Kafka Integration Testing

Integration Testing for Kafka Applications using Confluent's Schema Registry

Luis Castillo

17-04-2023

Introduction

- Welcome fellow developers!
- Today's presentation focuses on integration testing for Kafka applications using Confluent's Schema Registry
- We'll cover isolated testing with WireMock and Embedded Kafka in JUnit

Overview

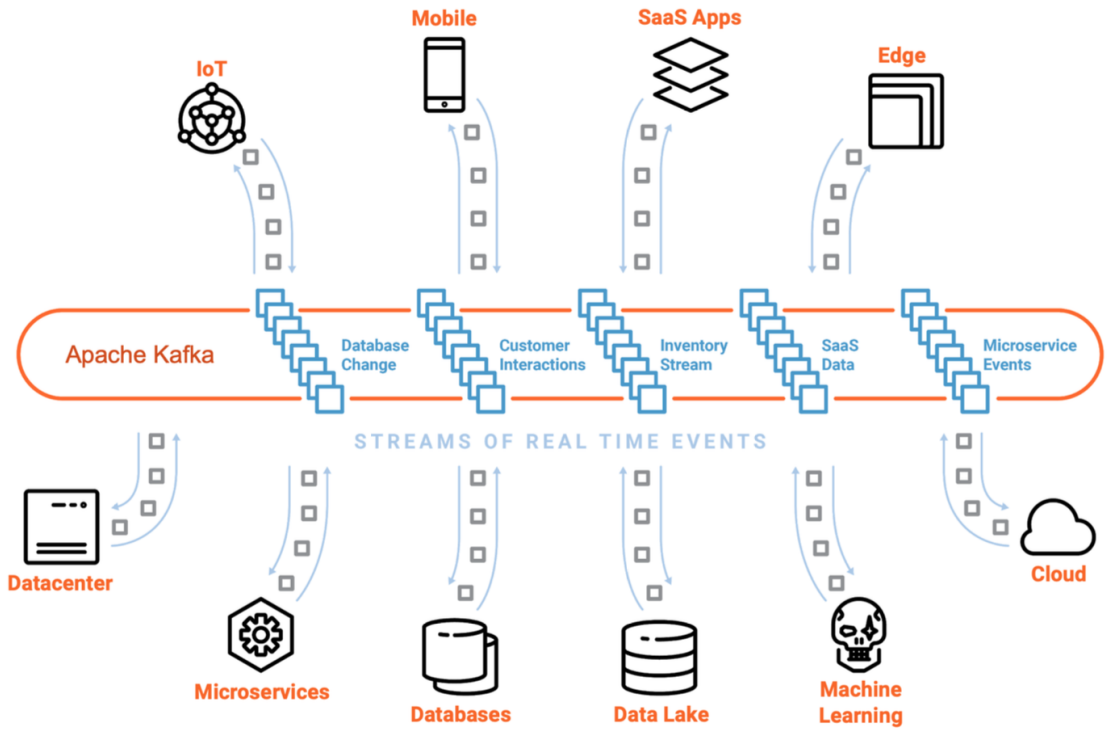
- What is integration testing?
- Importance of testing Kafka applications
- Confluent's Schema Registry and its role
- WireMock and Embedded Kafka for isolated testing

Integration Testing

- **Definition**: Integration testing verifies that different components of a system work together as expected
- **Aim**: To identify issues that may arise during component interaction
- Helps catch potential incompatibilities early in the development process

