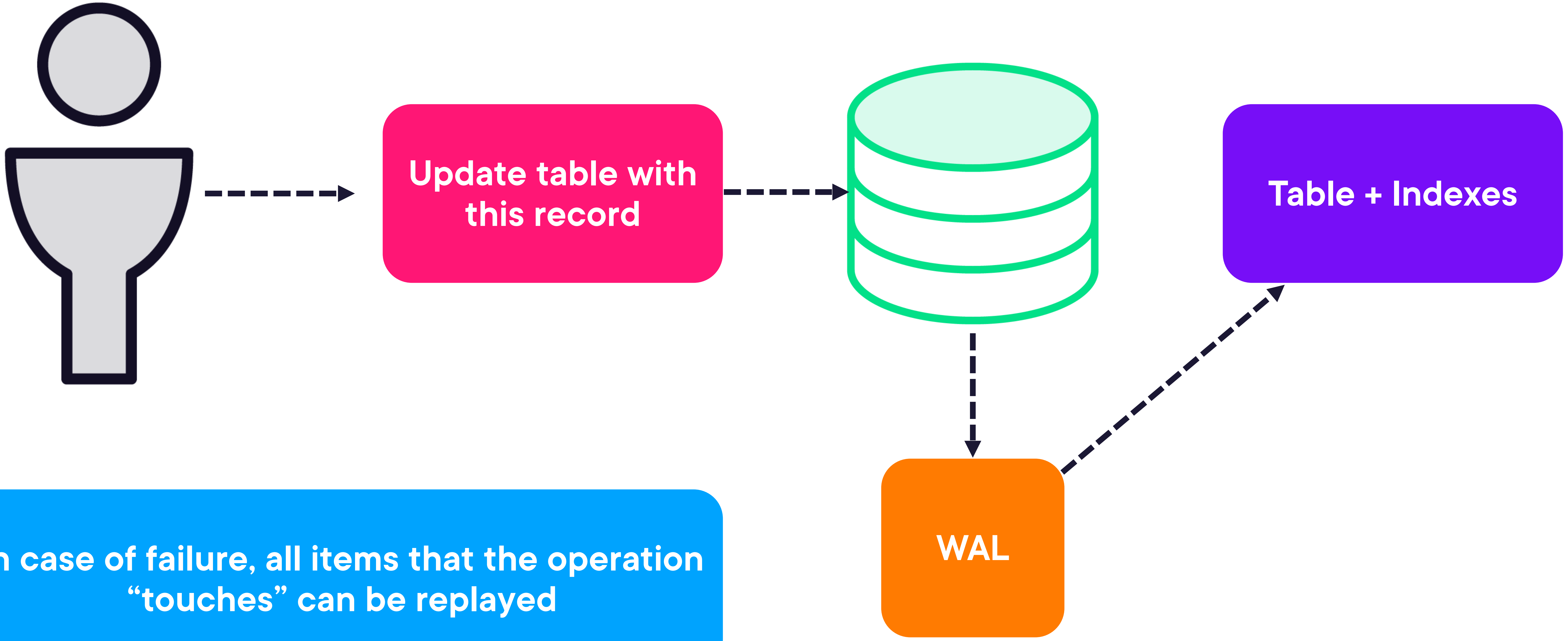
Kafka Streams

Axel Sirota

AI and Cloud Consultant

@AxelSirota

The WAL is the source of truth!



