Securing Kafka Schema Registry in Java Microservices

Thessaloniki not-only Java Meetup Group

October, 23rd 2025

Today’s agenda

About me

Basic Terminology

Schema evolution best practices

Securing the Schema Registry using ACLs and RBAC

Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven

About me

Matthaios Stavrou

Custom Software Engineering Assoc Mgr
mstauroy@gmail.com

- 12+ years in software engineering & solution architecture across energy, telecom & EU public sector
- Passion for designing secure, scalable & integrated digital solutions
- Experience leading agile teams in Greece & India, fostering collaboration & growth
- Daily work with Java (Spring Boot), Kafka, gRPC & Cloud (Azure & AWS)
- Dual MSc in Computer Science & Digital Systems Security – blending tech depth with strategic thinking
- Outside of work, I enjoy gaming, playing saxophone, traveling & cinema

Today's agenda

About me

Basic Terminology

Schema evolution best practices

Securing the Schema Registry using ACLs and RBAC

Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven

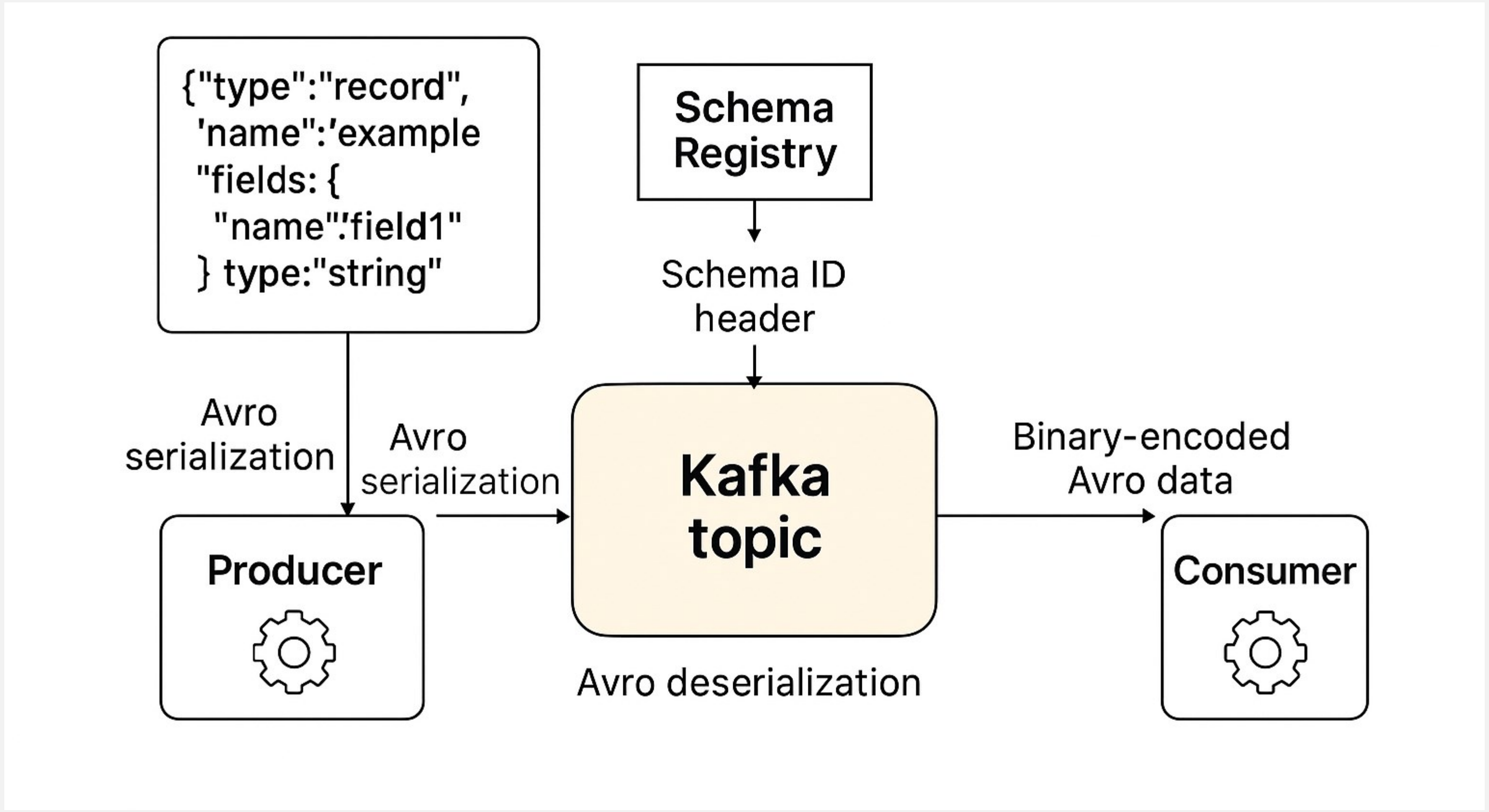
Basic Terminology (1/3)

