Apache Kafka with Spring Boot- 2021-2022

**Basic Kafka with Spring Boot Microservice**

In this example, we will have two different microservices with Kafka, one microservice will publish and another microservice will listen or consume. One is called **Producer** Microservice and another is **Consumer** microservice.

**Producer Project Structure**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0*

*https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>basic-kafka-springboot-producer</groupId>

<artifactId>basic-kafka-springboot-producer</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>basic-kafka-springboot-producer</name>

<description>basic-kafka-springboot-producer</description>

**<parent>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-parent</artifactId>**

**<version>2.4.9</version>**

**<relativePath />**

**</parent>**

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.kafka</groupId>**

**<artifactId>spring-kafka</artifactId>**

**</dependency>**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

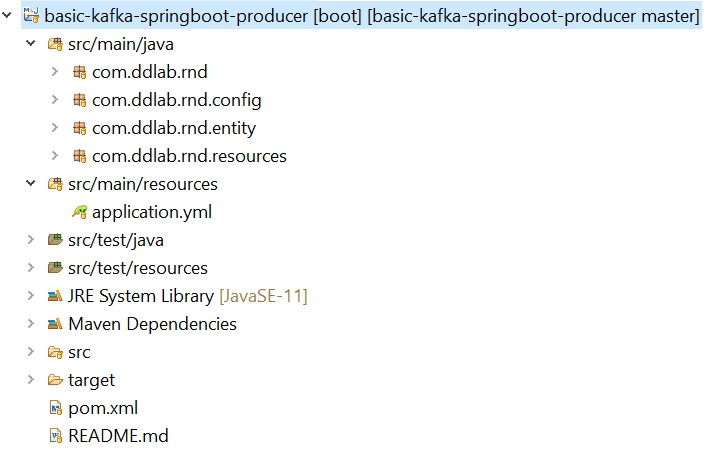
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>



**application.yml**

**Entity Layer**

**import** lombok.Data;

@Data

**public** **class** Employee {

**private** **int** id;

**private** String name;

}

server:

port: 8081

kafka:

bootstrapAddress: localhost:9092

emp.topic.name: emp-kafka-topic-1

str.topic.name: str-kafka-topic-2

Spring Boot Main Application

@SpringBootApplication

**public** **class** KafkaProducerApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.run(KafkaProducerApplication.**class**, args);

}

}

**Kafka Configuration Class**

@Configuration

**public** **class** KafkaProducerConfig {

@Value(value = "${kafka.bootstrapAddress}")

**private** String bootstrapAddress;

// 1. Send string to Kafka

@Bean

**public** ProducerFactory<String, String> producerFactory() {

Map<String, Object> props = **new** HashMap<>();

props.put(ProducerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, bootstrapAddress);

props.put(ProducerConfig.***KEY\_SERIALIZER\_CLASS\_CONFIG***, StringSerializer.**class**);

props.put(ProducerConfig.***VALUE\_SERIALIZER\_CLASS\_CONFIG***, StringSerializer.**class**);

**return** **new** DefaultKafkaProducerFactory<>(props);

}

@Bean

**public** KafkaTemplate<String, String> kafkaTemplate() {

**return** **new** KafkaTemplate<>(producerFactory());

}

// 2. Send Employee objects to Kafka

@Bean

**public** ProducerFactory<String, Employee> empProducerFactory() {

Map<String, Object> configProps = **new** HashMap<>();

configProps.put(ProducerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, bootstrapAddress);

configProps.put(ProducerConfig.***KEY\_SERIALIZER\_CLASS\_CONFIG***, StringSerializer.**class**.getName());

configProps.put(ProducerConfig.***VALUE\_SERIALIZER\_CLASS\_CONFIG***, JsonSerializer.**class**.getName());

**return** **new** DefaultKafkaProducerFactory<>(configProps);

}

@Bean

**public** KafkaTemplate<String, Employee> empKafkaTemplate() {

**return** **new** KafkaTemplate<>(empProducerFactory());

}

}

**Controller Layer**

@RestController

**public** **class** ProducerController {

@Value(value = "${kafka.emp.topic.name}")

