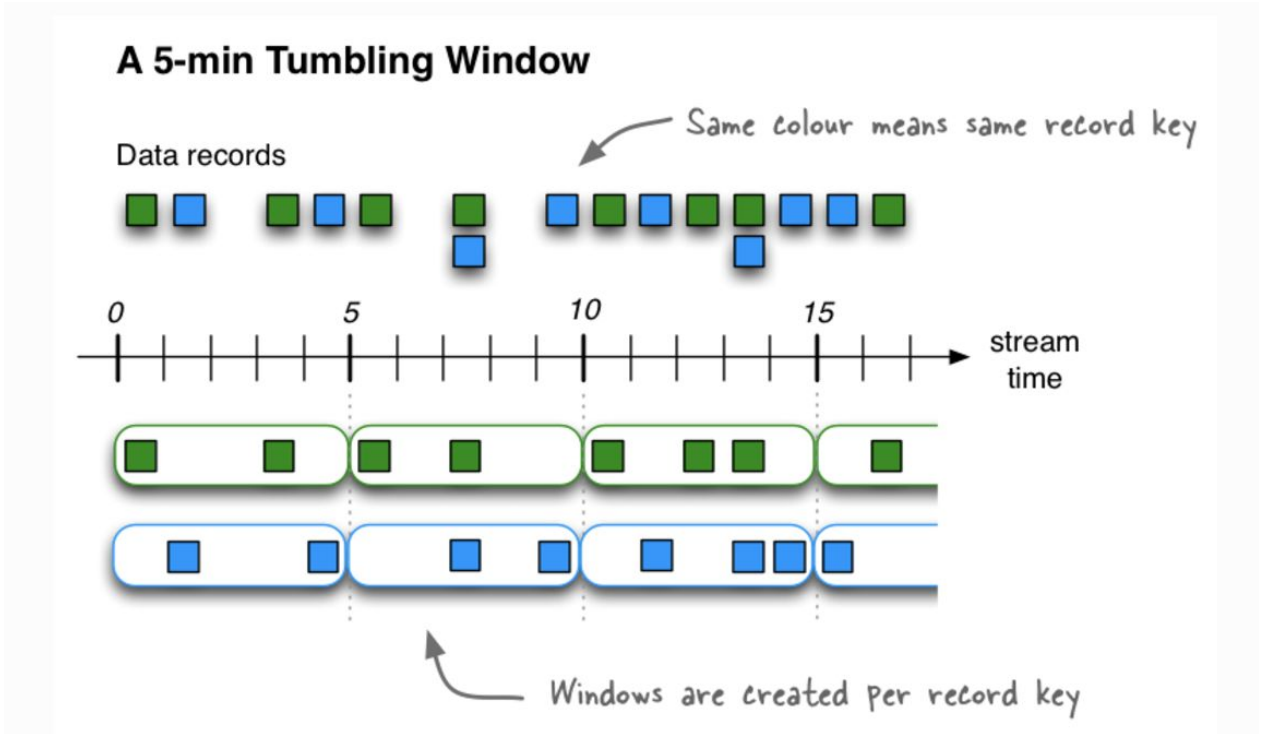
- Event time (extending TimestampExtractor)
- Processing time
- Ingestion time

