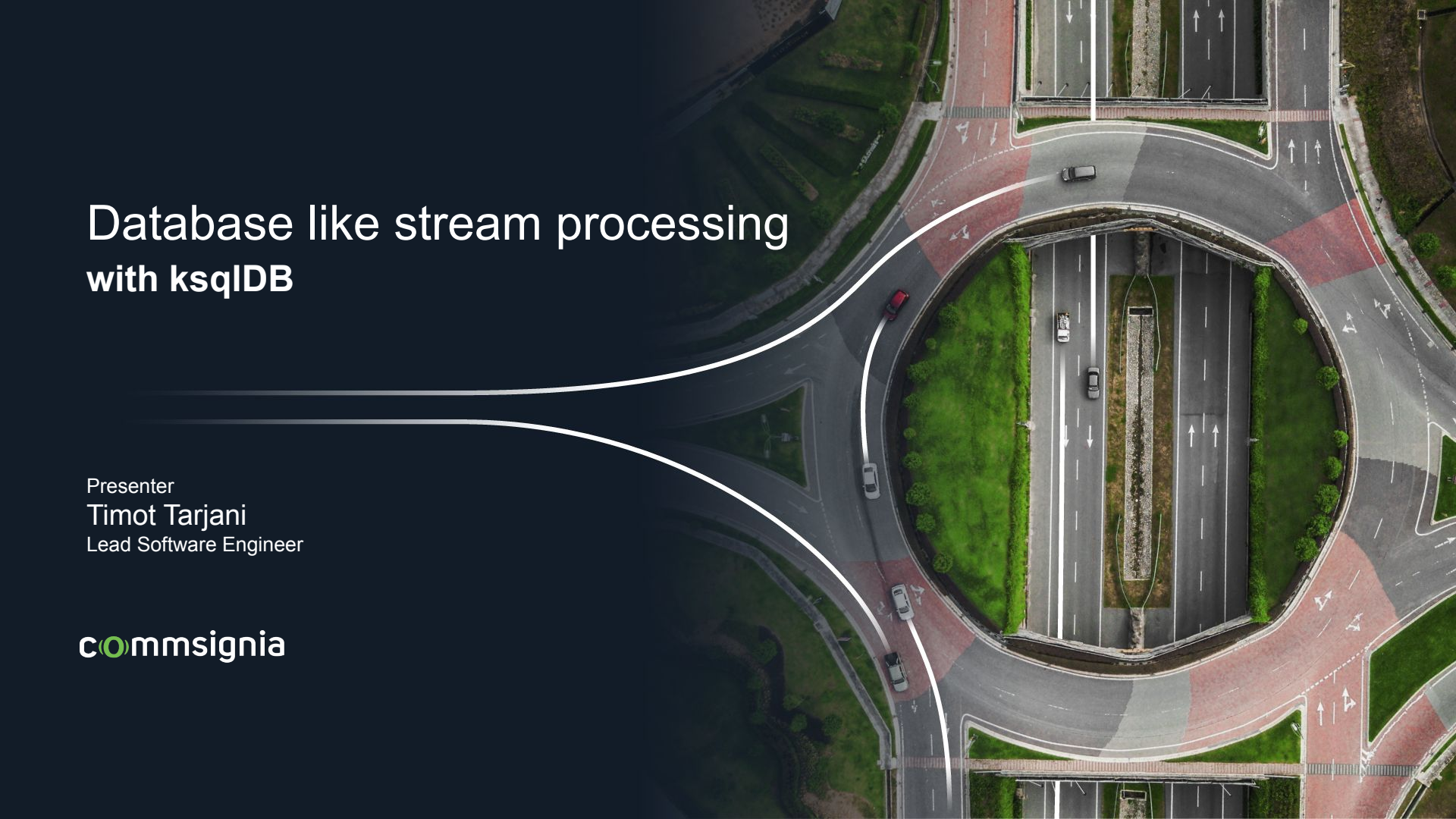


Database like stream processing with ksqlDB

Presenter
Timot Tarjani
Lead Software Engineer

commsignia

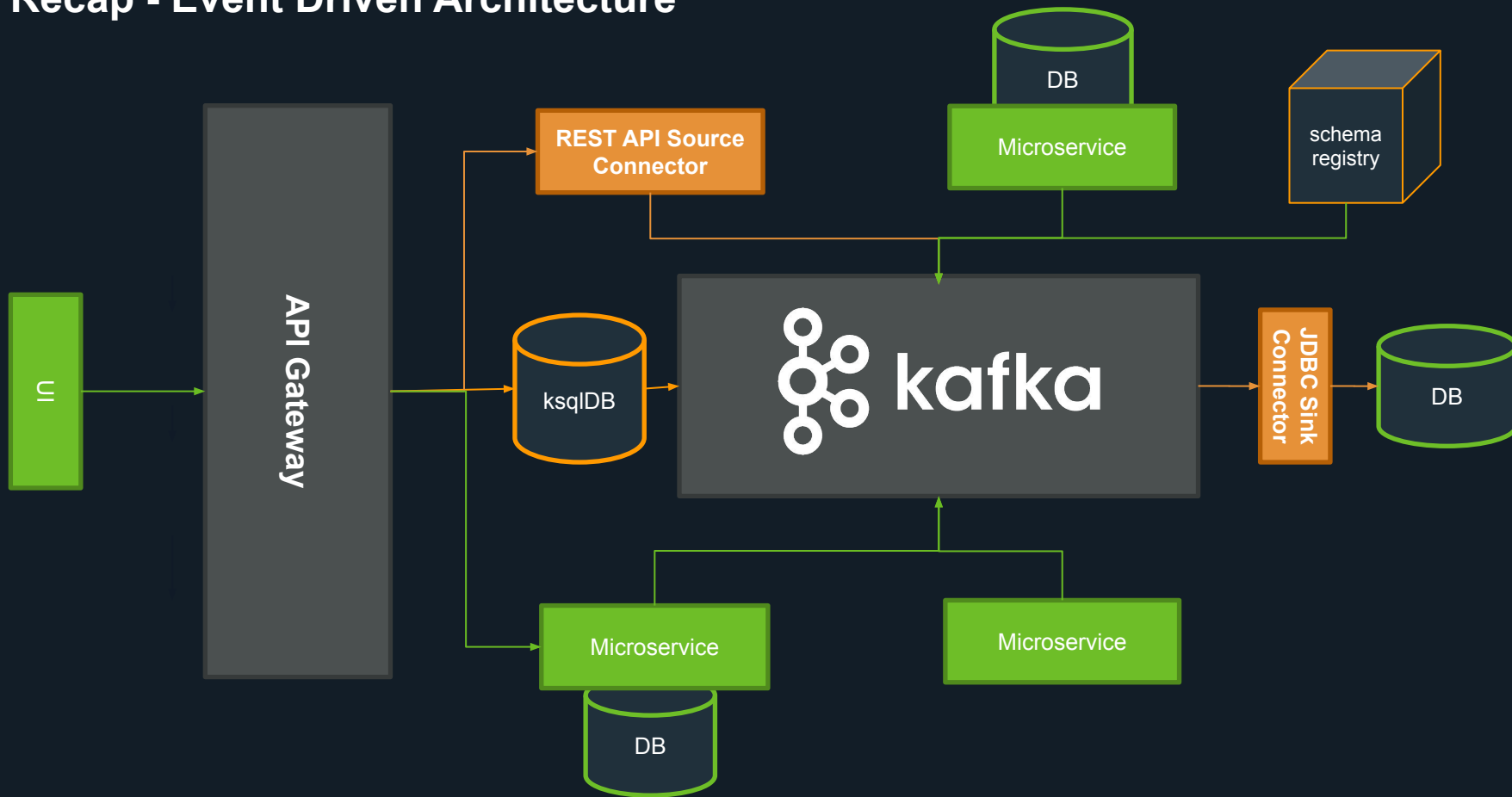


Agenda