What is Apache Kafka and Confluent

Basic Terms	Description
Apache Kafka	• A **distributed streaming platform** for building real-time data pipelines and event-driven applications. • It enables **high-throughput, low-latency** storage, processing, and consumption of event data.
Confluent	• Founded by the creators of Kafka, it provides the **Confluent Platform.** Enterprise-ready distribution of Kafka with additional features like **Schema Registry, Kafka Connect, ksqlDB**, security, and management tools.
Broker	• A Kafka server responsible for storing and serving messages. • A Kafka cluster consists of multiple brokers for scalability and fault tolerance.
Topic	• A logical channel for messages. • Topics are divided into **partitions** for parallel processing and better performance.
Producer	• Publishes **messages** (events) to a topic.
Consumer	• Reads messages from a topic. Consumers belong to **consumer groups** for load-balanced consumption.
Schema Registry	• A central service for **managing and storing Avro or JSON schemas**, ensuring **compatibility** and **validation** of messages.

Basic Terminology (2/3)

What is Avro and how it works



Why Avro Schema matters?

- ✓ Ensures producers and consumers share a common, versioned data contract.
- ✓ Supports schema evolution with compatibility checks (e.g., backward, forward).
- ✓ Reduces payload size due to binary serialization instead of raw JSON.

Basic Terminology (3/3)

JSON	VS	Avro
👎 **Format** Text based (human readable, slower parsing)		👍 **Format** Binary (compact, efficient)
👎 **Schema Definition** JSON (embedded in messages or standalone)		👍 **Schema Definition** JSON (separate .avsc file)
👎 **Message Size** Larger (verbose text)		👍 **Message Size** Small (compressed binary)
👎 **Performance** Slower, larger payload size		👍 **Performance** High throughput, low latency
👎 **Integration** Supported but less efficient in Kafka		👍 **Integration** Native with Kafka + Schema Registry
👎 **Use Case** Debugging, APIs, interoperability		👍 **Use Case** Event streaming, microservices, big data

Today's agenda

About me

Basic Terminology

Schema evolution best practices

Securing the Schema Registry using ACLs and RBAC

Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven

Schema evolution best practices

For Avro and JSON Schema

- Always register schemas in Schema Registry – never hardcode them. 01
- Define compatibility rules (usually BACKWARD or FORWARD or FULL) and enforce them at runtime. 02
- Use aliases in Avro for renaming fields safely. 03
- Provide default values for new optional fields. 04
- Keep schemas versioned and documented (include changelogs). 05
- Educate teams: Schema is a data contract, not just documentation. 06