Kafka-Windowing



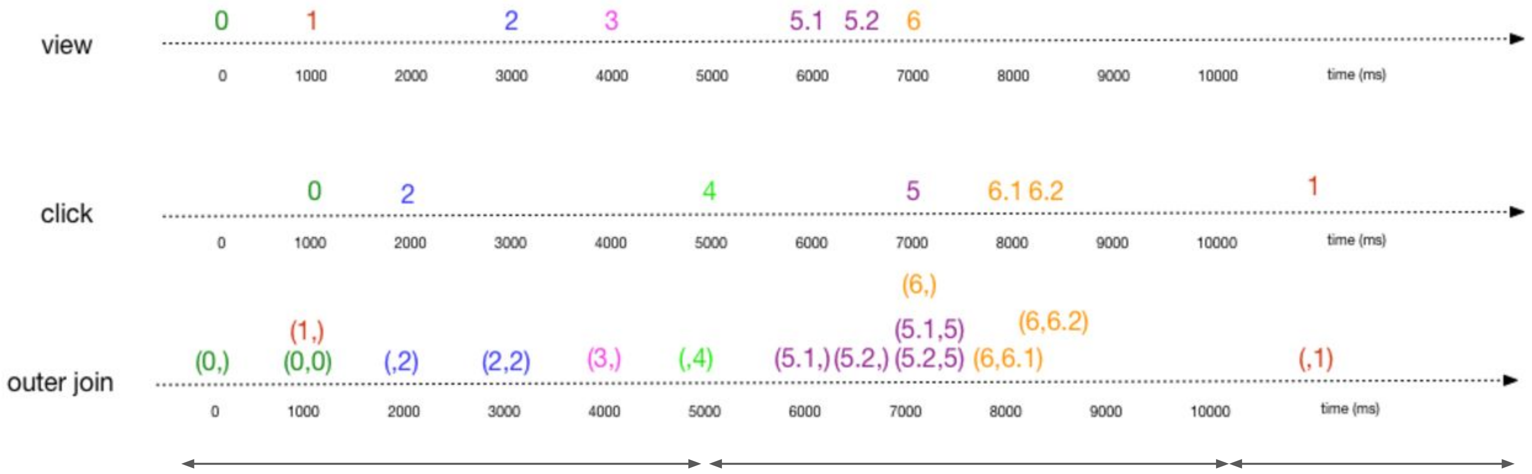
Source: https://softwaremill.com/windowing-in-big-data-streams-spark-flink-kafka-akka/

Tumbling window-Grouping


A 5-min Tumbling Window
Data records
Same colour means same record key
0
5
10
15
stream time
Windows are created per record key

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

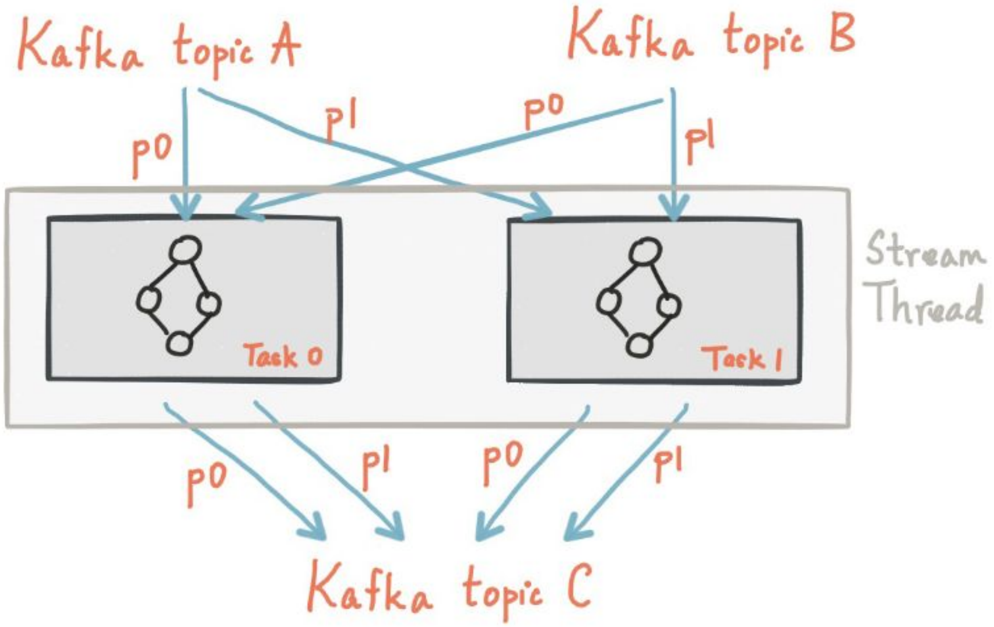
Outer Join Stream-Stream (Windowed over 5000 ms)



