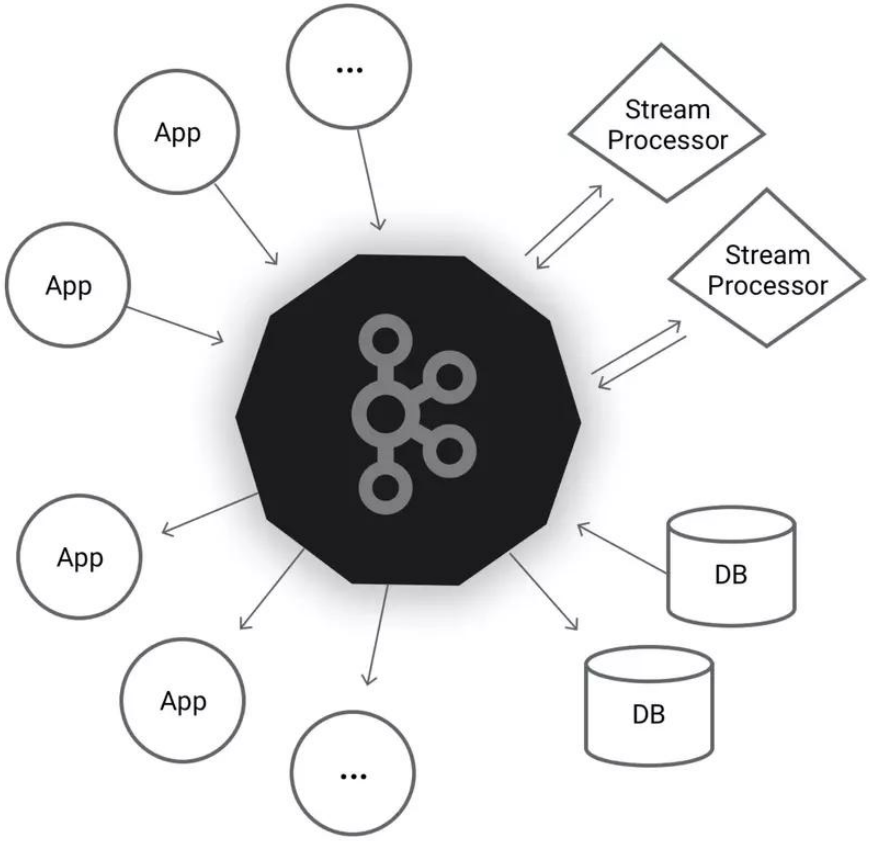
confluent

Kafka Streams: What it is, and how to use it?

