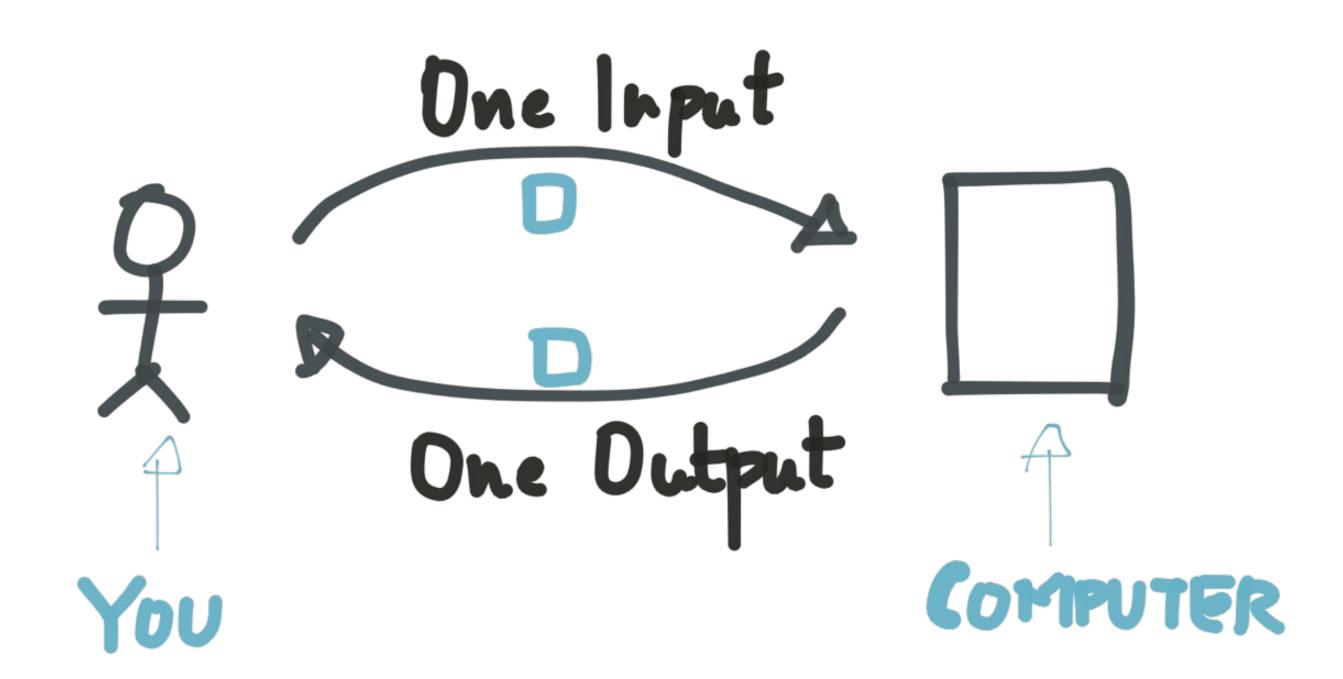
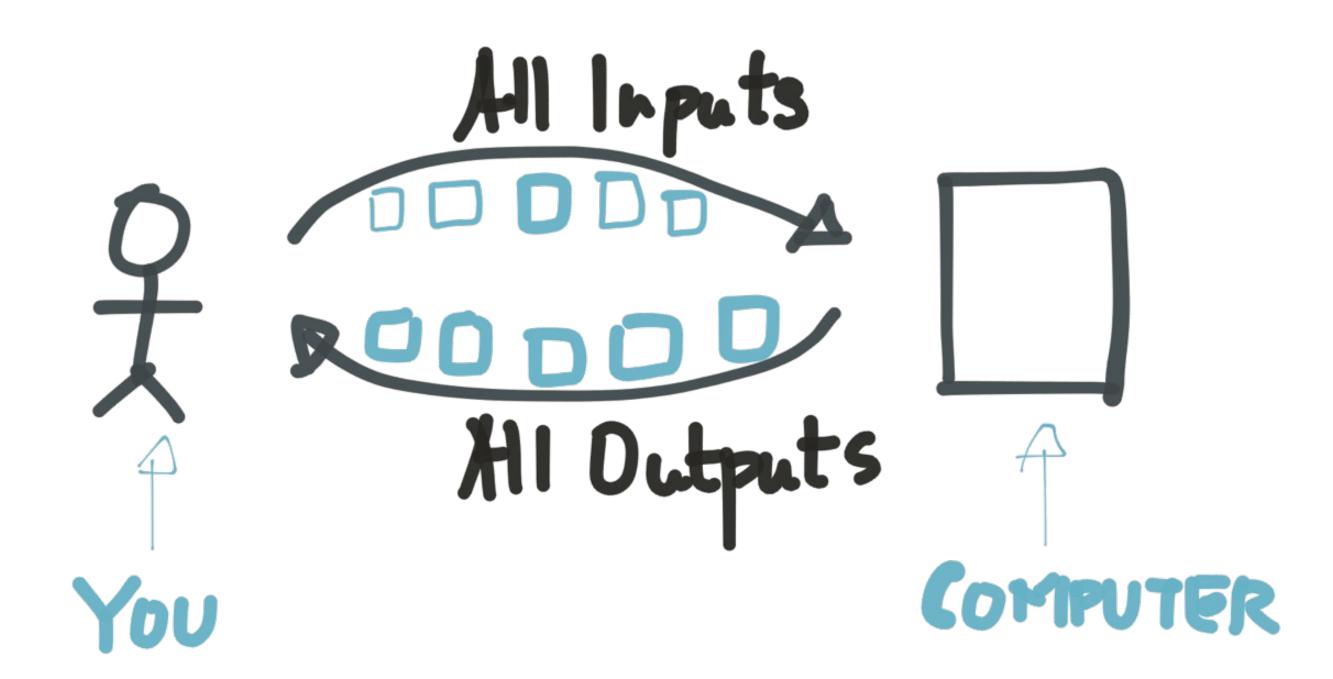
Kafka Streams