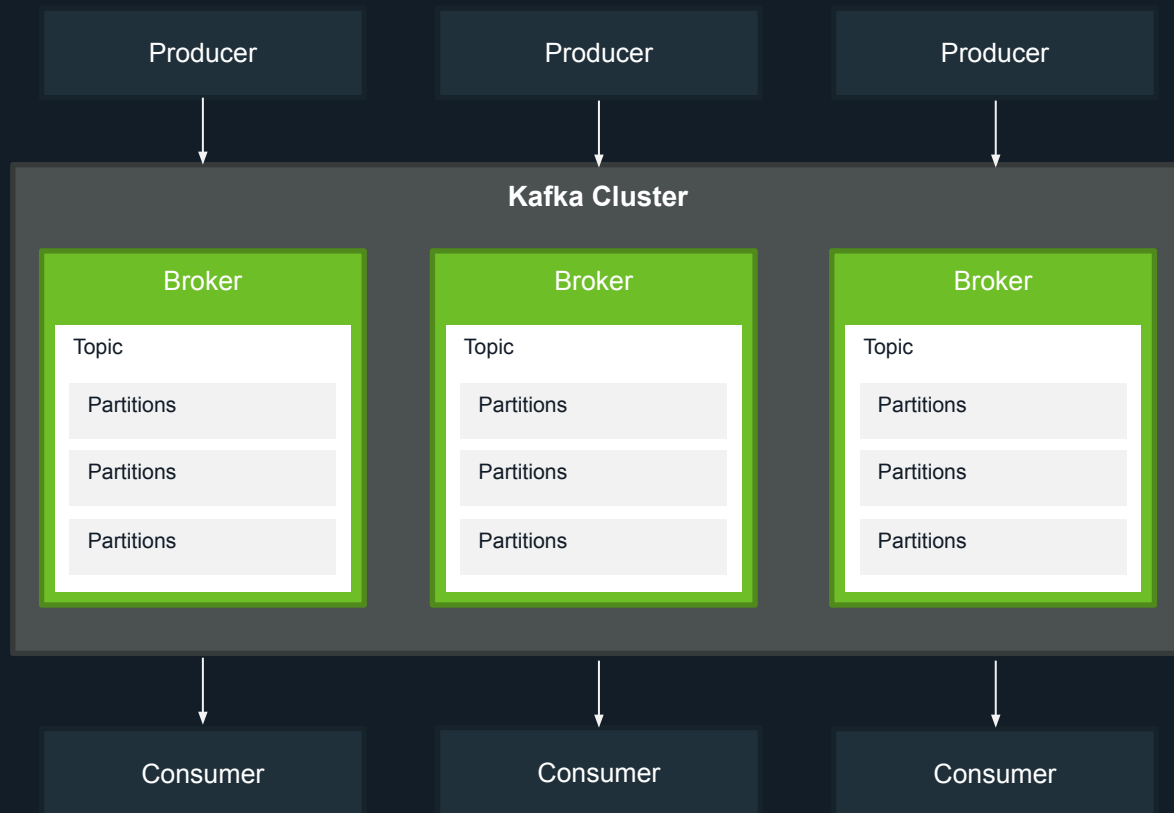
1. Recap
2. Schema Registry
3. ksqIDB
4. Examples



Recap - Event Driven Architecture



Recap - Kafka Architecture



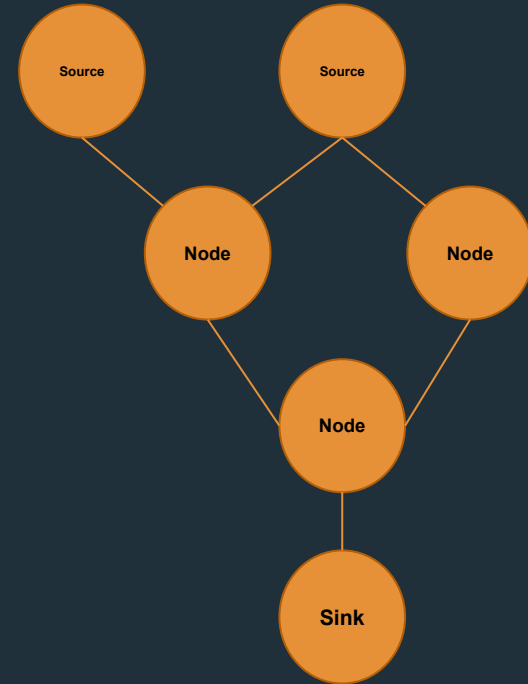
Recap - Kafka Streams Processor Topology

A **processor topology** or simply **topology** defines the computational logic of the data processing that needs to be performed by a stream processing application.