**private** String objTopicName;

@Value(value = "${kafka.str.topic.name}")

**private** String strTopicName;

@Autowired

**private** KafkaTemplate<String, Employee> objKafkaTemplate;

@Autowired

**private** KafkaTemplate<String, String> stringKafkaTemplate;

@PostMapping(path = "/publish/obj", consumes = MediaType.***APPLICATION\_JSON\_VALUE***, produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String postObjectMessage(@RequestBody Employee emp) {

System.***out***.println("Employee : " + emp);

objKafkaTemplate.send(objTopicName, emp);

**return** "Employee Object Published Successfully";

}

@PostMapping(path = "/publish/str", consumes = MediaType.***TEXT\_PLAIN\_VALUE***, produces = MediaType.***TEXT\_PLAIN\_VALUE***)

**public** String postStringMessage(@RequestBody String msg) {

System.***out***.println("String msg : " + msg);

stringKafkaTemplate.send(strTopicName, msg);

**return** "String Published Successfully";

}

}

The server will be started in 8081.

**Consumer Microservice with Kafka**

The project structure, pom.xml, entity and spring boot application will have similar structure and nature.

**Service Layer with Consumer application**

@Service

**public** **class** KafkaConsumerListener {

@KafkaListener(topics = "emp-kafka-topic-1", groupId = "group\_json1",

containerFactory = "empKafkaListenerContainerFactory")

**public** **void** listenWithHeaders(@Payload Employee emp, @Header(KafkaHeaders.***RECEIVED\_PARTITION\_ID***) **int** partition) {

System.***out***.println("Received Message: \n" + emp + "\nfrom partition: " + partition);

System.***out***.println("Employee Id: " + emp.getId());

System.***out***.println("Employee Id: " + emp.getName());

}

@KafkaListener(topics = "str-kafka-topic-2", groupId = "group\_json1",

containerFactory = "strKafkaListenerContainerFactory")

**public** **void** listenStringMsg(@Payload String msg, @Header(KafkaHeaders.***RECEIVED\_PARTITION\_ID***) **int** partition) {

System.***out***.println("Received Message: \n" + msg + "\nfrom partition: " + partition);

}

}

**Kafka Consumer Configuration**

@EnableKafka

@Configuration

**public** **class** KafkaConsumerConfig {

@Value(value = "${kafka.bootstrapAddress}")

**private** String bootstrapAddress;

// 2. Consume json data from Kafka

**public** ConsumerFactory<String, Employee> userConsumerFactory() {

Map<String, Object> props = **new** HashMap<>();

props.put(ConsumerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, bootstrapAddress);

props.put(ConsumerConfig.***GROUP\_ID\_CONFIG***, "group\_json1");

props.put(JsonDeserializer.***TRUSTED\_PACKAGES***, "\*");

**return** **new** DefaultKafkaConsumerFactory<>(props,

**new** StringDeserializer(),

**new** JsonDeserializer<>(Employee.**class**));

}

@Bean

**public** ConcurrentKafkaListenerContainerFactory<String, Employee> empKafkaListenerContainerFactory() {

ConcurrentKafkaListenerContainerFactory<String, Employee> factory =

**new** ConcurrentKafkaListenerContainerFactory<>();

factory.setConsumerFactory(userConsumerFactory());

**return** factory;

}

// 1. Consume string data from Kafka

@Bean

**public** ConsumerFactory<String, String> strConsumerFactory() {

Map<String, Object> props = **new** HashMap<>();

props.put(ConsumerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, bootstrapAddress);

props.put(ConsumerConfig.***GROUP\_ID\_CONFIG***, "group\_json2");

props.put(ConsumerConfig.***KEY\_DESERIALIZER\_CLASS\_CONFIG***,

StringDeserializer.**class**);

props.put(ConsumerConfig.***VALUE\_DESERIALIZER\_CLASS\_CONFIG***,

StringDeserializer.**class**);

**return** **new** DefaultKafkaConsumerFactory<>(props);

}

@Bean

**public** ConcurrentKafkaListenerContainerFactory<String, String> strKafkaListenerContainerFactory() {

ConcurrentKafkaListenerContainerFactory<String, String> factory

= **new** ConcurrentKafkaListenerContainerFactory<>();

factory.setConsumerFactory(strConsumerFactory());

**return** factory;

}

}

application.yml

server:

port: 8082

kafka:

bootstrapAddress: localhost:9092

The above both producer and consumer are the old way of achieving Kafka integration. In the next version, it has been changed as Kafka Cloud Stream.