Stream processing Made Simple with Kafka

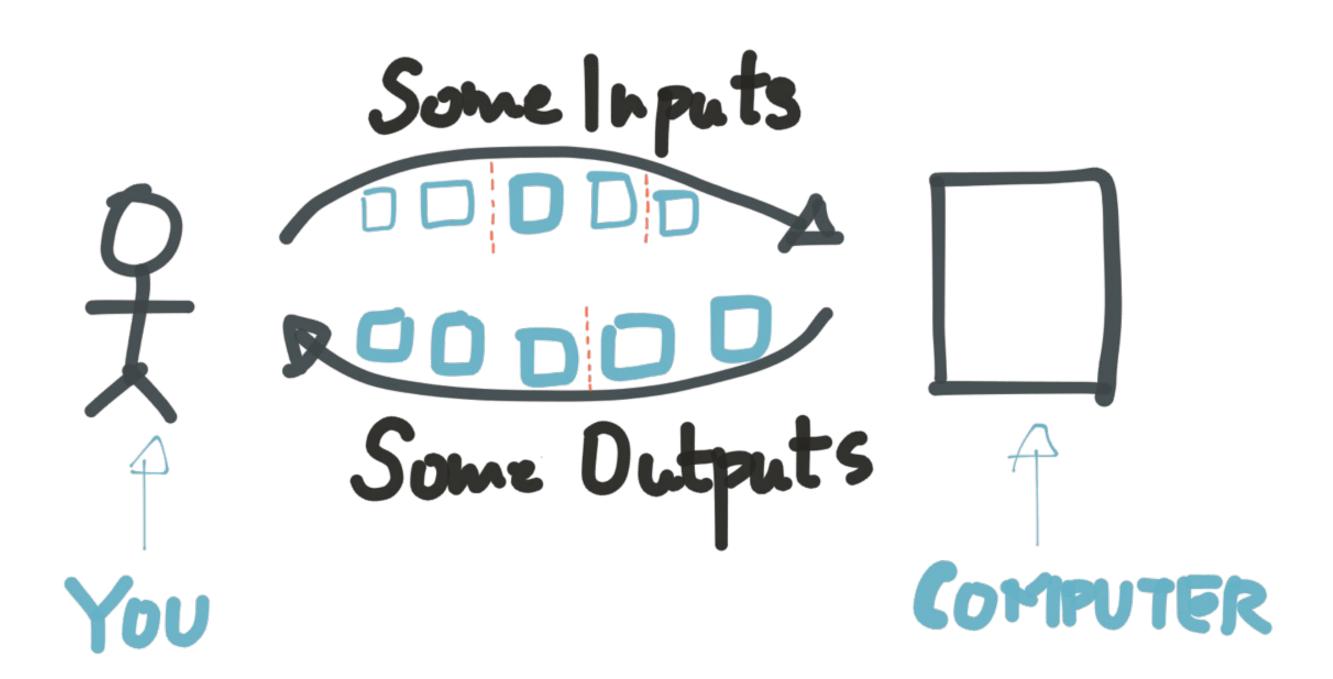
REQUEST/RESPONSE



BATCH



STREAM ROCESSING



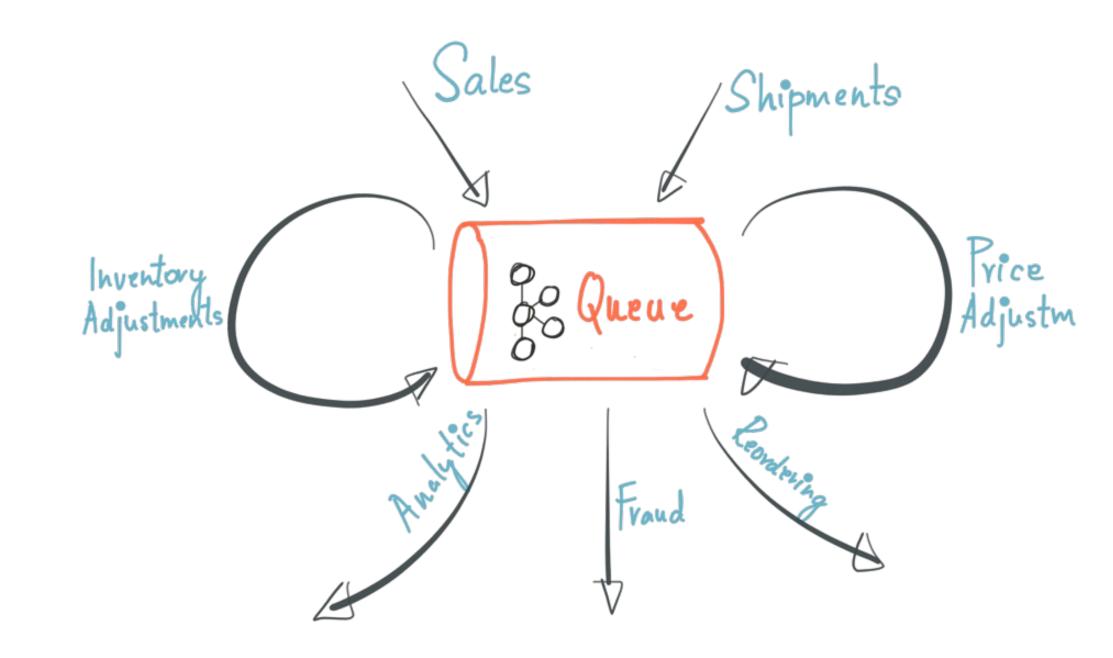
Stream Processing

- A different programming paradigm
 - .. that brings computation to unbounded data
 - .. with tradeoffs between latency / cost / correctness

Kafka Streams is a lightweight, yet powerful Java library for enriching, transforming, and processing real-time streams of data.

Kafka: Real-time Platforms

- Persistent Buffering
- Logical Ordering
- Scalable "source-of-truth"



APIs for moving data to and from Kafka

API	Topic interaction	Examples
Producer API	Writing messages to Kafka topics.	FilebeatrsyslogCustom producers
Consumer API	Reading messages from Kafka topics.	LogstashkafkacatCustom consumers
Connect API	Connecting external data stores, APIs, and filesystems to Kafka topics. Involves both reading from topics (sink connectors) and writing to topics (source connectors).	 JDBC source connector Elasticsearch sink connector Custom connectors