Matthias J. Sax | Software Engineer | @MatthiasJSax
Apache Kafka PMC member

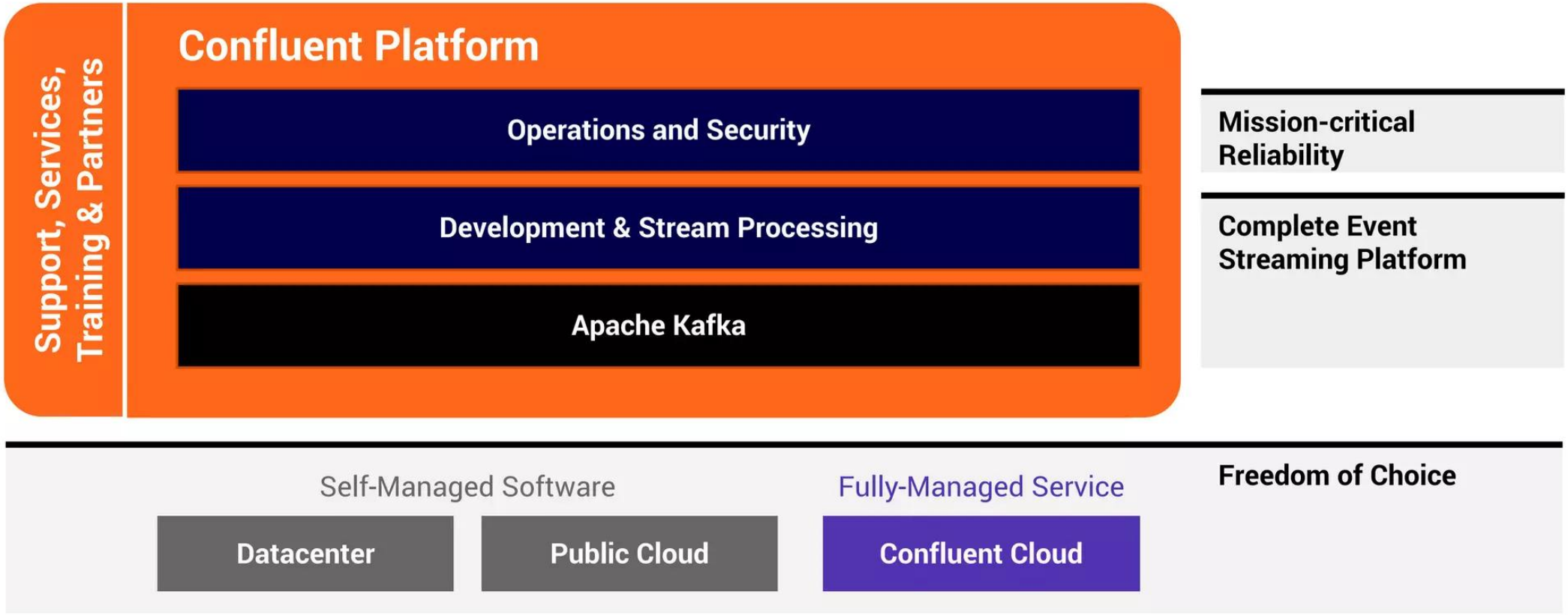

apache kafka

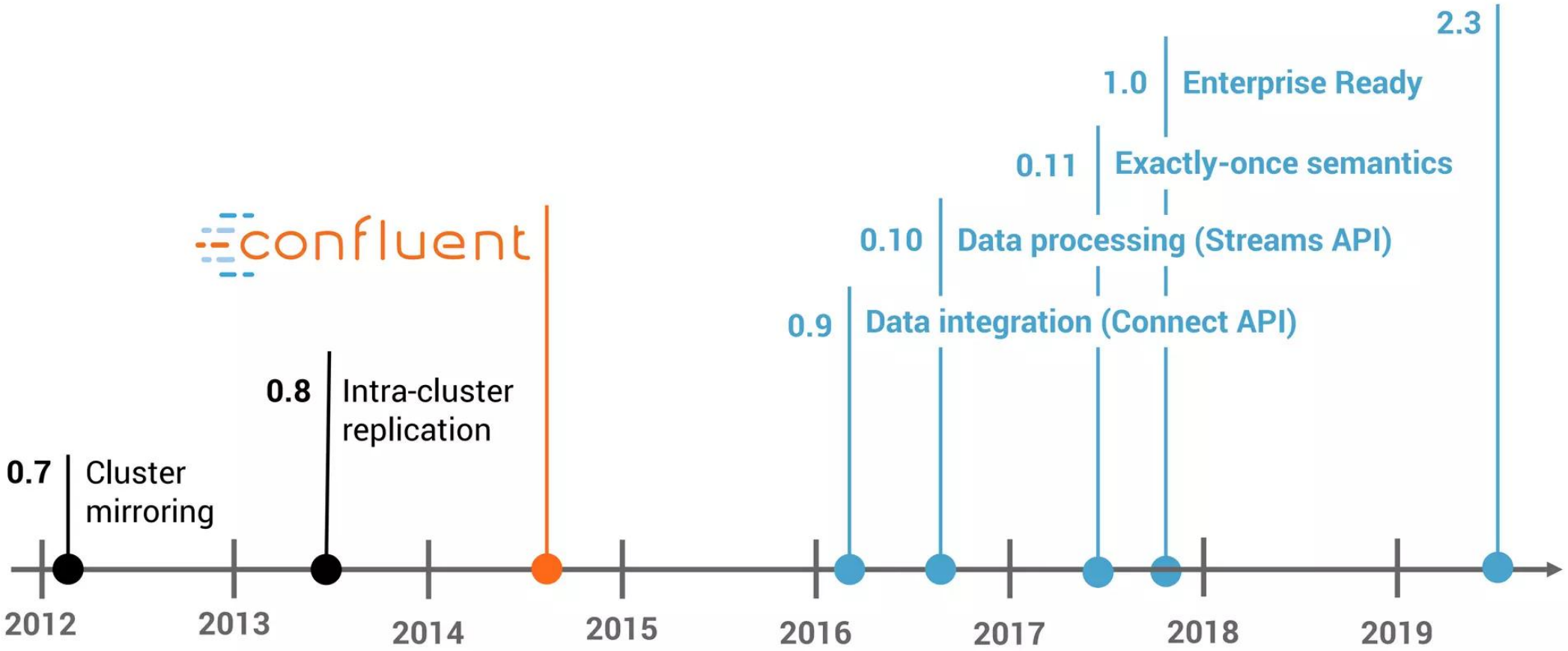
Apache Kafka®: A Distributed Streaming Platform



Confluent Platform

The Event Streaming Platform Built by the Original Creators of Apache Kafka®




2.3
1.0 Enterprise Ready
0.11 Exactly-once semantics
confluent
0.10 Data processing (Streams API)
0.9 Data integration (Connect API)
0.8 Intra-cluster replication
0.7 Cluster mirroring
2012
2013
2014
2015
2016
2017
2018
2019

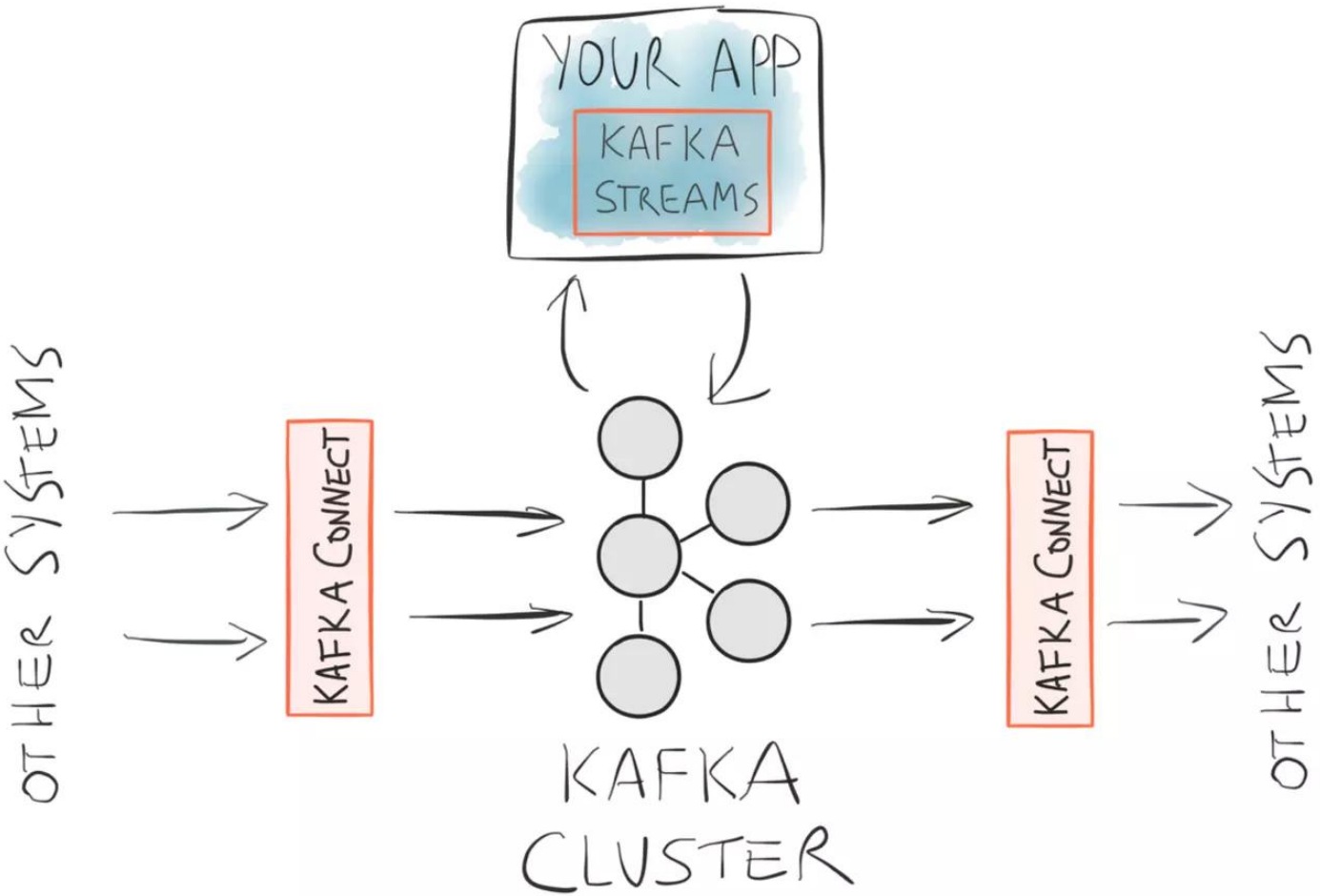
Kafka Streams

(aka Streams API)