Spring Cloud Kafka Stream

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0*

*https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>spring-cloud-stream-publisher1</groupId>

<artifactId>spring-cloud-stream-publisher1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-cloud-stream-publisher1</name>

<description>spring-cloud-stream-publisher1</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

**<version>2.2.2.RELEASE</version>**

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

**<java.version>11</java.version>**

**<spring-cloud.version>Hoxton.RELEASE</spring-cloud.version>**

</properties>

**<dependencyManagement>**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>${spring-cloud.version}</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

**</dependencies>**

**</dependencyManagement>**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-stream</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-stream-kafka</artifactId>**

**</dependency>**

<!-- Also install the Lombok plugin in your IDE -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

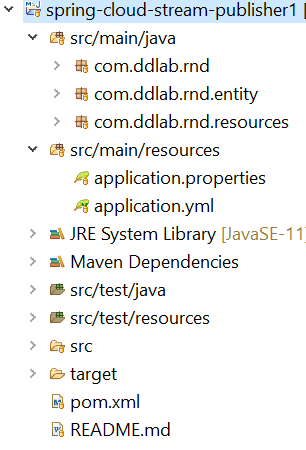
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>



**application.properties**

server.port=9192

**application.yml**

spring:

cloud:

stream:

kafka:

binder:

brokers: localhost:9092

bindings:

output:

destination: emp-kafka-topic-1

**Entity**

@Data

@AllArgsConstructor

@NoArgsConstructor

**public** **class** Employee {

**private** **int** id;

**private** String name;

}

**Spring Boot Application**

@SpringBootApplication

**public** **class** SpringCloudStreamPublisherApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringCloudStreamPublisherApplication.**class**, args);

}

}

**Controller**

**@EnableBinding(Source.class)**

@RestController

**public** **class** PublishController {

**@Autowired**

**private MessageChannel output;**

@PostMapping("/publish")

**public** Employee publishEvent(@RequestBody Employee emp) {

output.send(MessageBuilder.*withPayload*(emp).build());

**return** emp;

}

}

**Consumer Kafka Stream Application**

application.yml

spring:

cloud:

stream:

kafka:

binder:

brokers: localhost:9092

bindings:

input:

destination: emp-kafka-topic-1

Application.properties

server.port=8081

**Consumer Service**

**@EnableBinding(Sink.class)**

**public** **class** ConsumerService {

**@StreamListener("input")**

**public** **void** consumeMessage(Employee emp) {

System.out.println("Consume payload : " + emp);

System.out.println("Emp ID: "+emp.getId());

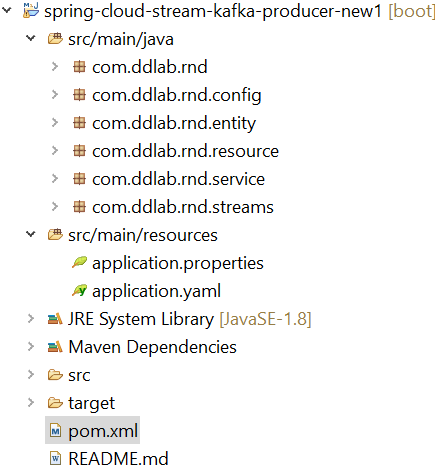
System.out.println("Emp Name: "+emp.getName());

}

}

Spring Cloud Kafka Stream New Spring Way

Here also we have Producer and Consumer as two Microservices, one will produce and another consume.