Stream Processing with Kafka

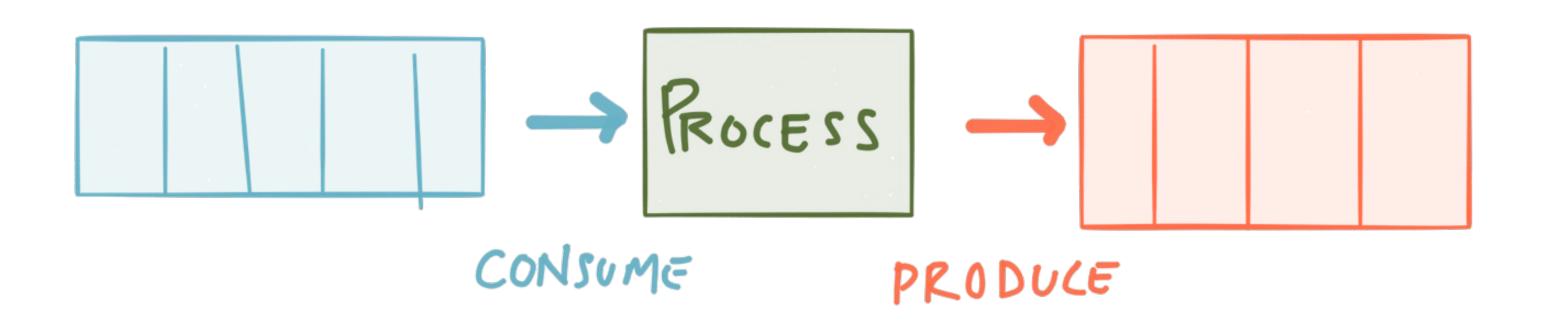
Option I:Do It Yourself!

```
while (isRunning) {
    // read some messages from Kafka
    inputMessages = consumer.poll();

    // do some processing...

// send output messages back to Kafka
    producer.send(outputMessages);
}
```

DILY STREAM PROCESSING



DIY Stream Processing is Hard

Ordering

• State Management

Partitioning &
 Scalability

- Time, Window &
 Out-of-order Data
- Fault tolerance
- Re-processing

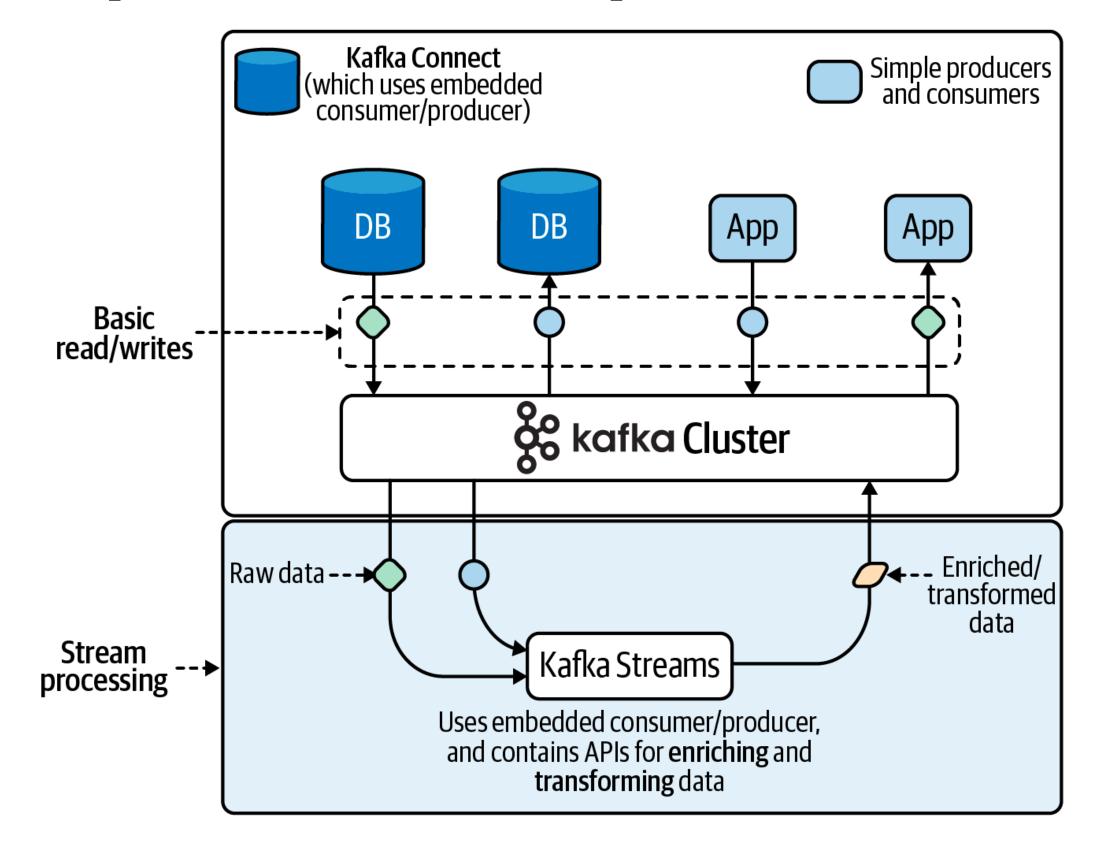


- In Apache Kafka since v0.10, May 2016
- Powerful yet easy-to-use stream processing library
 - Event-at-a-time, Stateful
 - Windowing with out-of-order handling
 - Highly scalable, distributed, fault tolerant
 - and more..