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

Kafka stream-Features

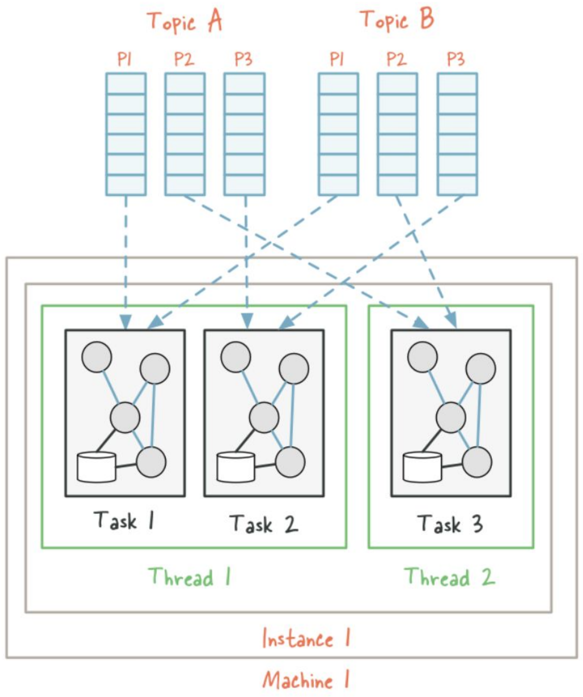
Stream tasks



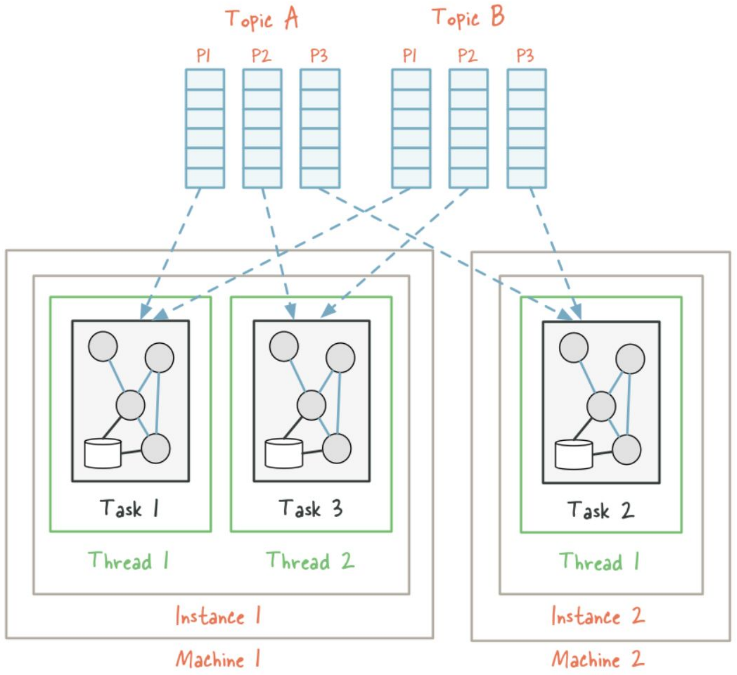
One stream thread running two stream tasks.