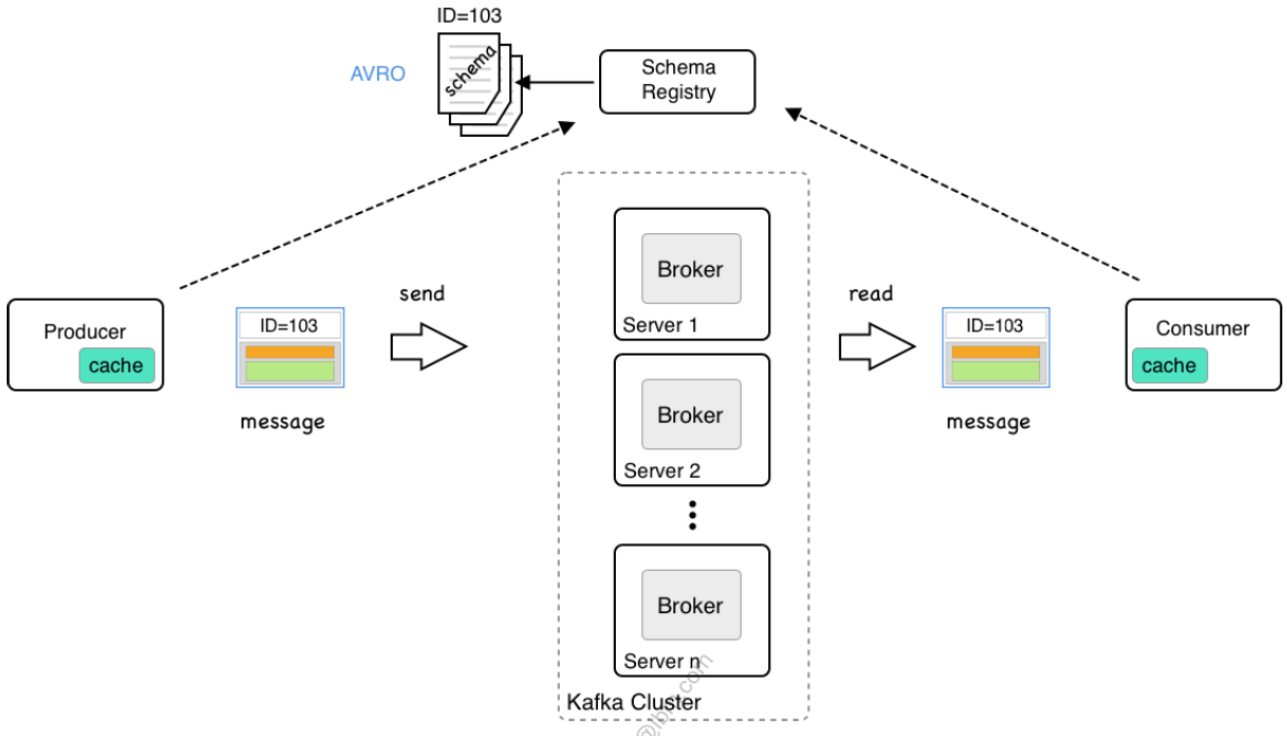
Kafka Applications and Testing

- Kafka applications often involve complex data processing and multiple microservices
- Ensuring data consistency and correctness is essential
- Integration testing helps validate correct message handling, serialization/deserialization, and error handling



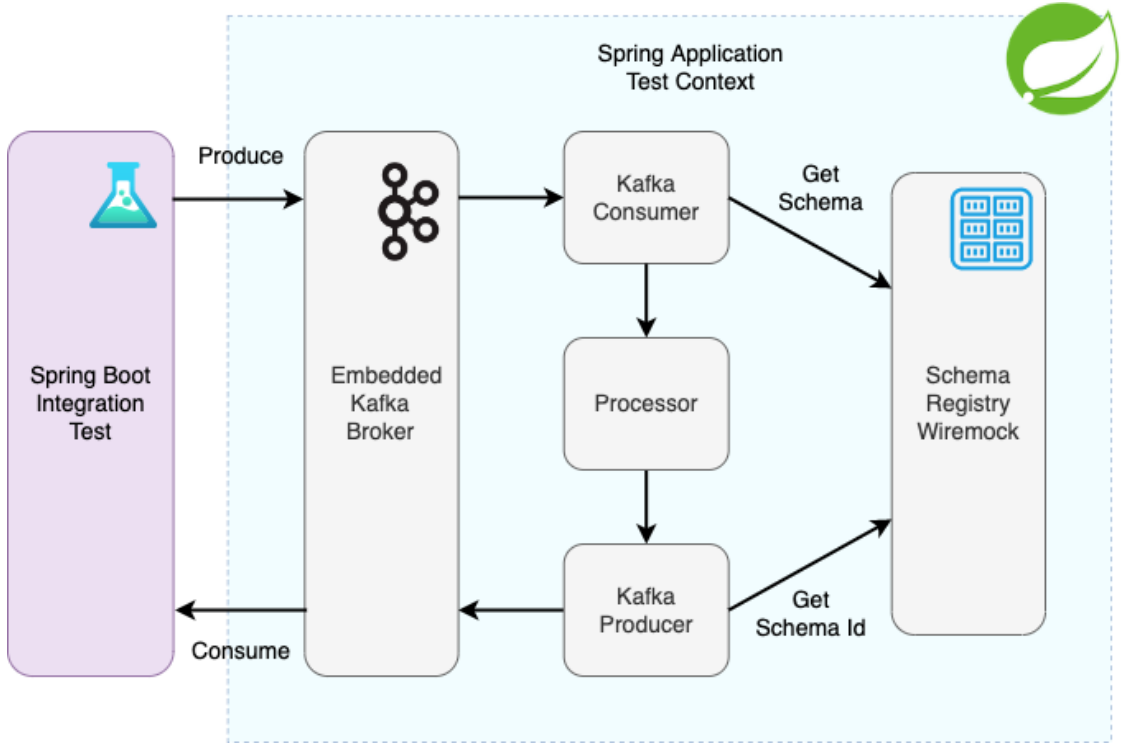
Confluent's Schema Registry

- Centralized storage and management of Avro schemas
- Ensures compatibility and consistency between producers and consumers
- Provides schema evolution support



WireMock and Embedded Kafka

- **WireMock**: HTTP service mocking library
- **Embedded Kafka**: In-memory Kafka broker and Zookeeper for testing
- Allow for isolated, controlled, and reproducible testing environments



Setting Up WireMock and Embedded Kafka

- **Step 1**: Add dependencies to your build configuration
- **Step 2**: Set up Embedded Kafka and WireMock in your test class
- **Step 3**: Configure Kafka producer and consumer to use the embedded broker and schema registry

Writing Integration Tests

- **Step 1**: Define the message schema
- **Step 2**: Set up WireMock to stub Schema Registry requests
- **Step 3**: Write producer and consumer tests
- **Step 4**: Validate message handling, serialization/deserialization, and error handling

Running and Debugging Tests

- Use JUnit to run tests and validate expected results
- Use spring-kafka-test utilities to validate the consumer/producer behavior
- Leverage logging and debugging tools to diagnose issues

Conclusion

- Integration testing is crucial for maintaining reliable Kafka applications
- Confluent's Schema Registry ensures compatibility and consistency between components
- WireMock and Embedded Kafka provide a controlled, isolated testing environment
- Write tests to verify message handling, serialization/deserialization, and error handling

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 integration testing efforts!