In case of failure, all items that the operation "touches" can be replayed

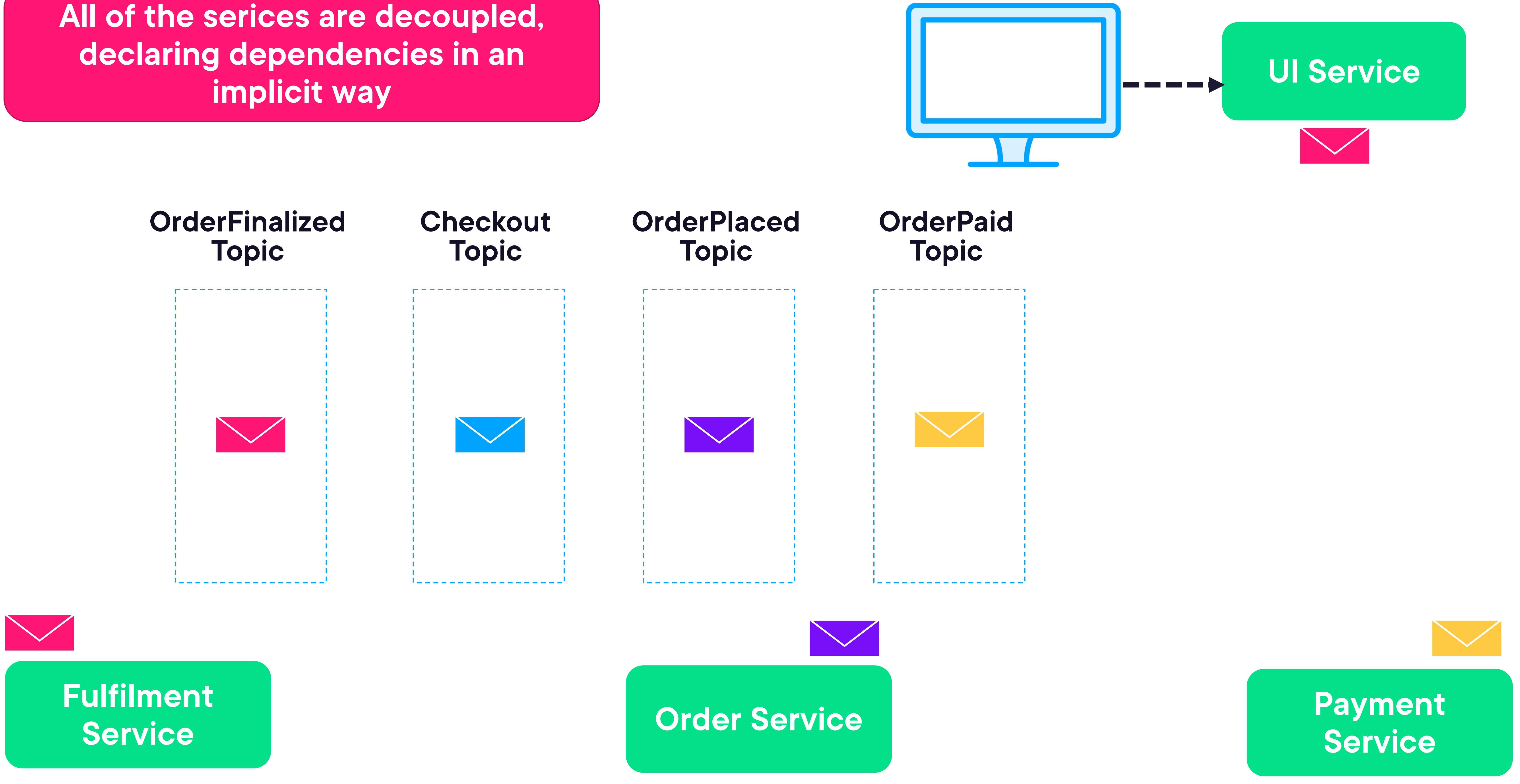
Each of these services will be a Kafka Streams app in general

Synchronous Microservices via REST, gRPC

Asynchronous Microservices via messaging (Kafka, Message Bus, etc...)

We are going to evaluate this one!