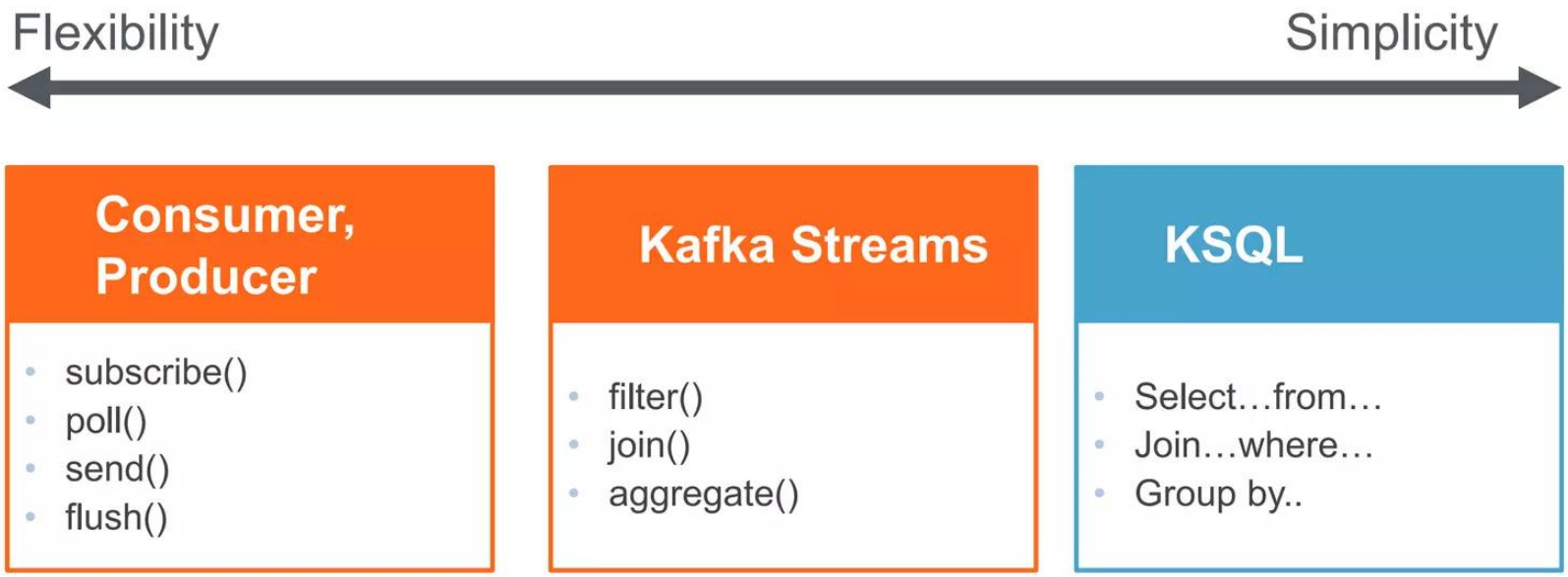
confluent

Streams API

- Java client library for building distributed apps
- Part of the Apache Kafka project
- Read-process-write pattern
- State handling, fault-tolerance, scaling


YOUR APP
KAFKA STREAMS
OTHER SYSTEMS
KAFKA CONNECT
KAFKA CLUSTER
KAFKA CONNECT
OTHER SYSTEMS

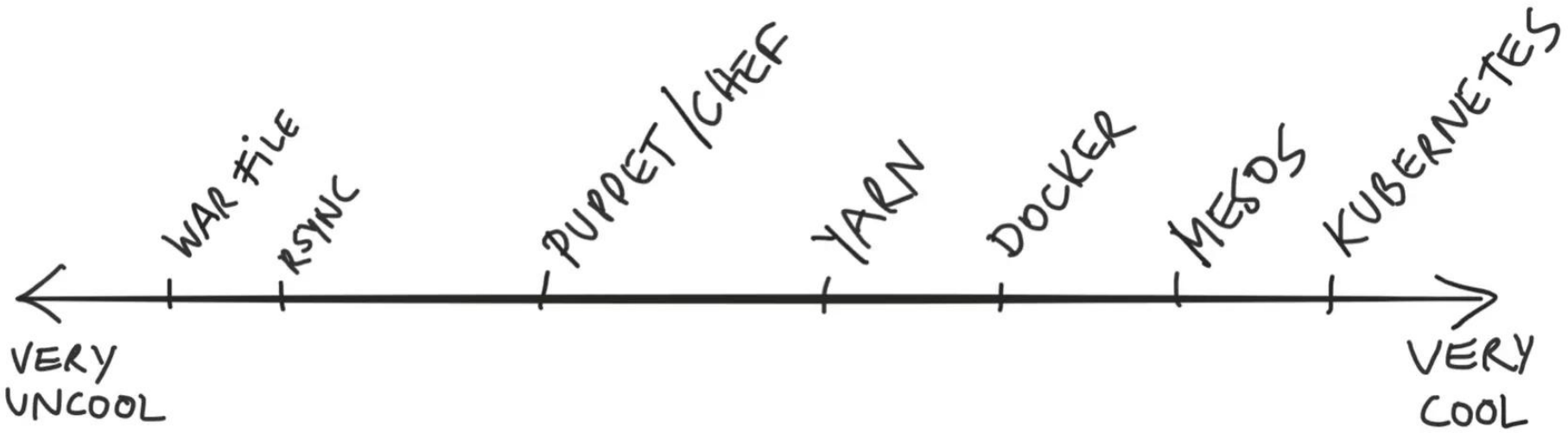
Why should I use it?



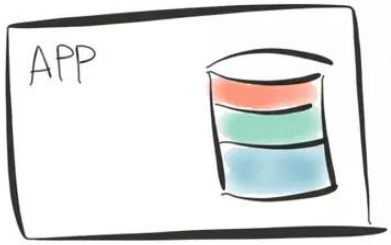
How can I use it?

```
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-streams</artifactId>
  <version>2.3.0</version>
</dependency>
```

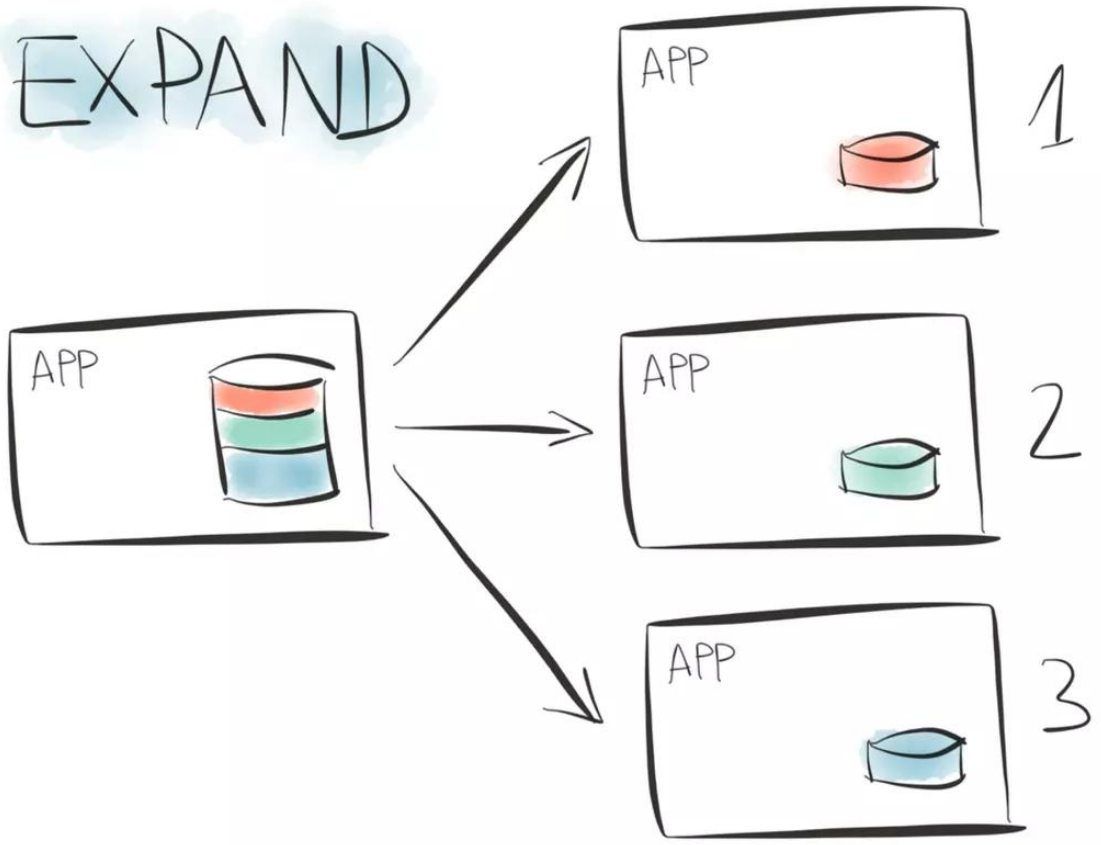
How do I deploy my application?



