Kafka with AVRO and Schema Registry

Kafka Applications using Confluent's Schema Registry

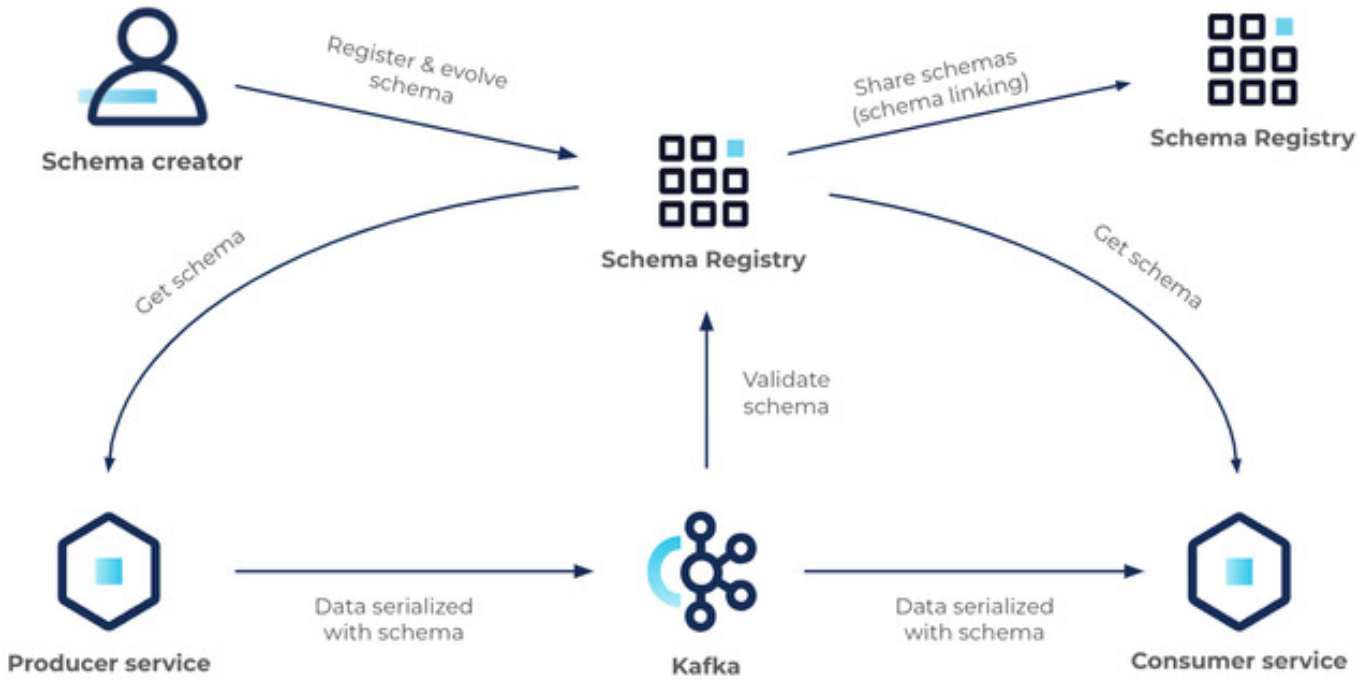
IBM Hunddle

14-07-2023

Introduction to Schema Registry

What is Schema Registry?

- Centralized repository for managing and validating schemas, essential in data processing and serialization.
- A schema defines the structure of data, including data types, format, relationships, and constraints.
- Schemas are utilized in data serialization, transforming data structures into a format that can be transmitted or stored.



Schema Registry Role

- Blueprint for data, describing data record structures, data field types, field relationships, and data rules.
- Integral in data serialization, defining the serialized data format, and validating the data during deserialization.

Benefits of Schema Registry

- **Data Validation**: Ensures the accuracy and quality of data by validating against defined schemas.
- **Compatibility Checking**: Checks compatibility of data against defined schemas, preventing issues.
- **Versioning and Evolution**: Supports changes and growth of schemas over time.
- **Simplified Development**: Eases the development and maintenance of data pipelines.
- **Risk Mitigation**: Reduces risk of data compatibility issues, corruption, and loss.

Services provided by Schema Registry

- **Well-defined Data Contract**: Facilitates communication between producers and consumers through well-defined data contract.
- **Schema Evolution Control**: Manages schema changes with explicit compatibility rules.
- **Payload Optimization**: Optimizes data transmission by passing a schema ID instead of the entire schema definition.