Today’s agenda

About me

Basic Terminology

Schema evolution best practices

Securing the Schema Registry using ACLs and RBAC

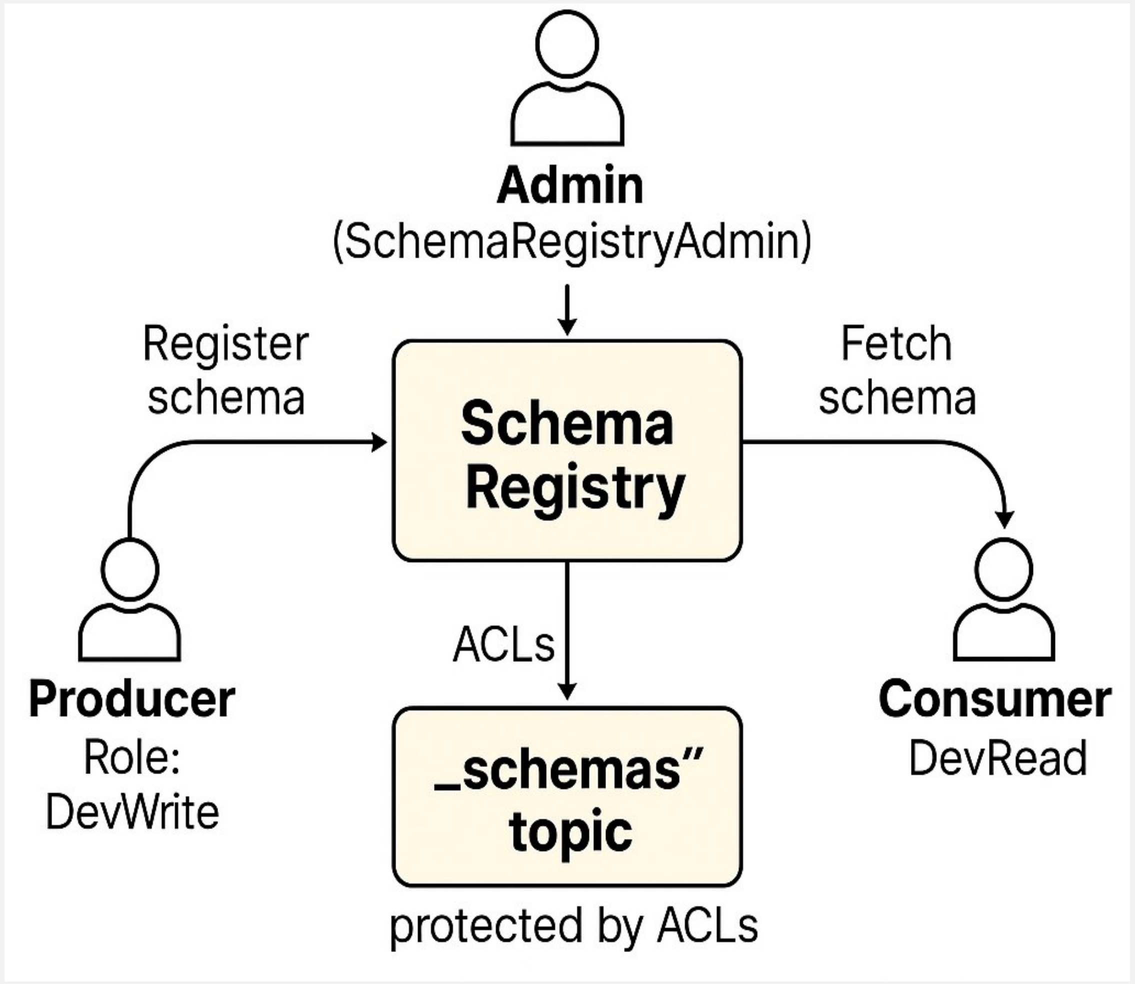
Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven

Securing the Schema Registry using ACLs and RBAC

Why secure it? → Prevent unauthorized schema changes, schema poisoning, and data contract breaks.



ACLs (Access Control Lists)

- Work at the Kafka level.
- Protect the internal topic _schemas (only Schema Registry can write).
- Define which service accounts can READ/WRITE/DESCRIBE Kafka resources.

RBAC (Role-Based Access Control)

- Works at the Schema Registry API level.
- Assigns roles to service accounts instead of ad-hoc permissions.
- Example roles:
  - SchemaRegistryAdmin → full admin rights.
  - DeveloperWrite → register new schemas.
  - DeveloperRead → fetch schemas only.