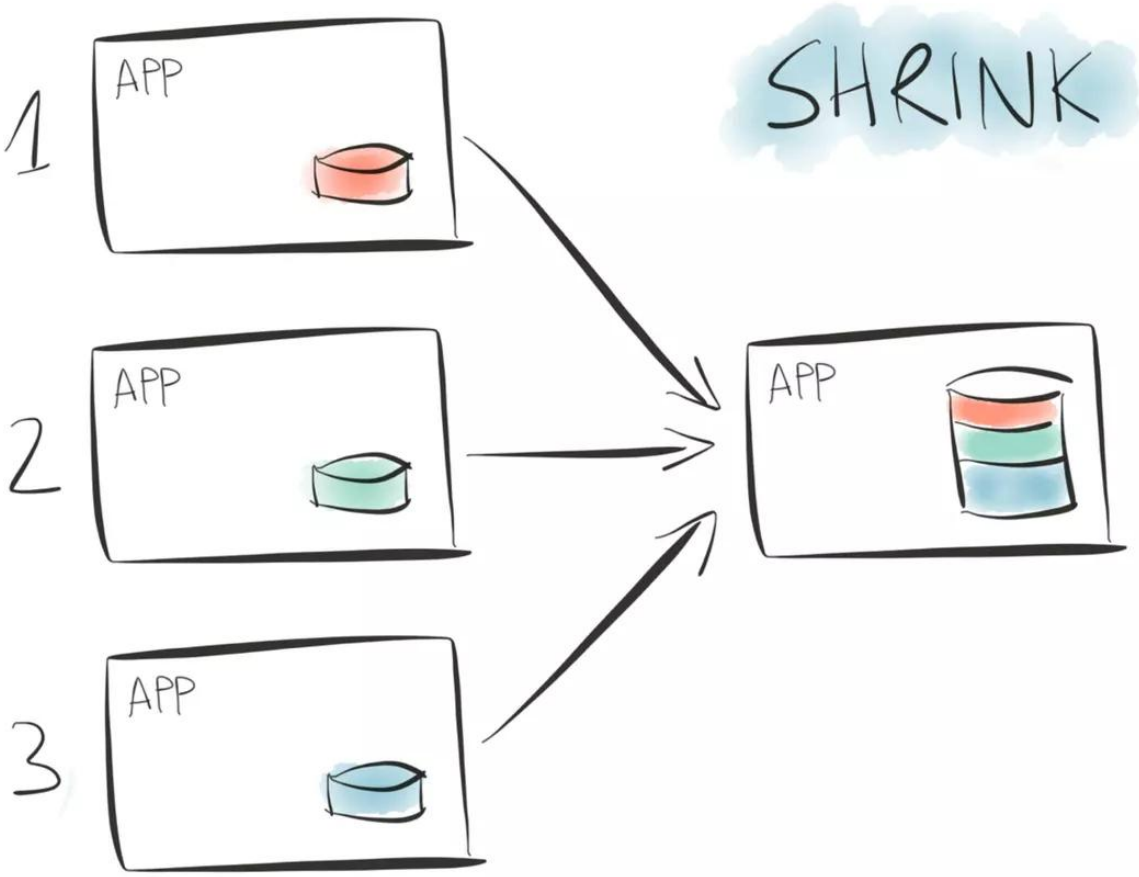
Scaling Your Application



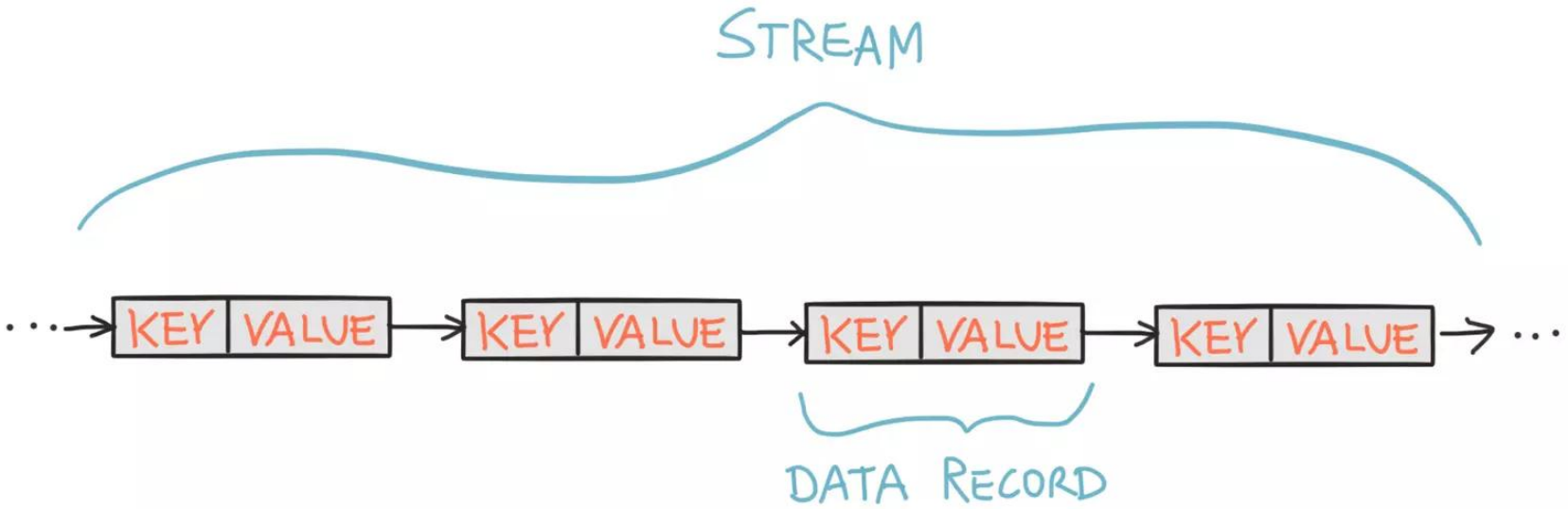
Scaling Your Application



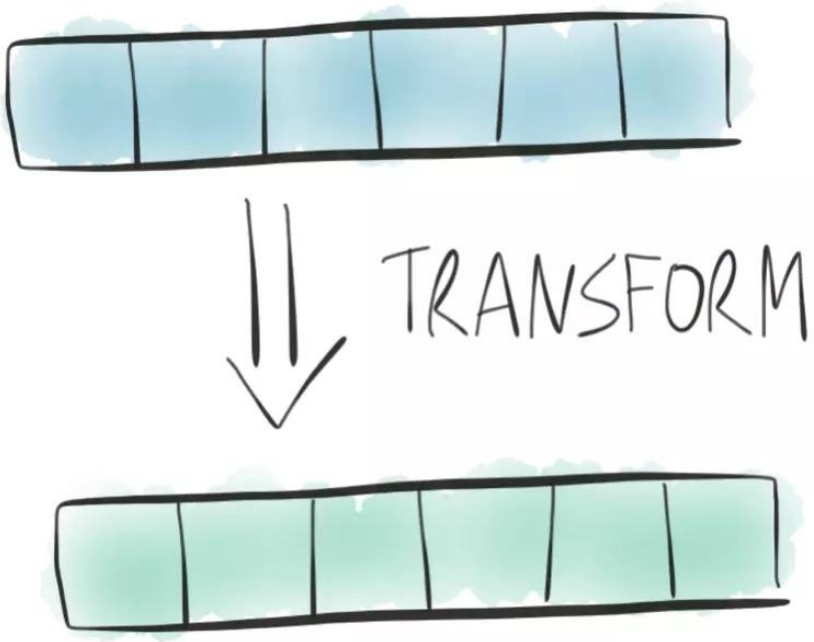
Scaling Your Application



Data Model



Transformations (map, filter, etc.)



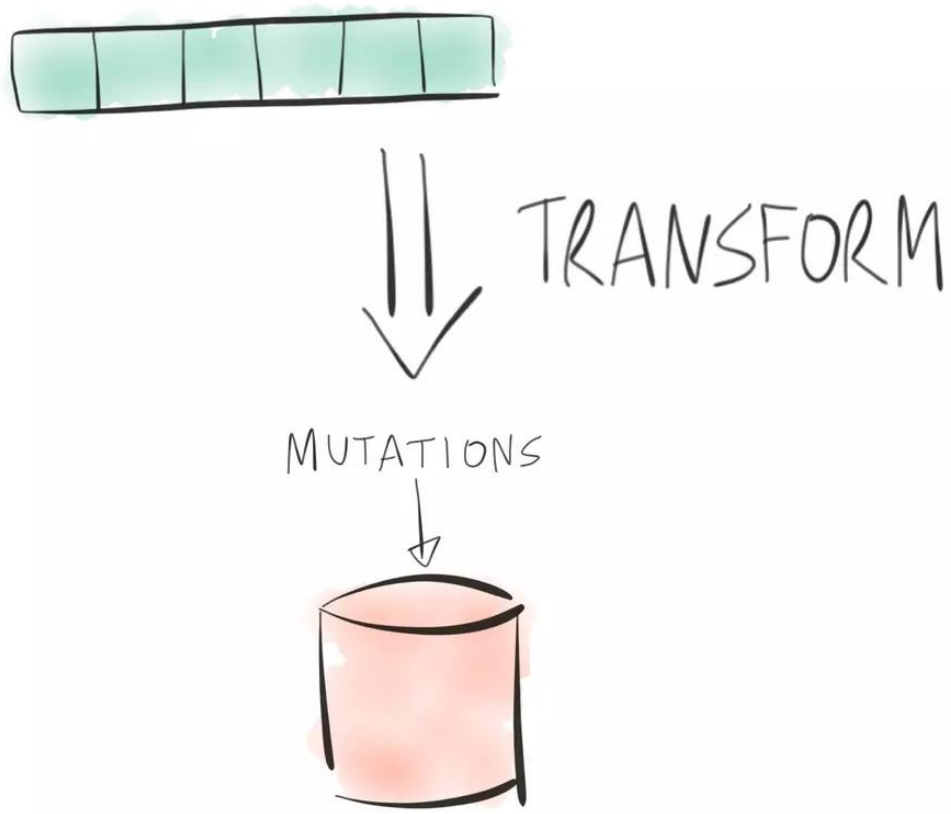

```