All of the serices are decoupled, declaring dependencies in an implicit way



Reasons

1ST

Every event happening in your application can be traced back to a Kafka topic, and that is stored

2ND

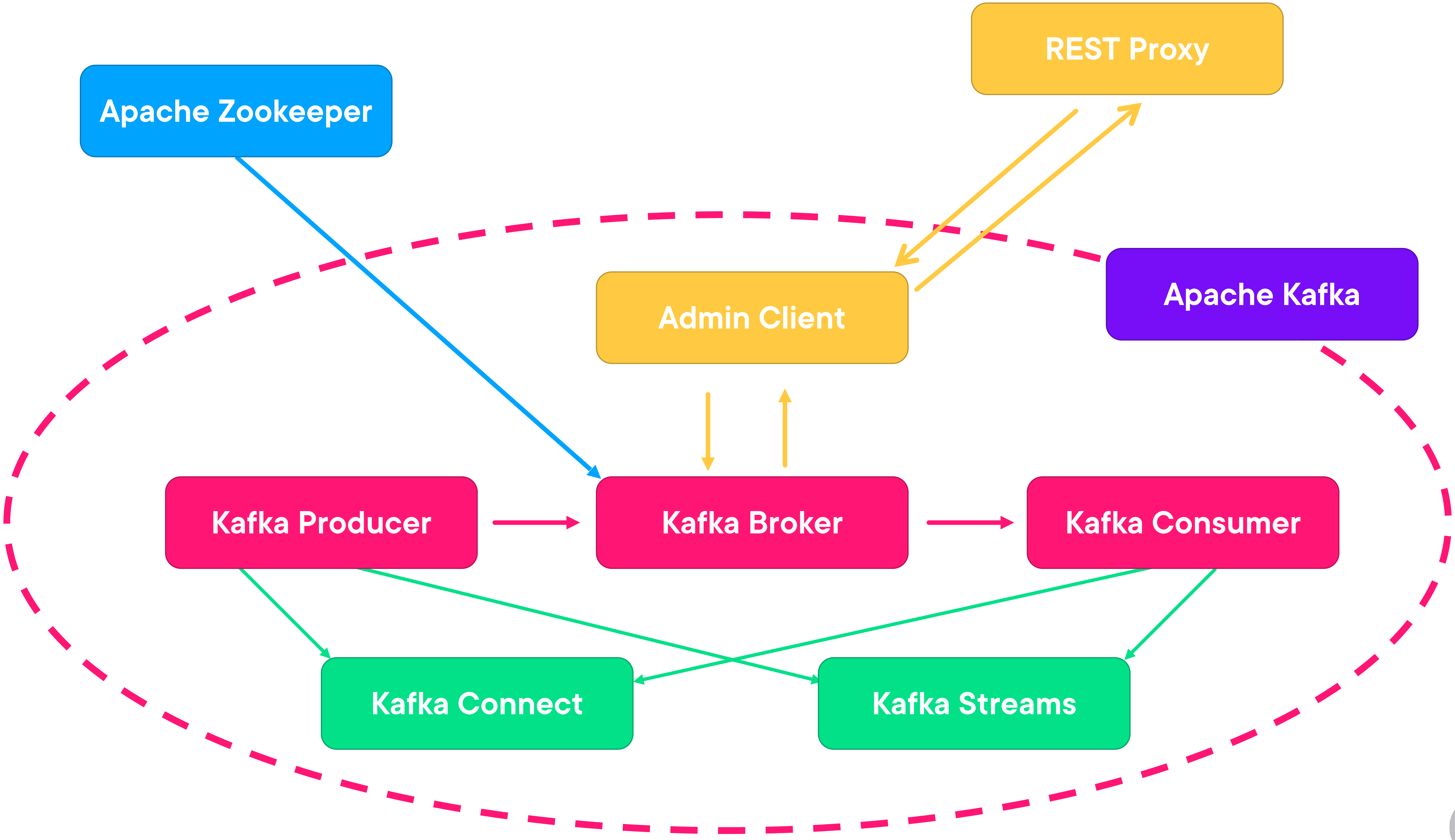
Not only this means that if each of these microservices are Kafka Streams apps, then everything in your app goes through Kafka. And Kafka topics have an abstraction called the log that makes it posible replay messages