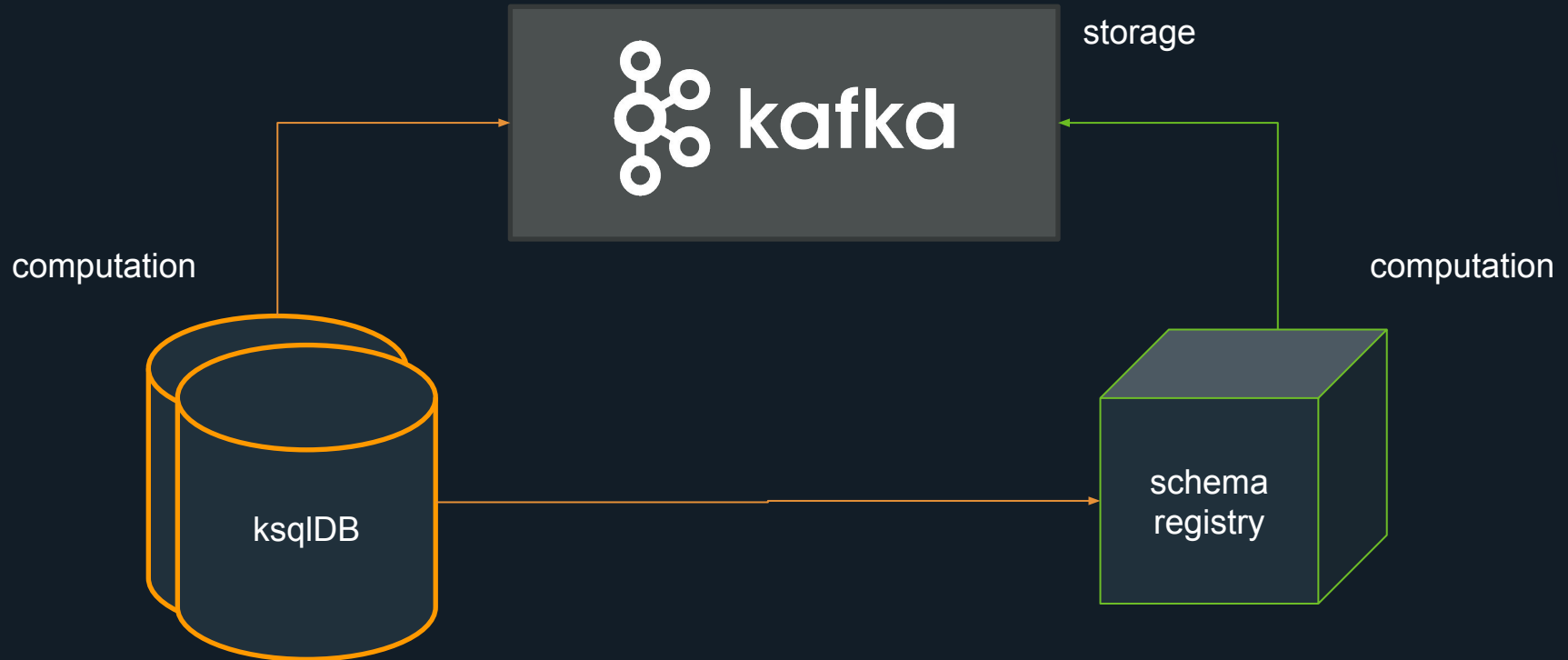
A topology is a graph of stream processors (nodes) that are connected by streams (edges). Developers can define topologies either via the **low-level Processor API** or via the **Kafka Streams DSL**, which builds on top of the former.