Anywhere, anytime

```
<dependency>
  <groupId>org.apache.kafka</groupId>
   <artifactId>kafka-streams</artifactId>
   <version>2.10.0.0</version>
</dependency>
```

Anywhere, anytime



Kafka Streams DSL

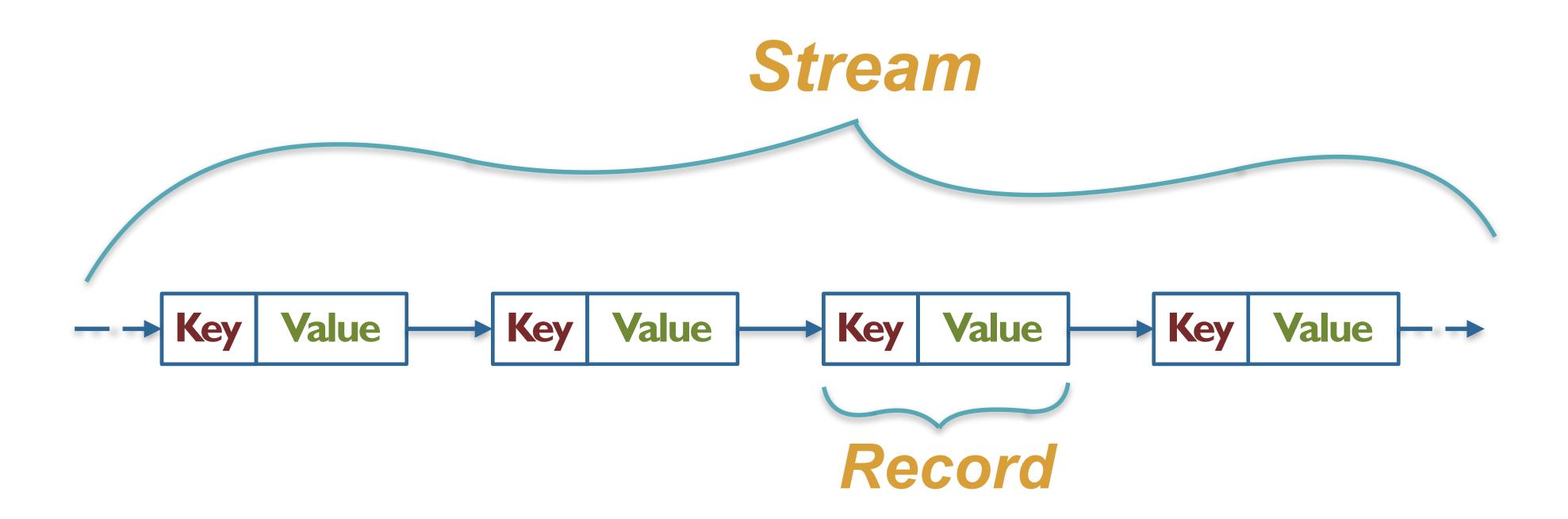
```
public static void main(String[] args) {
   // specify the processing topology by first reading in a stream from a topic
    KStream<String, String> words = builder.stream("topic1");
    // count the words in this stream as an aggregated table
    KTable<String, Long> counts = words.countByKey("Counts");
    // write the result table to a new topic
    counts.to("topic2");
    // create a stream processing instance and start running it
    KafkaStreams streams = new KafkaStreams(builder, config);
    streams.start();
```