StreamsBuilder builder = new StreamsBuilder();
```

```
KStream<Long,String> inputStream =
    builder.stream("input-topic");
```

```
KStream<Long,String> outputStream =
    inputStream.mapValues(
        value -> value.toLowerCase());
```

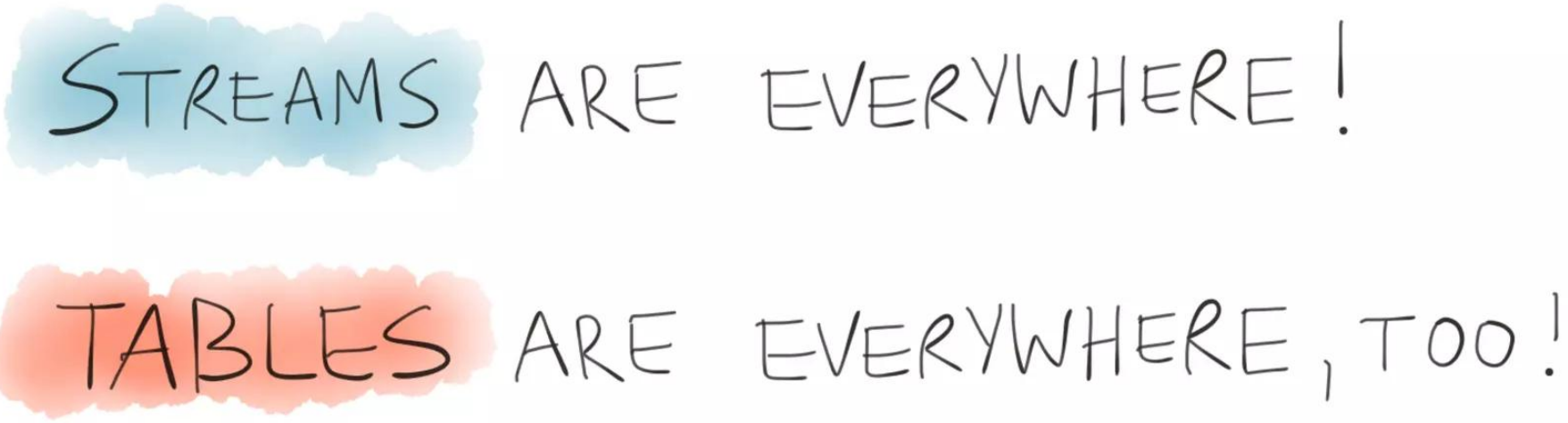
```
outputStream.to("output-topic");
```