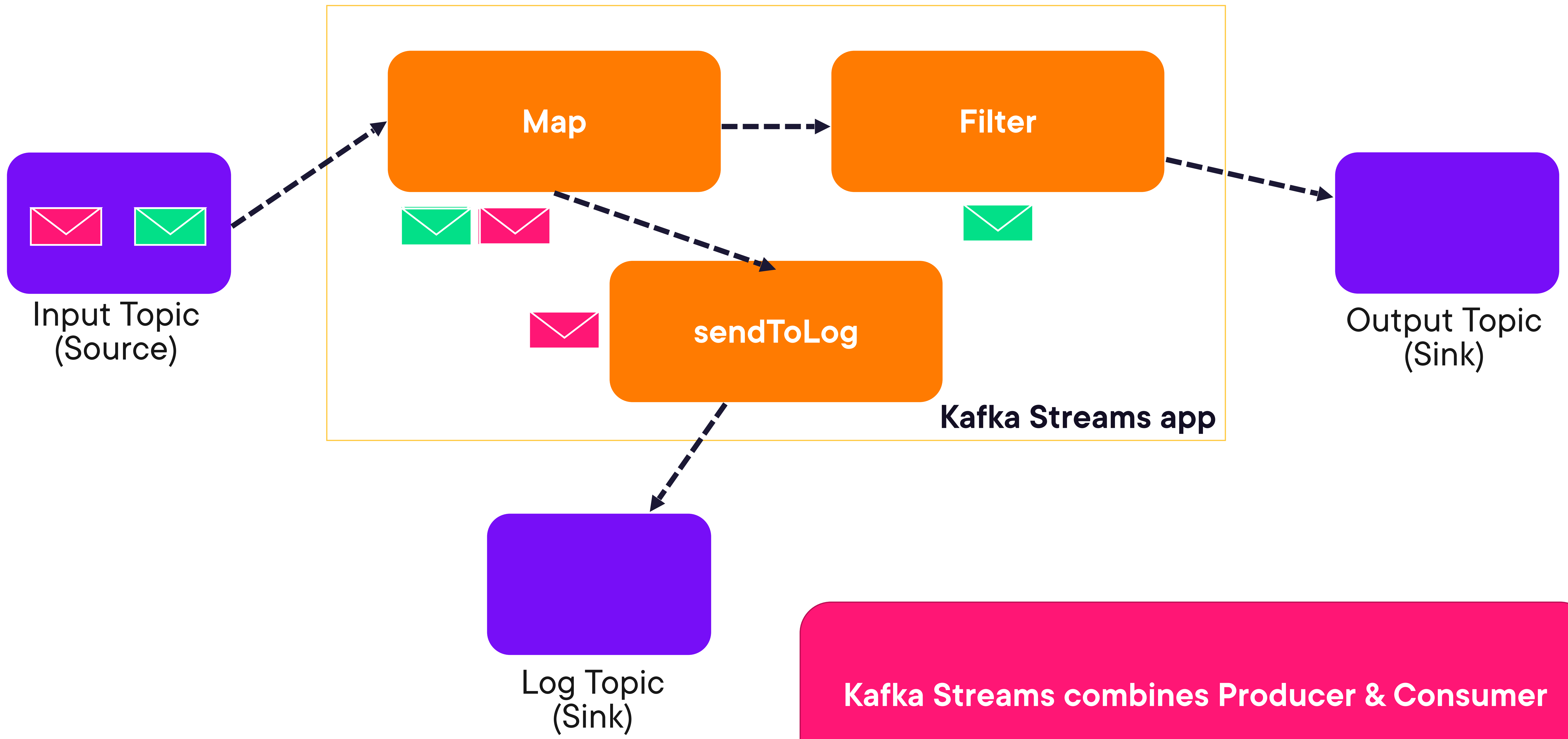
3RD

Which means Kafka becomes the WAL of the application and allows for things like multi-phase transactions

So the only remaining part missing is: can we query Kafka?