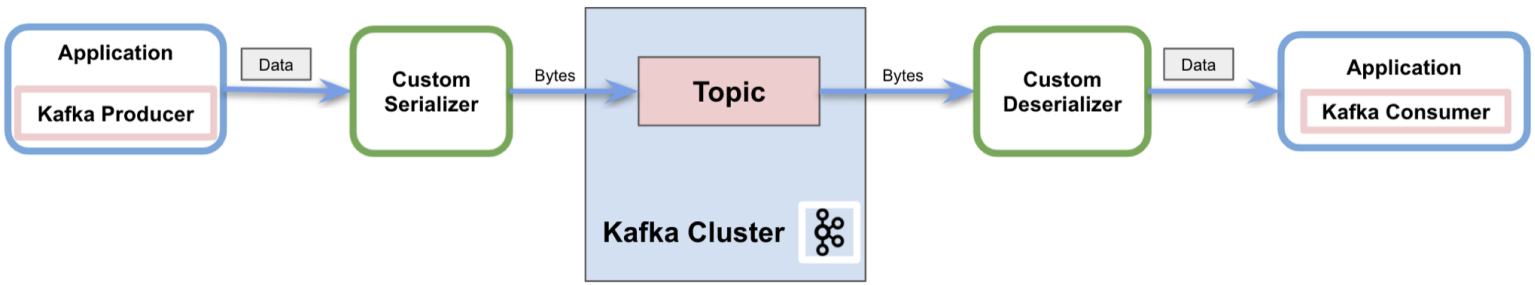
Common Problems Solved by Schema Registry

- **Data Inconsistency**: Ensures data adheres to schemas, reducing data inconsistency and increasing data quality.
- **Incompatible Data Formats**: Solves format issues by centralizing schema management and validation.
- **Schema Evolution**: Manages schema changes over time, preventing compatibility issues between different schema versions.
- **Schema Validation**: Checks data conformity to standard formats, mitigating data loss or corruption.
- **Data Governance**: Streamlines governance with a central location for managing and versioning data schemas.

Integrate Schema Registry with Kafka application in Java

Serialization and Deserialization in Kafka

Serialization is the process of converting objects into bytes. Deserialization is the inverse process — converting a stream of bytes into an object. In a nutshell, it transforms the content into readable and interpretable information.



Custom Serializers

Apache Kafka provides a pre-built serializer and deserializer for several basic types:

- StringSerializer
- ShortSerializer
- IntegerSerializer
- LongSerializer
- DoubleSerializer
- BytesSerializer

Avro serializer

You can plug `KafkaAvroSerializer` into `KafkaProducer` to send messages of Avro type to Kafka sending data of other types to `KafkaAvroSerializer` will cause a `SerializationException`

```
Properties props = new Properties();

props.put(
  ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
  org.apache.kafka.common.serialization.StringSerializer.cl
);

props.put(
  ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
  io.confluent.kafka.serializers.KafkaAvroSerializer.class
);
```

Avro deserializer 1/2

You can plug in `KafkaAvroDeserializer` to `KafkaConsumer` to receive messages of any Avro type from Kafka. In the following example, messages are received with a key of type string and a value of type Avro record from Kafka. When getting the message key or value, a `SerializationException` may occur if the data is not well formed.

Avro deserializer 2/2

```
Properties props = new Properties();

props.put(
  ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
  org.apache.kafka.common.serialization.StringDeserializer
);

props.put(
  ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
  io.confluent.kafka.serializers.KafkaAvroDeserializer
);
```

Spring Kafka consumer

```
spring.kafka.consumer.key-deserializer=
 org.apache.kafka.common.serialization.StringDeserializer
spring.kafka.consumer.value-deserializer=
 io.confluent.kafka.serializers.KafkaAvroDeserializer
```

Spring Kafka producer

```
spring.kafka.producer.key-serializer=
 org.apache.kafka.common.serialization.StringSerializer
spring.kafka.producer.value-serializer=
 io.confluent.kafka.serializers.KafkaAvroSerializer
```

Spring Kafka (yaml)

```yaml
spring:
  kafka:
    consumer:
      key-deserializer: org.apache.kafka.common.serializati
      value-deserializer: io.confluent.kafka.serializers.Ka
    producer:
      key-serializer: org.apache.kafka.common.serialization
      value-serializer: io.confluent.kafka.serializers.Kaf
```

Questions and Discussion

- Open the floor for questions and discussions
- Share experiences, challenges, and best practices

Thank You

- Thank you for your time and attention
- Good luck with your kafka applications!