Today's agenda

About me

Basic Terminology

Schema evolution best practices

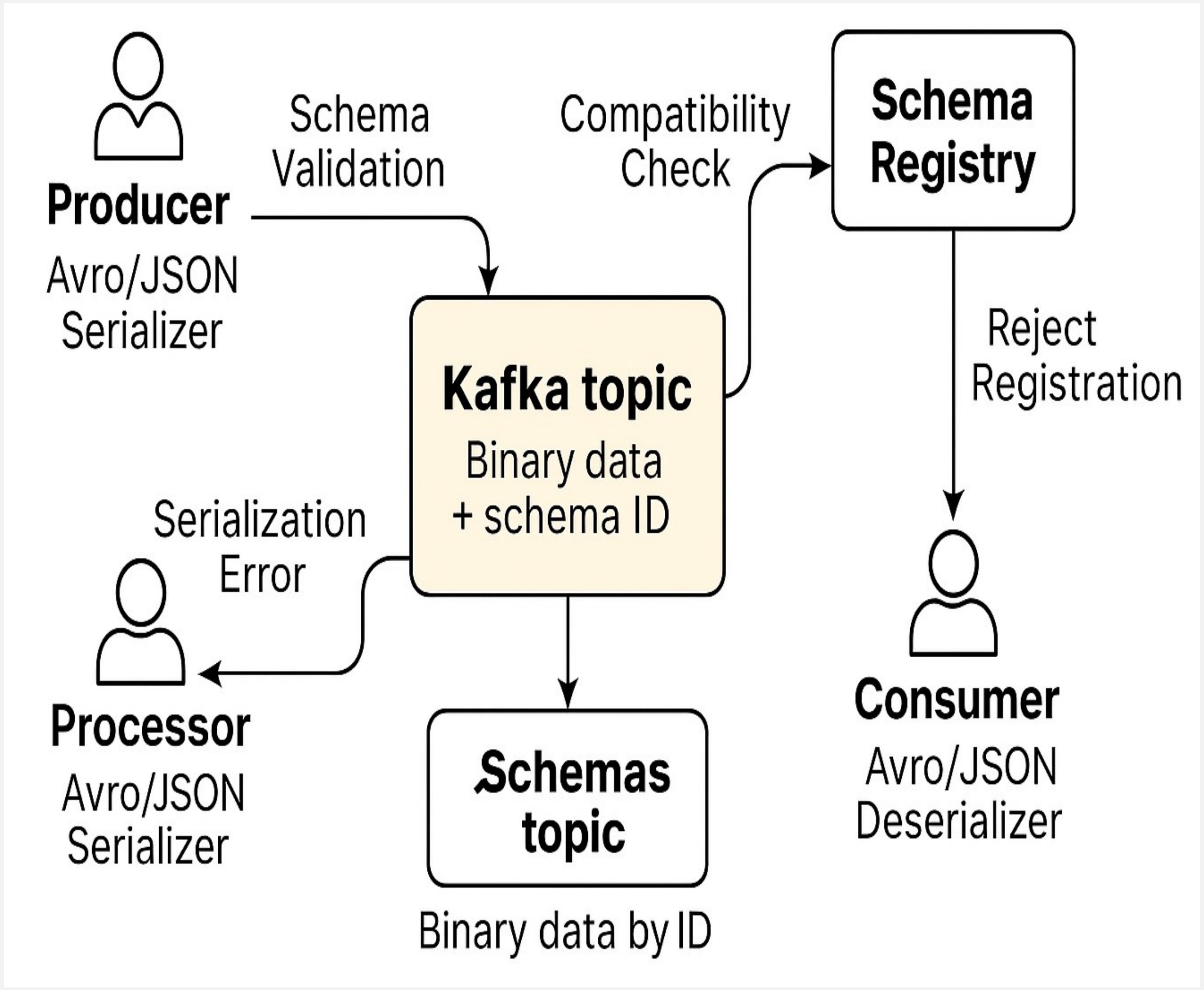
Securing the Schema Registry using ACLs and RBAC

Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven

Enforcing validation and compatibility at runtime



Producer-side validation

- The Avro/JSON/Protobuf serializer validates the record against the schema before sending to Kafka.
- If a required field is missing or a type is invalid → serialization error (message is rejected).

Consumer-side validation

- The deserializer reads the Schema ID from the message and fetches the schema from the Registry.
- If the schema is incompatible with the consumer's expected type → deserialization error.

Schema Registry compatibility checks

- When a new schema version is registered, the Registry enforces the configured compatibility mode (BACKWARD, FORWARD, FULL).
- If the new schema breaks compatibility, the registration is rejected immediately.