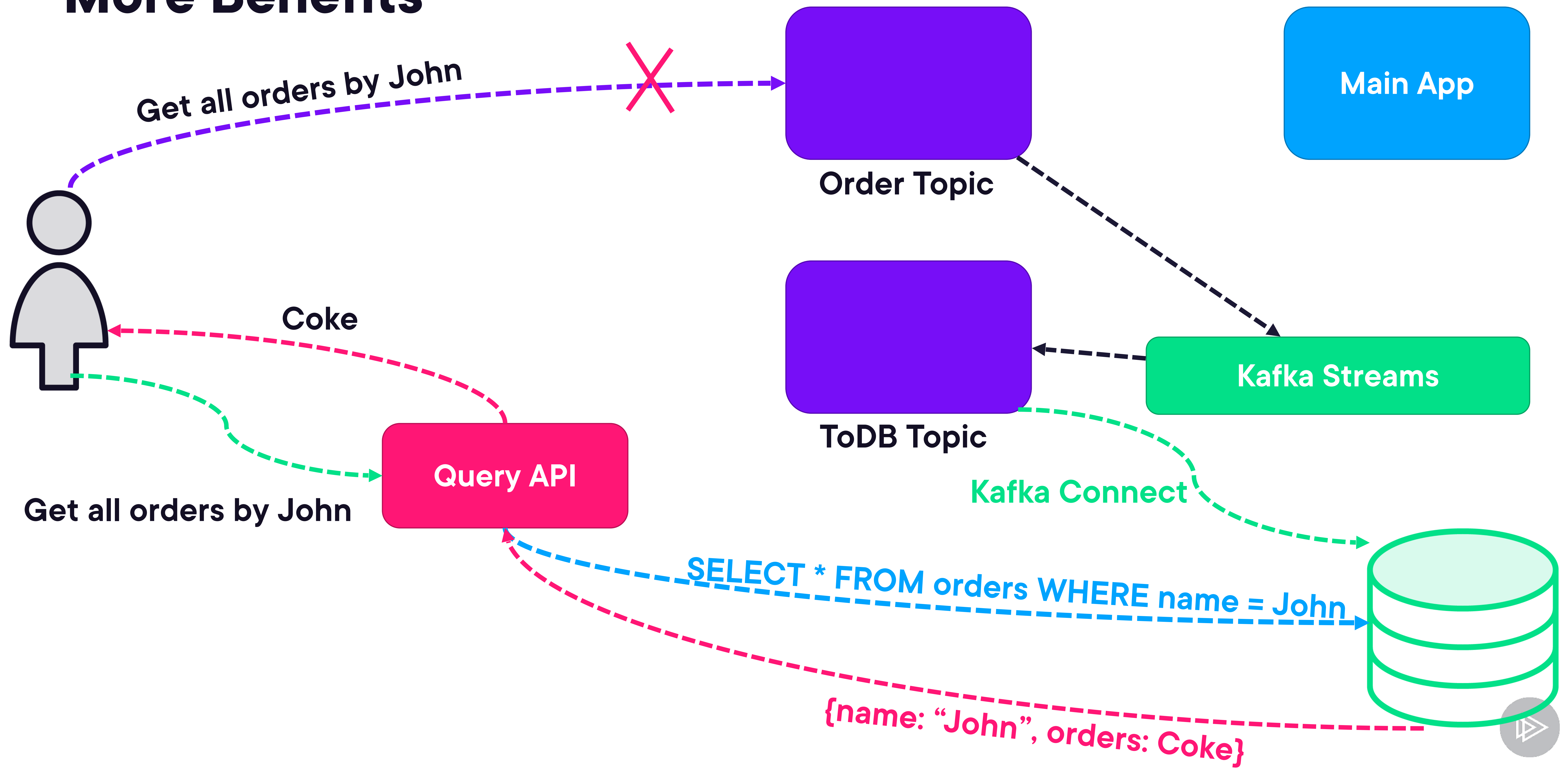
Kafka Streams and Stateless Stream Processing