1. **Source**
2. **Stream Node 1**
3. **Stream Node 2**
4. **Sink**

Stream Partition → Kafka Topic Partition



Relation between Kafka, ksqlDB and SchemaRegistry

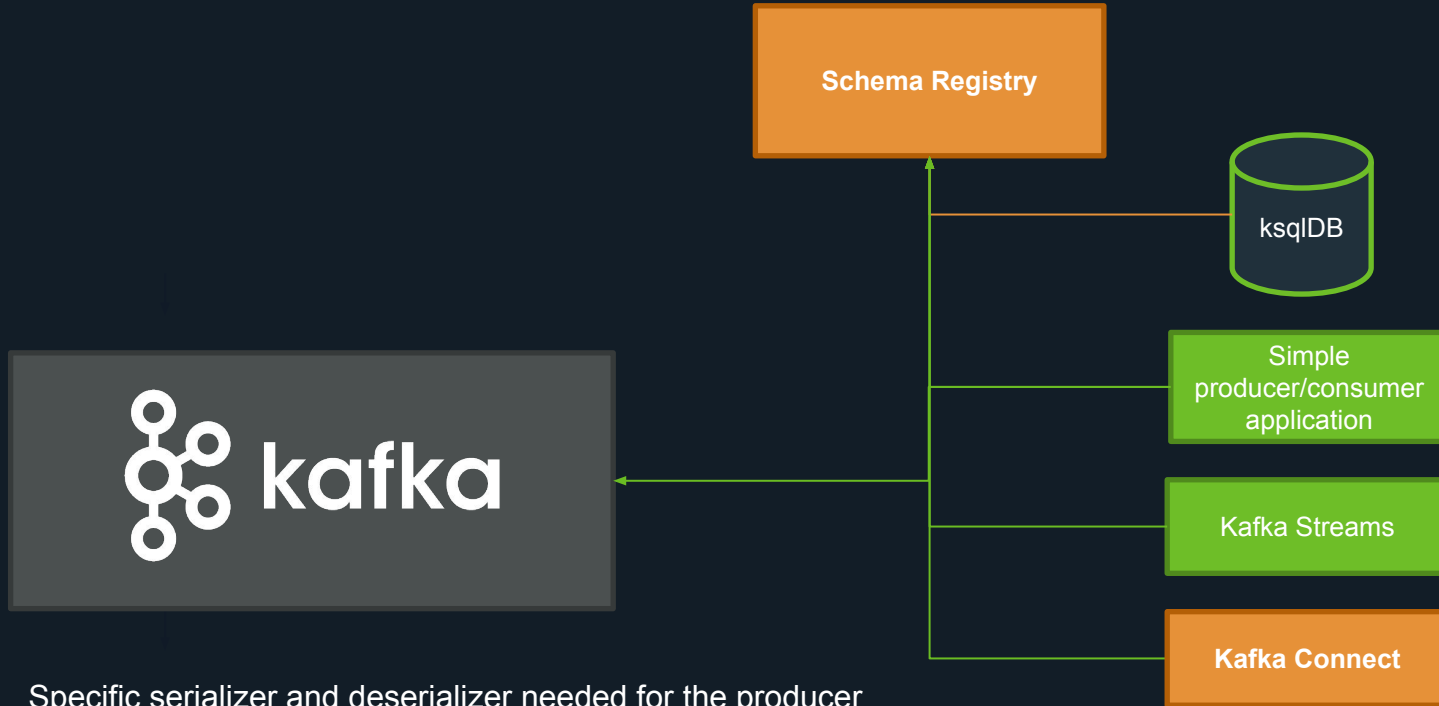


Schema Registry

Schema Registry is a distributed storage layer for schemas which uses Kafka as its underlying storage mechanism. Some key design decisions:

- Assigns globally unique ID to each registered schema. Allocated IDs are guaranteed to be monotonically increasing and unique, but not necessarily consecutive.
- Kafka provides the durable backend, and functions as a write-ahead changelog for the state of Schema Registry and the schemas it contains.
- Schema Registry is designed to be distributed, with single-primary architecture, and ZooKeeper/Kafka coordinates primary election (based on the configuration).
- Supported formats:
 - JSON Schema
 - Protobuf
 - Avro