Today's agenda

About me

Basic Terminology

Schema evolution best practices

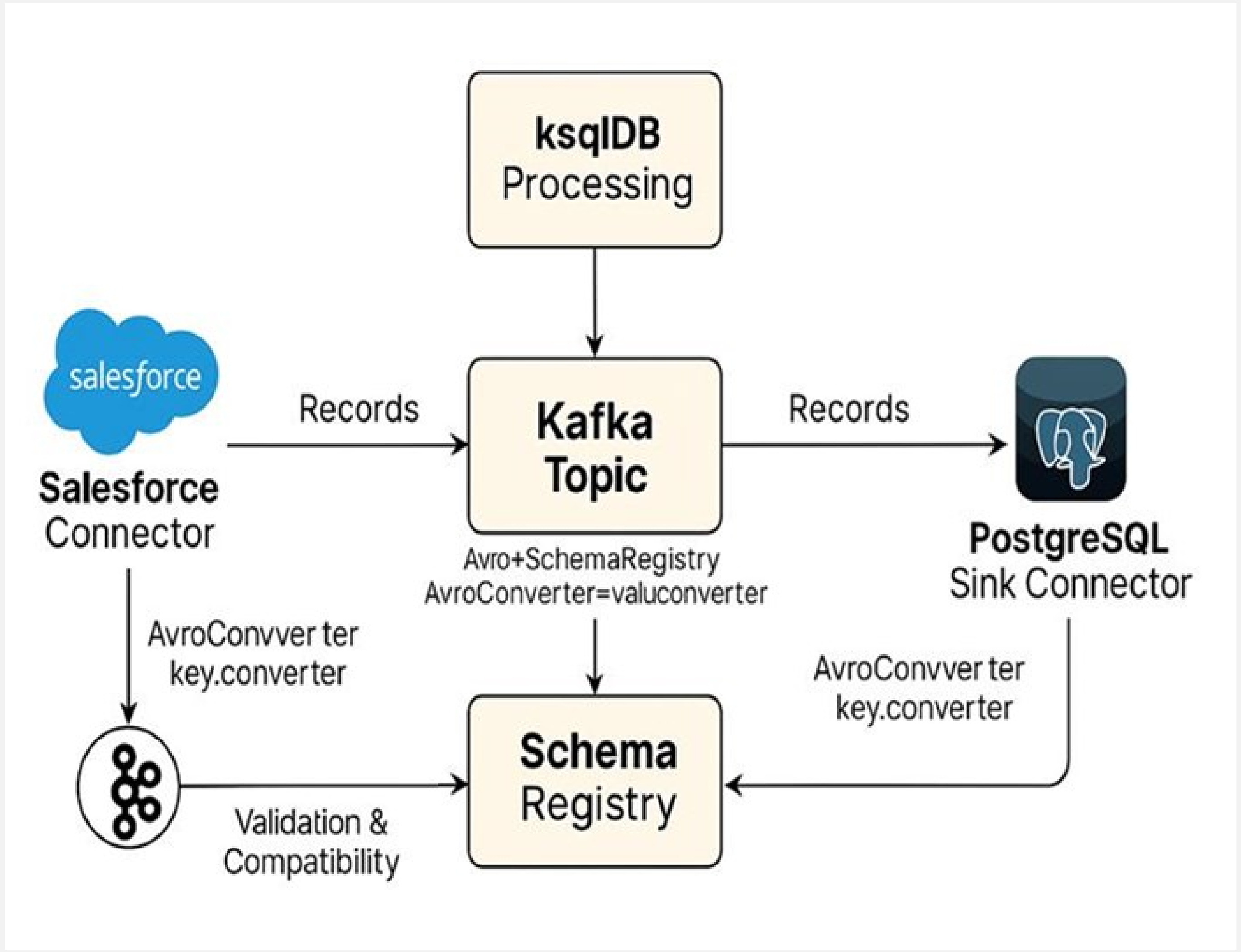
Securing the Schema Registry using ACLs and RBAC

Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven

Schema Registry in Action-Salesforce to PostgreSQL via Kafka



- Salesforce Source Connector captures Change Data Capture (CDC) events (account updates, new records).
- Each event is serialized using Avro and registered in Schema Registry (unique Schema ID per topic).
- Events flow into a Kafka topic (e.g., salesforce-change-events).
- ksqlDB processes and transforms the stream in real time (SQL over Kafka topics). Example: filter events, enrich data, or join with other streams.
- Processed events are published to an output topic (e.g., customer-updates).
- A PostgreSQL Sink Connector consumes the output topic and reflects changes into the PostgreSQL database.
- Schema Registry ensures compatibility and validation across all components (connectors, ksqlDB, consumers).