Stream Aggregations




```
KTable<Long,Long> countPerKey
    = stream.groupByKey().count();
```

Data Model (cont.)



confluent

Streams

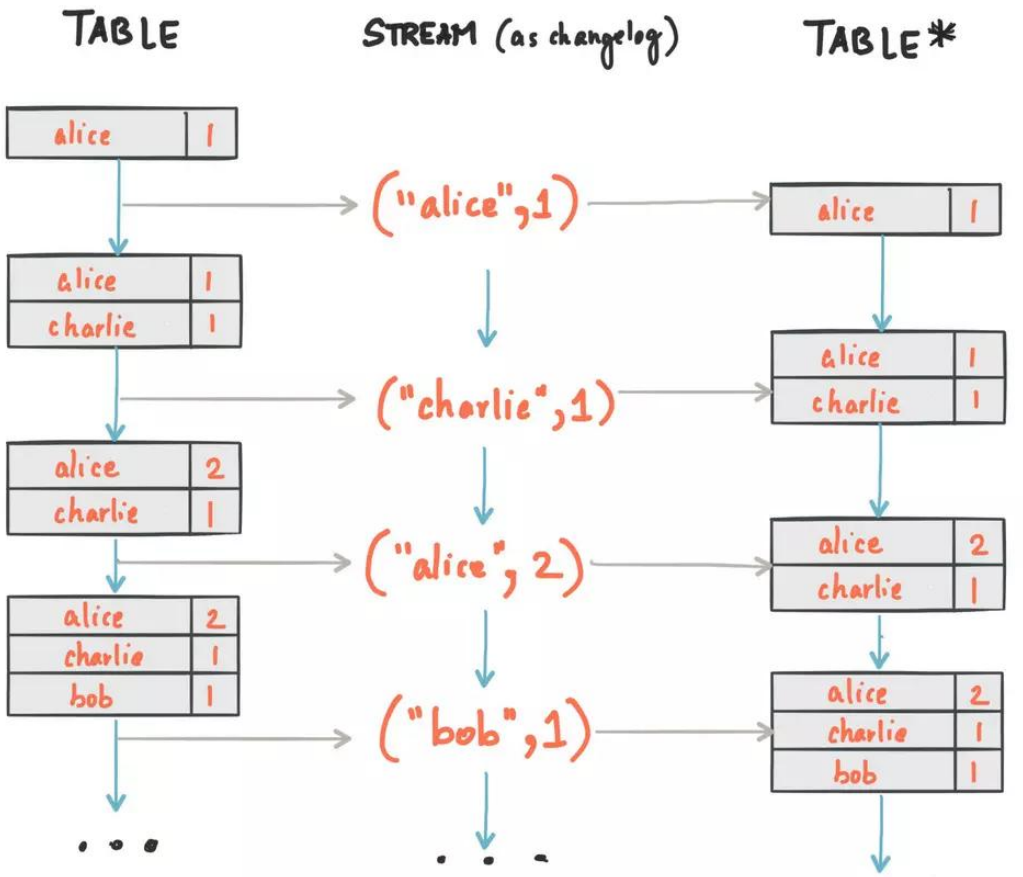
- **Facts/events**
- **Infinite/unbounded**
- **Append-only**
- **Immutable**