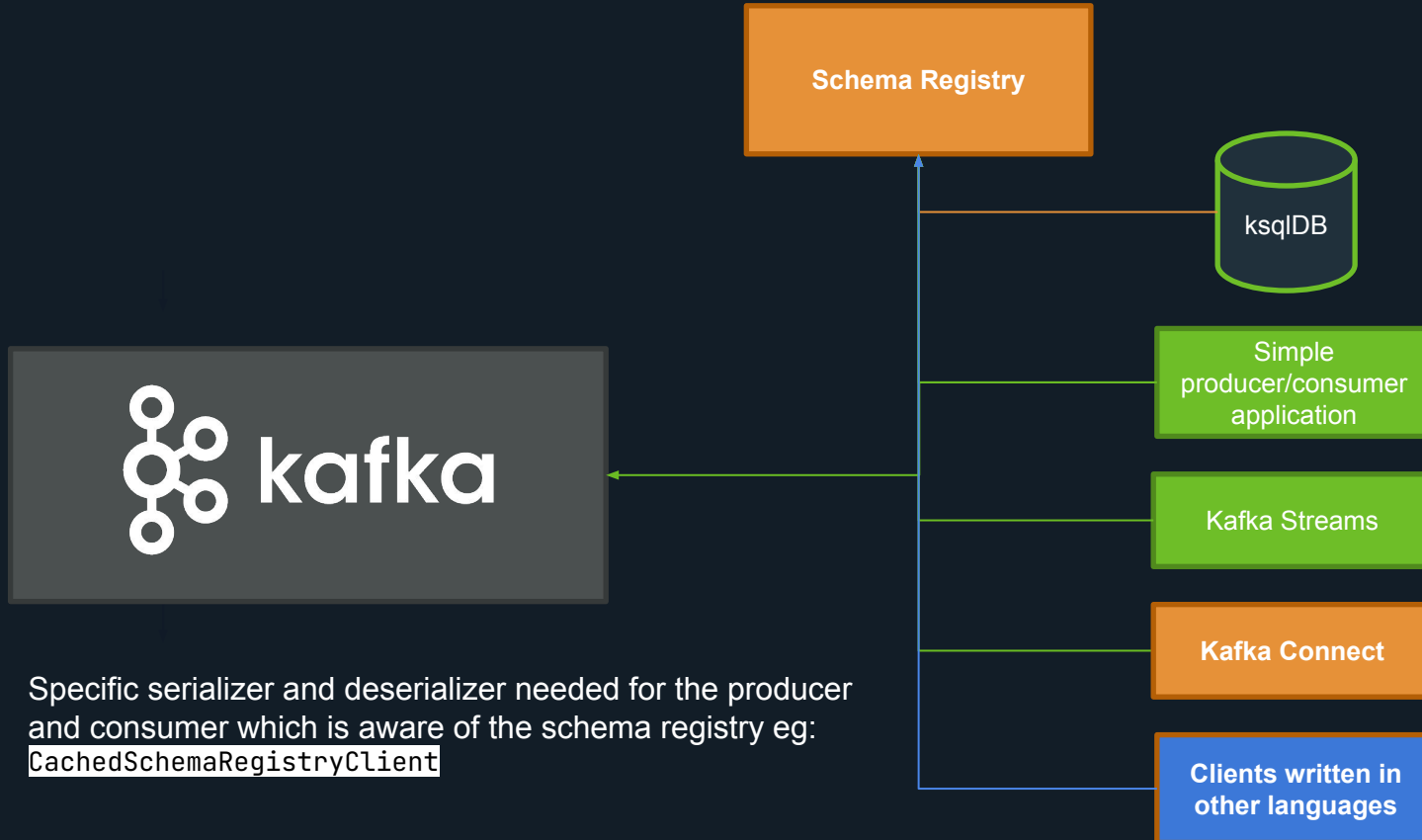
Schema Registry



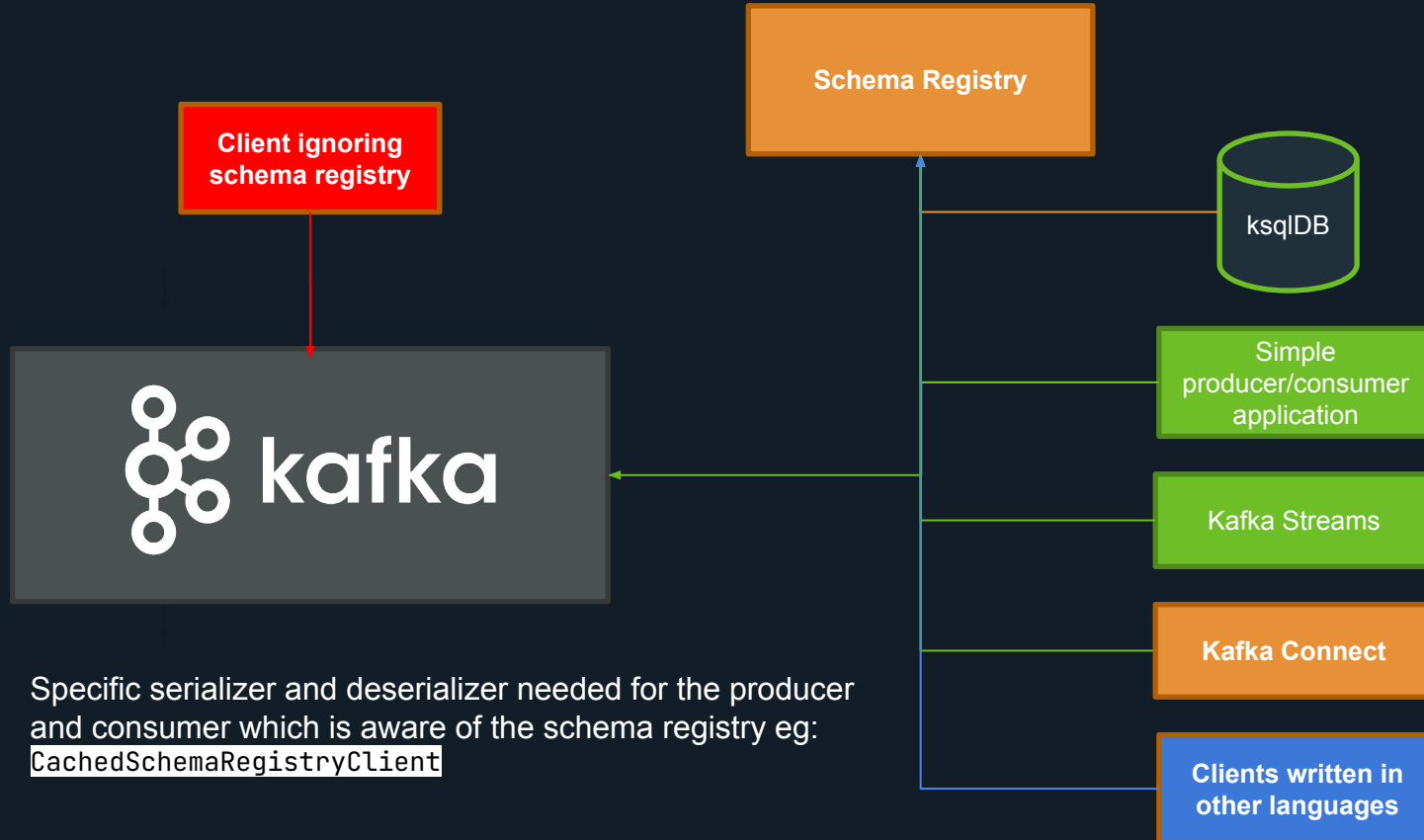
Specific serializer and deserializer needed for the producer and consumer which is aware of the schema registry eg:

`CachedSchemaRegistryClient`

Schema Registry



Schema Registry



KSQLDB

ksqlDB is a database that's purpose-built for stream processing applications.

It consolidates the many components found in virtually every stream processing architecture.

ksqlDB aims to provide one mental model for doing everything you need. You can build a complete streaming app against ksqlDB, which in turn has just one dependency: Apache Kafka.