Today’s agenda

About me

Basic Terminology

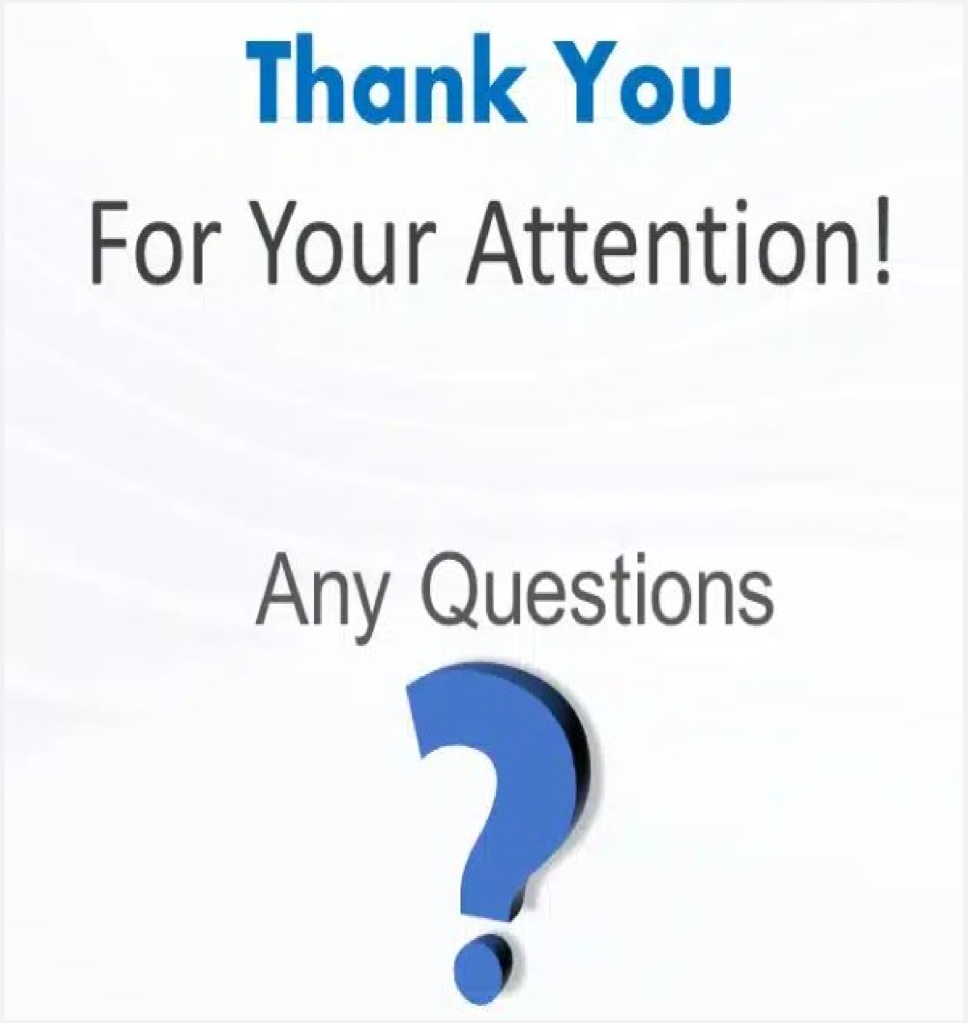
Schema evolution best practices

Securing the Schema Registry using ACLs and RBAC

Enforcing validation and compatibility at runtime

Schema Registry in Action-Salesforce to PostgreSQL via Kafka

Code examples with Spring Boot and Maven



Confluent

Dimitris Kontokostas

Eleftherios Chrysochoidis