What are Streams?

So far, we have discussed using Kafka for messaging (reliably passing data between two applications).

Kafka Streams allows us to build applications that process Kafka data in real-time with ease.

A Kafka Streams application is an application where both the input and the output are stored in Kafka topics.

Kafka Streams is a client library (API) that makes it easy to build these applications.

Kafka Streams Transformations

Kafka Streams provides a robust set of tools for processing and transforming data. The Kafka cluster itself serves as the backend for data management and storage.

There are two types of data transformations in Kafka Streams:

- Stateless transformations do not require any additional storage to manage the state.
- Stateful transformations require a state store to manage the state.

Stateless Transformations

Branch

Splits a stream into multiple streams based on a predicate.

Filter

Removes messages from the stream based on a condition.

FlatMap

Takes input records and turns them into a different set of records.

Foreach

Performs an arbitrary stateless operation on each record. This is a terminal operation and stops further processing.

Stateless Transformations

Branch

Splits a stream into multiple streams based on a predicate.

Filter

Removes messages from the stream based on a condition.

FlatMap

Takes input records and turns them into a different set of records.

Foreach

Performs an arbitrary stateless operation on each record. This is a terminal operation and stops further processing.

Stateless Transformations

GroupBy/GroupByKey

Groups records by their key. This is required to perform stateful transformations.

Map

Allows you to read a record and produce a new, modified record.

Merge

Merges two streams into one stream.

Peek

Similar to `Foreach`, but does not stop processing.

Kafka Streams Aggregations

Stateless transformations, such as `groupByKey` and `groupBy` can be used to group records that share the same key.

Aggregations are stateful transformations that always operate on these groups of records sharing the same key.

Aggregations

Aggregate

Generates a new record from a calculation involving the grouped records.

Count

Counts the number of records for each grouped key.

Reduce

Combines the grouped records into a single record.

Kafka Streams Joins

Inner Join

The new stream will contain only records that have a match in both joined streams.

Left Join

The new stream will contain all records from the first stream, but only matching records from the joined stream.

Outer Join

The new stream will contain all records from both streams.

Streams vs. Tables

Here are some example use cases:

Stream:

- Credit card transactions in real time.
- A real-time log of attendees checking in to a conference.
- A log of customer purchases which represent the removal of items from a store's inventory.

Table:

- A user's current available credit card balance.
- A list of conference attendee names with a value indicating whether or not they have checked in.
- A set of data containing the quantity of each item in a store's inventory.

Streams vs. Tables

Kafka Streams models data in two primary ways: streams and tables.

Streams: Each record is a self-contained piece of data in an unbounded set of data. New records do not replace an existing piece of data with a new value.

Tables: Records represent a current state that can be overwritten/ updated.