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



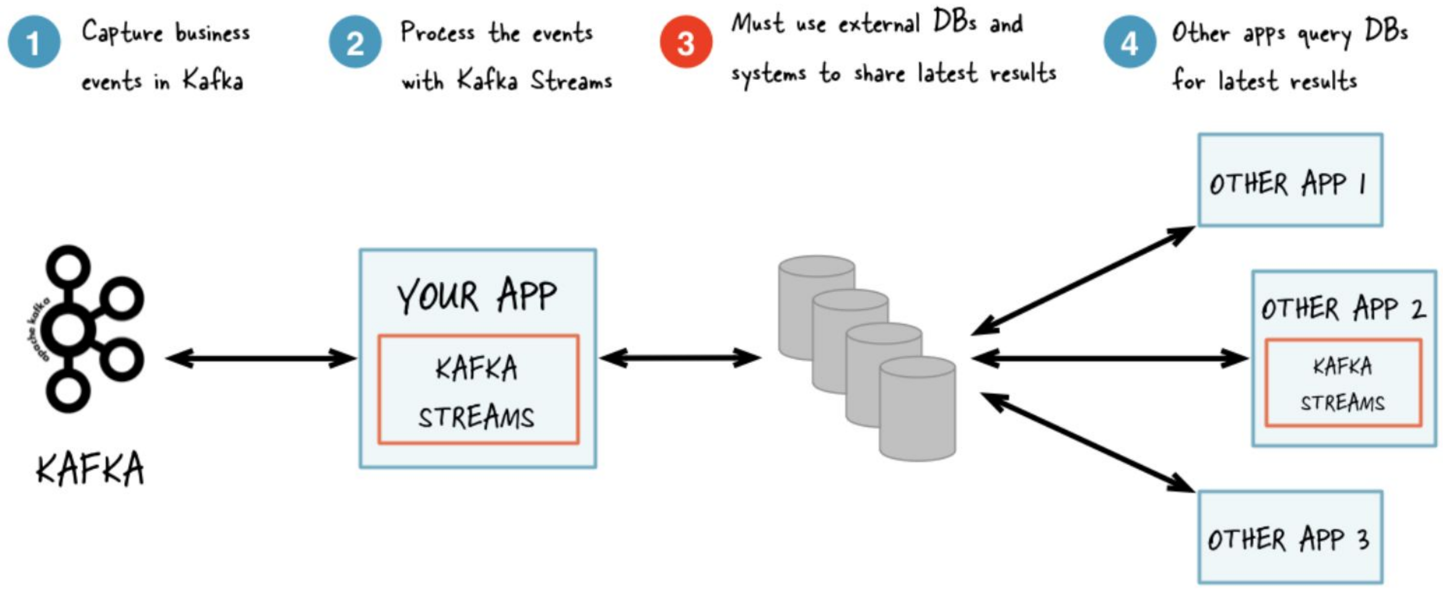
Joins

- 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
    - Hash(Key)%partition_count

Global KTable

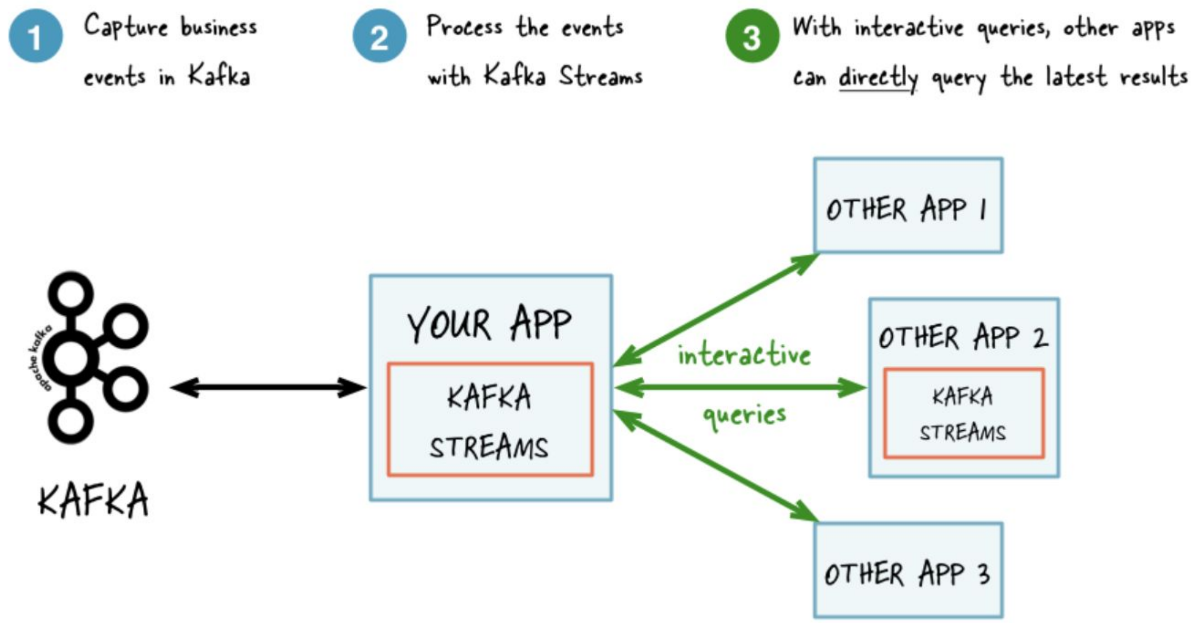
- Act as a broadcast variable
- Global KTable: Data in all the partitions in available to all instances
- Benefits:
    - More convenient and effective joins
    - No need to co-partition data
- Drawbacks:
    - Increase local storage
    - Increased network load
- Ideally, should be done for less data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