confluent

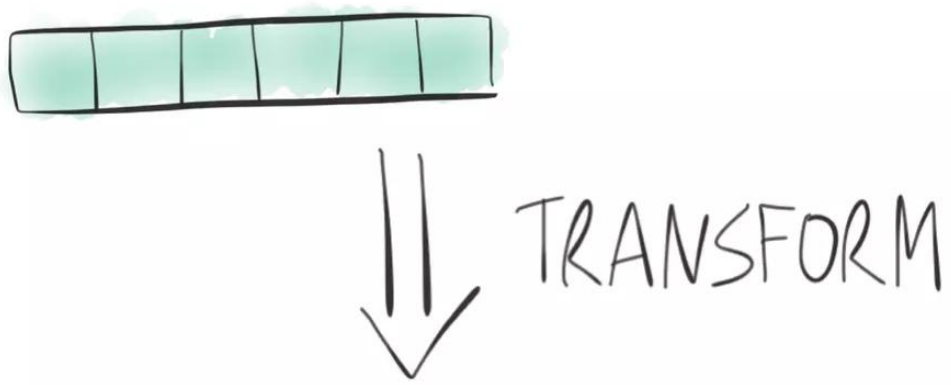
Tables

- Current state
- Finite/Bounded
- Insert, Update, Delete
- Mutable

Stream-Table Duality

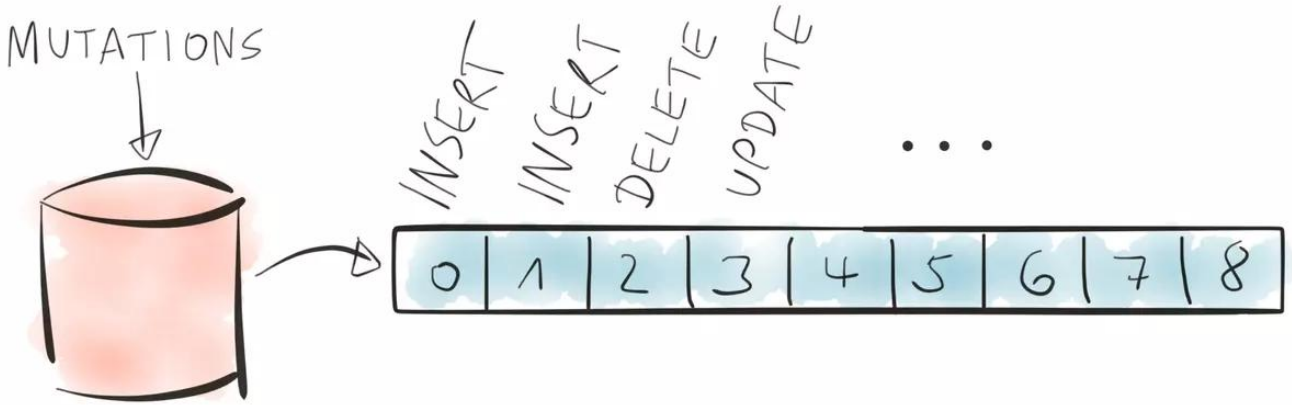


Stream Aggregations: The Changelog



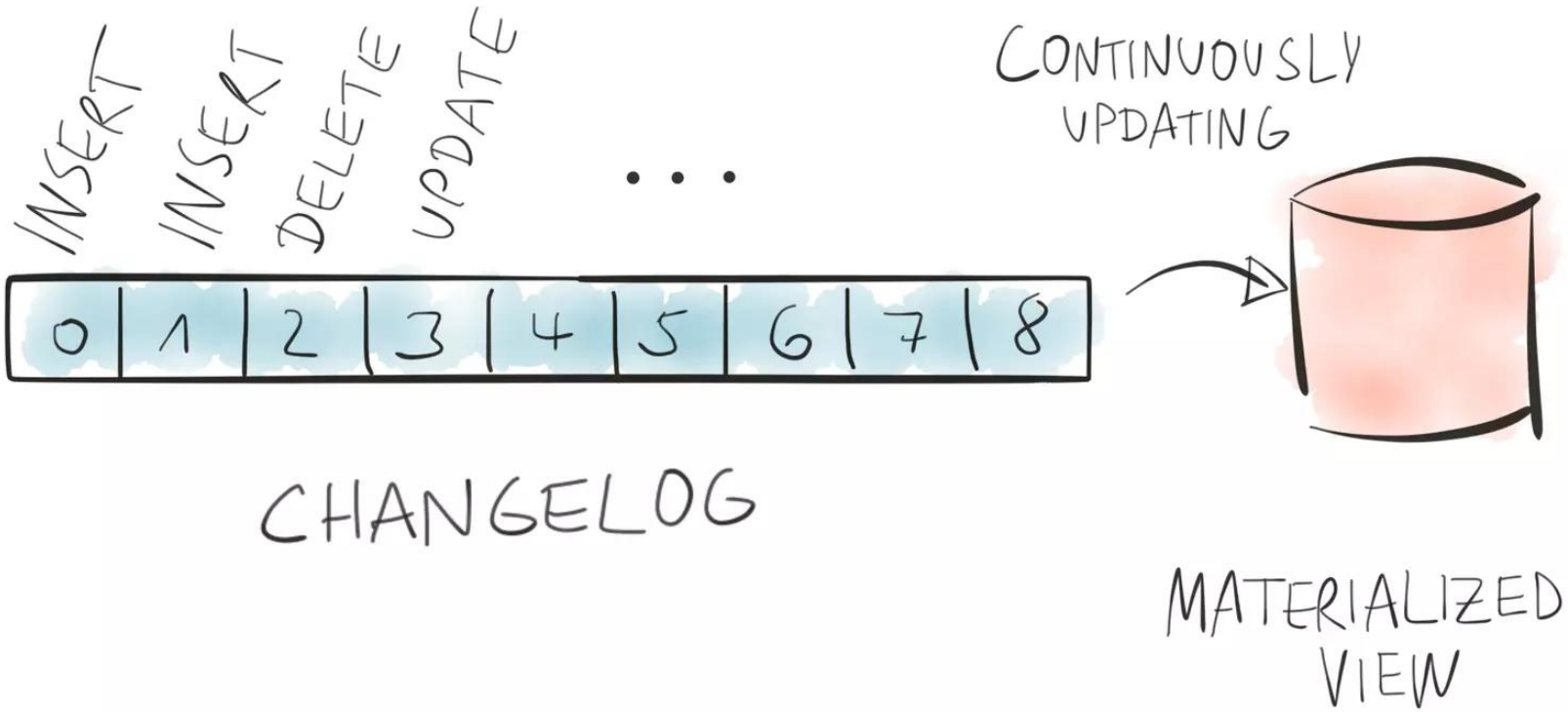

```
KTable<Long,Long> countPerKey
    = stream.groupByKey().count();

countPerKey.toStream().to(“topic”);
```



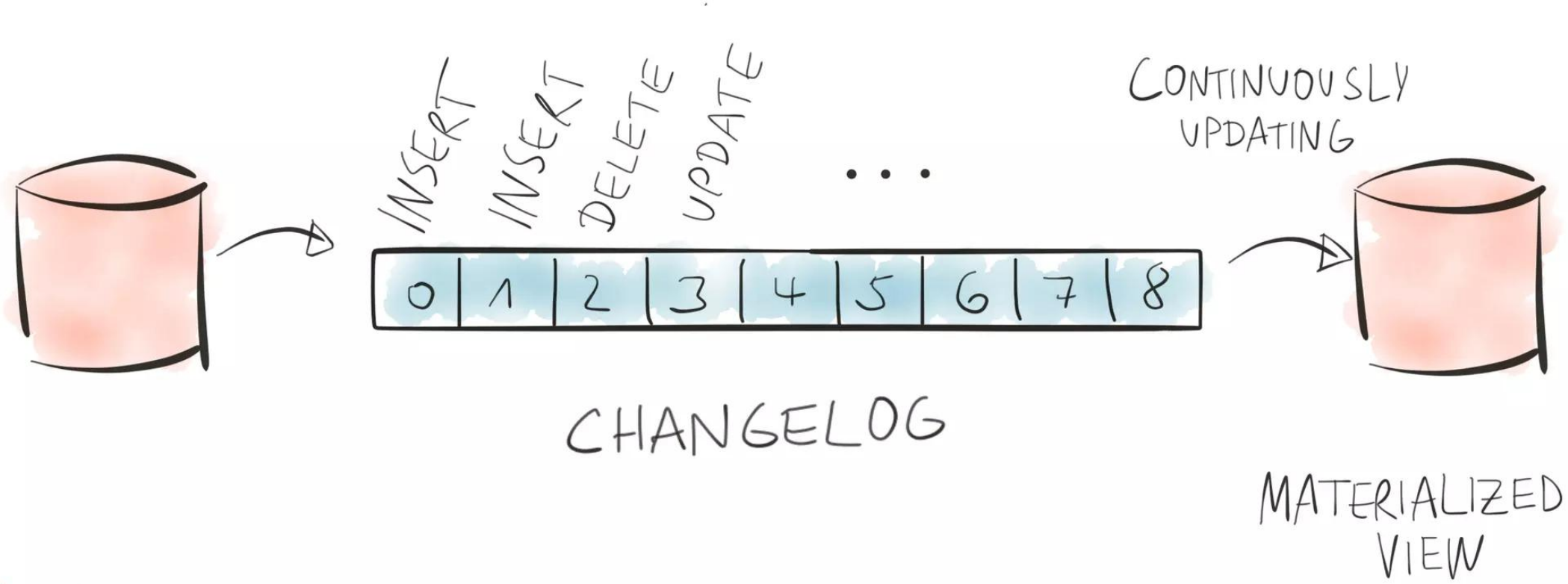
Creating Tables

```
KTable<Long,String> inputTable = builder.table("changelog-topic");
```