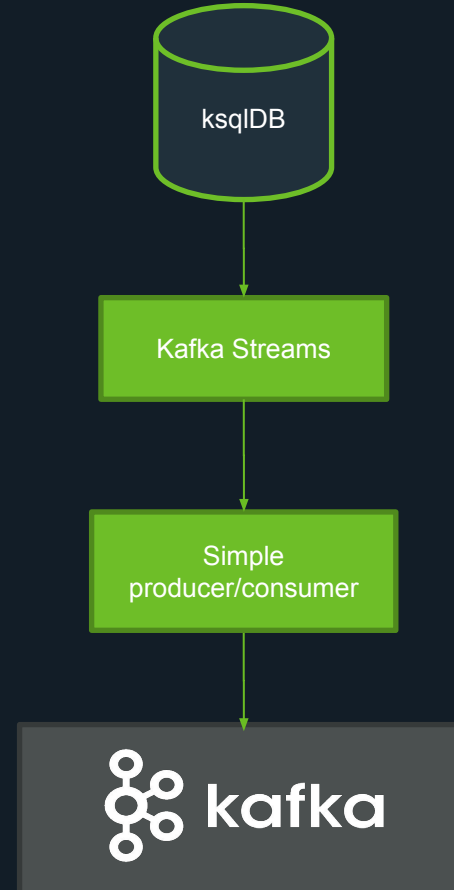


Kafka Streams vs ksqlDB

ksqlDB is actually a Kafka Streams application, meaning that ksqlDB is a completely different product with different capabilities, but uses **Kafka Streams** internally. Hence, there are both similarities and differences.

ksqlDB: actual service

Kafka Streams: client library



Key concepts: Streams & Tables

STREAM
Stateless
Insert
Aggregate: sum
Produce records

TABLE
Stateful
Update by key
Aggregate: replace
Materialized

Stream-Table duality:

Essentially, this means that a stream can be viewed as a table, and a table can be viewed as a stream.

KEY and PRIMARY KEY

	STREAM	TABLE
Key column type	KEY	PRIMARY KEY
NULLABLE key	YES	messages with NULL PRIMARY KEY are ignored
UNIQUE KEY	NO	YES - existing message with same KEY replaced
Tombstones	NO - messages with null value are ignored	YES - existing message with same KEY deleted

Push and Pull queries with Windows

Pull

- “Snapshot”
- NOT PERSISTED

Push

- “RealTime”
- NOT PERSISTED
- EMIT CHANGES
 - Continuous
- EMIT FINAL
 - End of the window

Window

- HOPPING window
- TUMBLING window
- SESSION window
- ~~SLIDING~~ window
- WITHIN and GRACE PERIOD

SELECT STATEMENTS

Stream time

Scenario: Event time added by producer



Out-of-order: event-time < stream-time

Solution: grace period

Stream and Table Joins

A **join** operation merges two input streams and/or tables based on the keys of their data records, and yields a new stream/table.

Primary	Secondary	Inner Join	Left Join	Outer Join	Result
KStream	KStream	supported	supported	supported	new topic
KTable	KTable	supported	supported	supported	new table
KStream	KTable	supported	supported	-	new topic
KStream	Global KTable	supported	supported	-	new topic

ksqlDB - Tooling

- **UDF** - User defined functions
 - functions written in Java and added to Kafka as a “plugin”
- **REST API**
 - /query REST Endpoint to manipulate the “database”
- **ksqlDB CLI**
 - command line tool
- **Java client library**
- **Embedded Connect Platform**
 - Connectors for systems which does not have producer or consumer
 - runs connectors on itself

Show me the code!

- Ecosystem setup: Schema Registry, ksqlDB
- Java - Spring Boot project setup
 - kafka-streams
- Demo



Want to learn more?

- Kafka Connect
- SpringFlo
- Spring Cloud libraries
- <https://kafka.apache.org/quickstart>
- <https://github.com/ttimot24/webinar-kafka-demo>

Questions & Answers

c**o**mmsignia