<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0*

*https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>spring-cloud-stream-kafka-producer-new1</groupId>

<artifactId>spring-cloud-stream-kafka-producer-new1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-cloud-stream-kafka-producer-new1</name>

<description>spring-cloud-stream-kafka-producer-new1</description>

**<parent>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-parent</artifactId>**

**<version>2.2.4.RELEASE</version>**

**<relativePath/>**

**</parent>**

<properties>

<java.version>1.8</java.version>

<!-- Spring Boot -->

<spring-cloud.version>Hoxton.RELEASE</spring-cloud.version>

</properties>

**<dependencyManagement>**

**<dependencies>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-dependencies</artifactId>**

**<version>${spring-cloud.version}</version>**

**<type>pom</type>**

**<scope>import</scope>**

**</dependency>**

**</dependencies>**

**</dependencyManagement>**

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-stream</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>org.springframework.cloud</groupId>**

**<artifactId>spring-cloud-starter-stream-kafka</artifactId>**

**</dependency>**

<!-- Also install the Lombok plugin in your IDE -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

</project>

**Spring Boot Application**

@SpringBootApplication

**public** **class** SpringCloudStreamPublisherApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.run(SpringCloudStreamPublisherApplication.**class**, args);

}

}

**Entity**

@Data

**public** **class** Employee {

**private** **int** id;

**private** String name;

}

**Controller**

@RestController

**public** **class** StreamPublishController {

@Autowired

PublisherService pubServie;

@PostMapping("/publish")

**public** String publishEvent(@RequestBody Employee emp) {

pubServie.sendEmpInfo(emp);

**return** "Message Posted Successfully";

}

}

**Service**

@Service

**public** **class** PublisherService {

@Autowired

**private** EmpStream empStream;

**public** **void** sendEmpInfo(**final** Employee emp) {

**MessageChannel messageChannel = empStream.outboundMessage();**

**boolean** sent = **messageChannel.send(MessageBuilder.*withPayload*(emp)**

**.setHeader(MessageHeaders.*CONTENT\_TYPE*, MimeTypeUtils.*APPLICATION\_JSON*)**

**.build());**

System.***out***.println("Message sent successfully ..."+sent);

}

}

**Stream Config (Important)**

**public** **interface** EmpStream {

String ***OUTPUT*** = "emp-out";

**@Output(*OUTPUT*)**

**MessageChannel outboundMessage();**

}

**Binding Configuration (Important)**

**@EnableBinding**(EmpStream.**class**)

**public** **class** StreamsConfig {

}

**application.yml**

spring:

cloud:

stream:

kafka:

binder:

brokers: localhost:9092

#brokers: rh8amq01:9092,rh8amq02:9092,rh8amq03:9092

auto-create-topics: true

configuration:

auto.offset.reset: latest

bindings:

emp-out:

destination: greetings-sample

contentType: application/json

application.properties

server.port = 9091

**Consumer Kakfa Stream Microservice**

The overall structure is similar to Producer Microservice. Consumer related code is given for consuming the event.

**Service**

@Component

**public** **class** ListenerService {

@StreamListener(EmpStream.***INPUT***)

**public** **void** handleGreetings(@Payload Employee emp, @Headers Map<String, Object> headers) {

System.***out***.println("Received greetings: "+emp);

System.***out***.println("Partition: "+headers.get(KafkaHeaders.***RECEIVED\_PARTITION\_ID***));

System.***out***.println("Offset: "+headers.get(KafkaHeaders.***OFFSET***));

System.***out***.println("Emp Name : "+emp.getName());

System.***out***.println("Emp Id: "+emp.getId());

}

}

**Stream Configuration**

**public** **interface** EmpStream {

String ***INPUT*** = "emp-in";

**@Input(*INPUT*)**

**SubscribableChannel inBoundMsg();**

}

**Binding Configuration**

**@EnableBinding(EmpStream.class)**

**public** **class** StreamsConfig {

}

**application.yml**

spring:

**application.properties**

server.port = 9092

cloud:

stream:

kafka:

binder:

brokers: localhost:9092

#brokers: rh8amq01:9092,rh8amq02:9092,rh8amq03:9092

auto-create-topics: true

configuration:

auto.offset.reset: latest

bindings:

emp-in:

destination: greetings-sample

group: emp-in-group

contentType: application/json