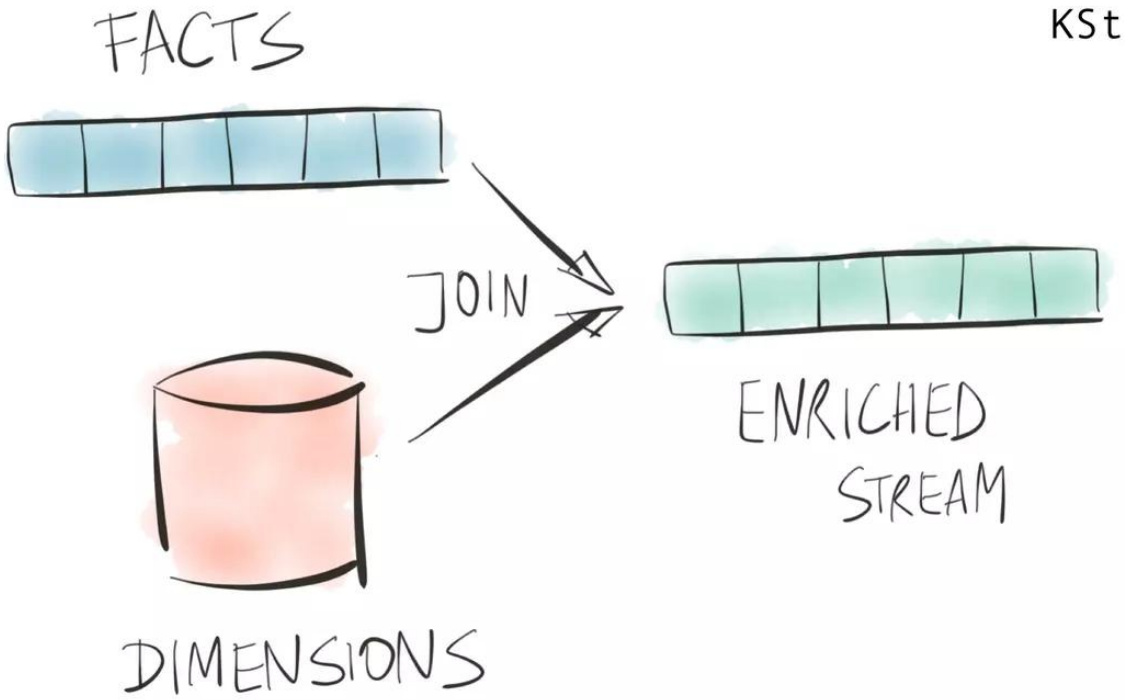


Transforming Tables

```
KTable<Long,String> outputTable = inputTable.filter();
```



Joining Streams and Tables




```
KStream<Long,String> enrichedStream =
    inputStream.join(inputTable, …);
```

groupBy

windowBy
(tumbling,
hopping,
session)

flatMap

filter

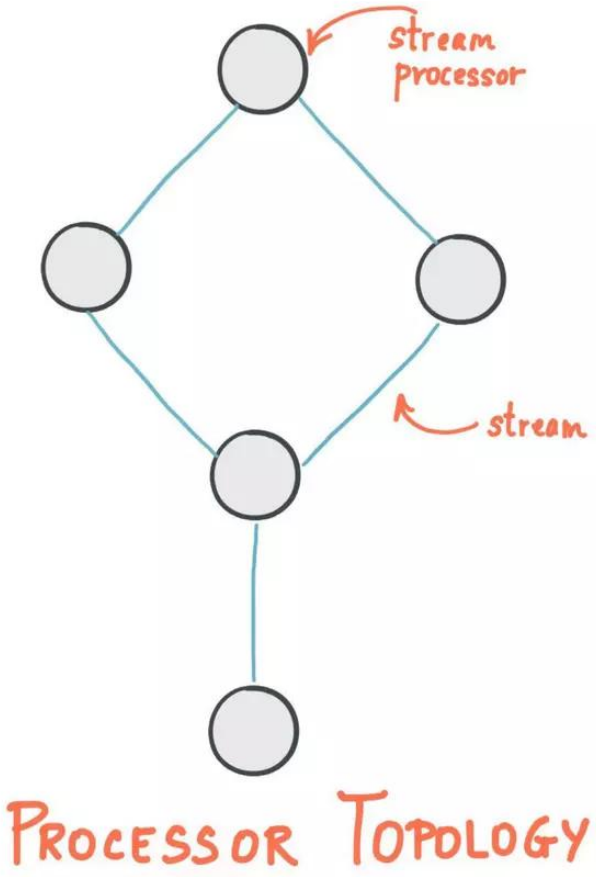
foreach

mapValues

join
(stream-stream,
stream-table,
table-table)

aggregate

Processor API




```
Topology topology = new Topology();

topology.addSource(…);
topology.addProcessor(…);
topology.addSink(…);

topology.addStateStore(…);

// DSL compiles down to a topology:
// StreamsBuilder builder = …
// Topology topology = builder.build();


KafkaStreams instance =
  new Kafka Streams(topology, config);
instance.start();
```

The Processor Interface

```
public class MyProcessor<K,V>
  implements Processor<K,V> {

  private ProcessorContext context;
  private KeyValueStore store;

  public void init(ProcessorContext c) {
    context = c;
    store = context.getStateStore(…);
  }

  public void process(K key, V value) {
    // business logic
    state.put(key, value);
    context.forward(key, value);
  }

  public void close() {}
}
```

- Full flexibility
- Emit zero of more output records
- Lookup or modify data in your state stores
- Access metadata (e.g., timestamp, headers, etc) via the `context`
- Schedule punctuations:
  - `context.schedule(…)`
  - Event or wall-clock time based
  - Regular callbacks


Flexibility
Simplicity
Consumer, Producer
subscribe()
poll()
send()
flush()
Kafka Streams
filter()
join()
aggregate()
KSQL
Select…from…
Join…where…
Group by..

Consumer, Producer

- subscribe()
- poll()
- send()
- flush()

Streams PAPI

- source / processor / sink
- state store
- process()
- punctuations

Streams DSL

- filter()
- join()
- aggregate()

KSQL

- Select…from…
- Join…where…
- Group by..

How to get started…

- https://docs.confluent.io/current/streams/index.html
- https://github.com/confluentinc/kafka-streams-examples
- https://www.confluent.io/resources/
  - Kafka Summit talk recording
  - White papers
  - Books
- https://www.manning.com/books/kafka-streams-in-action
  - By Bill Bejeck

Getting help:

- User mailing list: https://kafka.apache.org/contact
- Confluent Google Group: https://groups.google.com/forum/#!topic/confluent-platform
- Confluent Community Slack: https://launchpass.com/confluentcommunity
- StackOverflow

Until the end of 2019 Confluent are giving new users $50 of free usage per month for their first 3 months

Here's advice on how to use this promotion to try Confluent Cloud for free!

Sign up for a Confluent Cloud account

Please bear in mind that you will be required to enter credit card information but will not be charged unless you go over the $50 usage in any of the first 3 months or if you don't cancel your subscription before the end of your promotion.

You won't be charged if you don't go over the limit!

Get the benefits of Confluent Cloud, but keep an eye on your your account making sure that you have enough remaining free credits available for the rest of your subscription month!!

Cancel before the 3 months end If you don't want to continue past the promotion

If you fail to cancel within your first three months you will start being charged full price. To cancel, immediately stop all streaming and storing data in Confluent Cloud and email cloud-support@confluent.io

bit.ly/**TryConfluentCloud**

Available on

Google Cloud Platform

Microsoft Azure

aws

We are hiring!


@MatthiasJSax

matthias@confluent.io | mjsax@apache.org