REST Proxy
Apache Zookeeper
Admin Client
Apache Kafka
Kafka Producer
Kafka Broker
Kafka Consumer
Kafka Connect
Kafka Streams



Kafka Streams combines Producer & Consumer

More Benefits



Creating a Kafka Streams Application

```
Properties props = new Properties();
props.put(StreamsConfig.APPLICATION_ID_CONFIG, "my_stream_app");
props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.Integer().getClass());
props.put("schema.registry.url", "http://localhost:8081");
```

Establishing Properties

- Create an application.id that represents your application group or "team"
- Serde is combination of Serializer/Deserializer
- Every stream application and KSQL app (later) is a consumer-producer

```
StreamsBuilder builder = new StreamsBuilder();

KStream<String, DisasterValue> rawReadings = builder.stream("DisasterReadings",
Consumed.with(Serdes.String(), Serdes.Integer()))
```

Create a Stream Builder

- Always start with a StreamerBuilder object
- This is the GoF builder pattern, where we will create a Topology object that represents our data pipeline

Standard Functional Programming

- map
- filter
- flatMap
- groupBy
- reduce
- window
- join
- leftJoin
- outerJoin

Given a Stream

Mapping

```
(1, "Hello"), (2, "Zoom"), (3, "Fold")

stream.map((key, value) -> new KeyValue<>(key +
1, value + "!"));

(2, "Hello!"), (3, "Zoom!"), (4, "Fold!")
```

Applying map

Mapping

```
(1, "Hello"), (2, "Zoom"), (3, "Fold")
```

```
stream.map((key, value) -> new KeyValue<>(key +
1, value + "!"));
```

```
(2, "Hello!"), (3, "Zoom!"), (4, "Fold!")
```

Resulting in

Mapping

```
(1, "Hello"), (2, "Zoom"), (3, "Fold")

stream.map((key, value) -> new KeyValue<>(key +
1, value + "!"));

(2, "Hello!"), (3, "Zoom!"), (4, "Fold!")
```

Given a Stream

Filtering

```
(1, "Hello"), (2, "Zoom"), (3, "Fold")

stream.filter((key, value) -> key % 2 == 0);

(2, "Zoom"), (4, "Past")
```

Applying filter

Filtering

```
(1, "Hello"), (2, "Zoom"), (3, "Fold")
```

```
stream.filter((key, value) -> key % 2 == 0);
```

```
(2, "Zoom"), (4, "Past")
```

Resulting in

Filtering

```
(1, "Hello"), (2, "Zoom"), (3, "Fold")

stream.filter((key, value) -> key % 2 == 0);

(2, "Zoom"), (4, "Past")
```

```
KStream stream = builder.stream("my_topic");
stream.filter(...).through("new_topic").flatMap(...).to("other_topic")
```

Dump Results to a Topic

Dump the results to a topic using through to post to topic and continue

And Run

Build the Topology and Stream

```
Topology topology = builder.build();

KafkaStreams streams = new
KafkaStreams(topology, props);

streams.start();
```

```
Runtime.getRuntime().addShutdownHook(new Thread(streams::close));
```

Adding a Shutdown Hook

As always, be a good citizen, properly shutdown resources

Creating a Kafka Streams Application

Querying a Stream with ksql

Key, Takeaways, and Tips

Takeaways

Kafka can effectively act as the WAL for your app when using Kafka Streams

A streams app is just a topology of functional applications to a stream of incoming messages

One can deploy as many streams as you want and they act as microservices

KSQL is the CLI to query KSQLDB which is a thin layer over Kafka Streams and permits to do simple stuff without creating an streams app with code

Keys

- ✓ Try to create a table instead of a stream and query it
- ✓ Try to investigate how to use groupBy and perform JOINs
- ✓ Try to deploy the architecture we mentioned above to query a topic

Up Next:

Administrative Tasks on Kafka