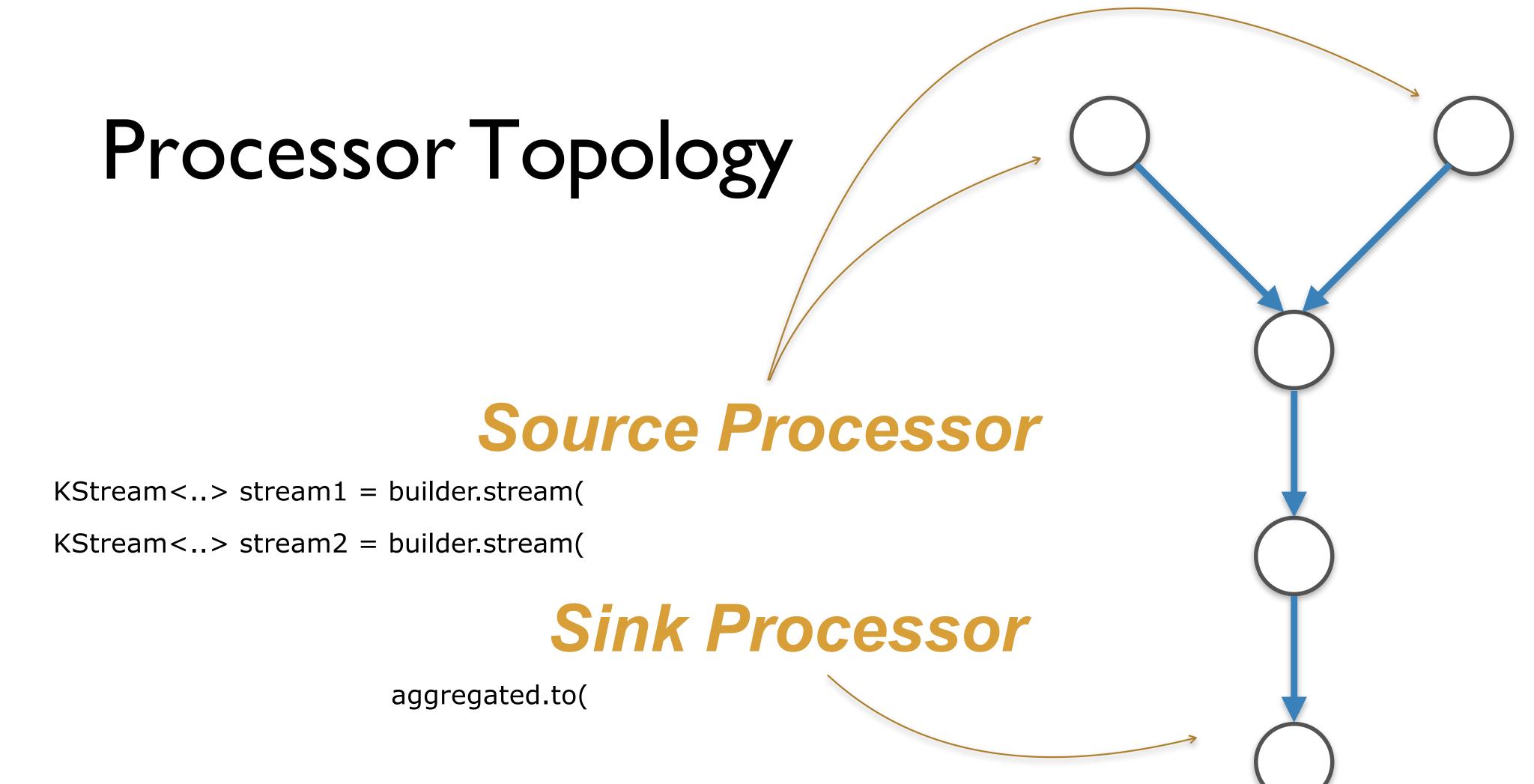
Native Kafka Integration

```
Property cfg = new Properties();
cfg.put(StreamsConfig.APPLICATION_ID_CONFIG, "my-streams-app");
cfg.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "broker1:9092");
cfg.put(ConsumerConfig.AUTO_OFFSET_RESET_CONIFG, "earliest");
cfg.put(CommonClientConfigs.SECURITY_PROTOCOL_CONFIG, "SASL_SSL");
cfg.put(KafkaAvroSerDeConfig.SCHEMA_REGISTRY_URL_CONFIG, "registry:8081");
StreamsConfig config = new StreamsConfig(cfg);
final StreamsBuilder builder = new StreamsBuilder();
KafkaStreams streams = new KafkaStreams(builder, config);
```

Kafka Streams: Key Concepts

Stream and Records





Stream API features:

- A high-level DSL that looks and feels like Java's streaming API
- A low-level Processor API that gives developers fine-grained control when they need it.
- Convenient abstractions for modeling data as either streams or tables
- The ability to join streams and tables, which is useful for data transformation and enrichment.
- Support for time-based operations, including windowing and periodic functions.
- Easy installation. It's just a library, so you can add Kafka Streams to any Java application.
- Scalability, reliability, maintainability

Uses Cases

- Financial data processing (Flipkart), purchase monitoring, fraud detection
- Algorithmic trading
- Stock market/crypto exchange monitoring
- Real-time inventory tracking and replenishment (Walmart)
- Event booking, seat selection (Ticketmaster)
- Email delivery tracking and monitoring (Mailchimp)

Lab: Using kafka Stream – 90 Minutes