Interactive queries


1 Capture business events in Kafka
2 Process the events with Kafka Streams
3 Must use external DBs and systems to share latest results
4 Other apps query DBs for latest results
apache kafka
KAFKA
YOUR APP
KAFKA STREAMS
OTHER APP 1
OTHER APP 2
KAFKA STREAMS
OTHER APP 3

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

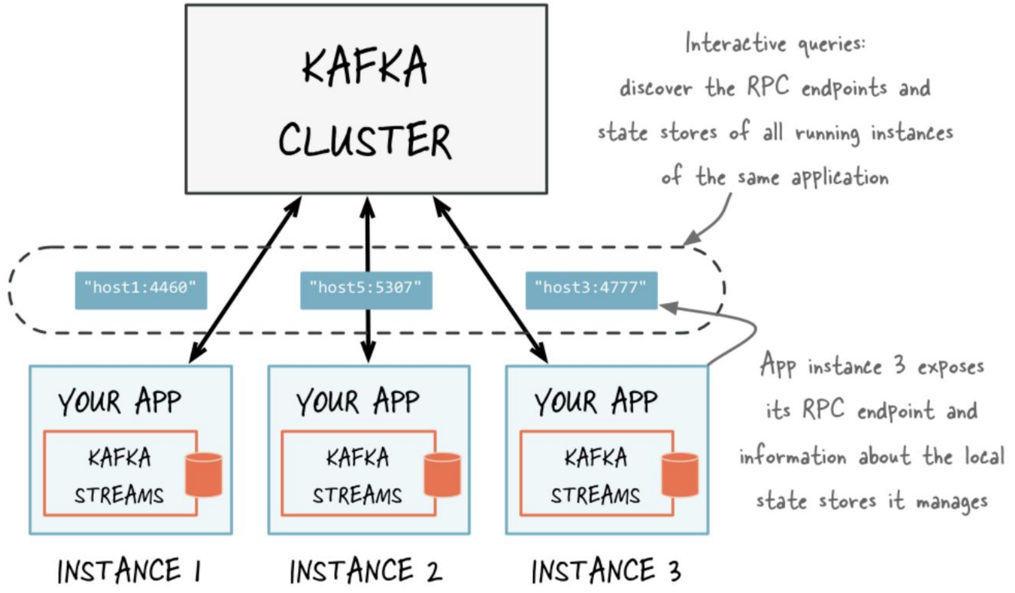
Interactive queries



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
apache kafka
KAFKA
YOUR APP
KAFKA STREAMS
interactive queries
OTHER APP 1
OTHER APP 2
KAFKA STREAMS
OTHER APP 3

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

Interactive queries-RPC


KAFKA CLUSTER
Interactive queries: discover the RPC endpoints and state stores of all running instances of the same application
"host1:4460"
"host5:5307"
"host3:4777"
YOUR APP
KAFKA STREAMS
YOUR APP
KAFKA STREAMS
YOUR APP
KAFKA STREAMS
App instance 3 exposes its RPC endpoint and information about the local state stores it manages
INSTANCE 1
INSTANCE 2
INSTANCE 3

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"