Step 1: Initialization producer:

Kafka producer service:

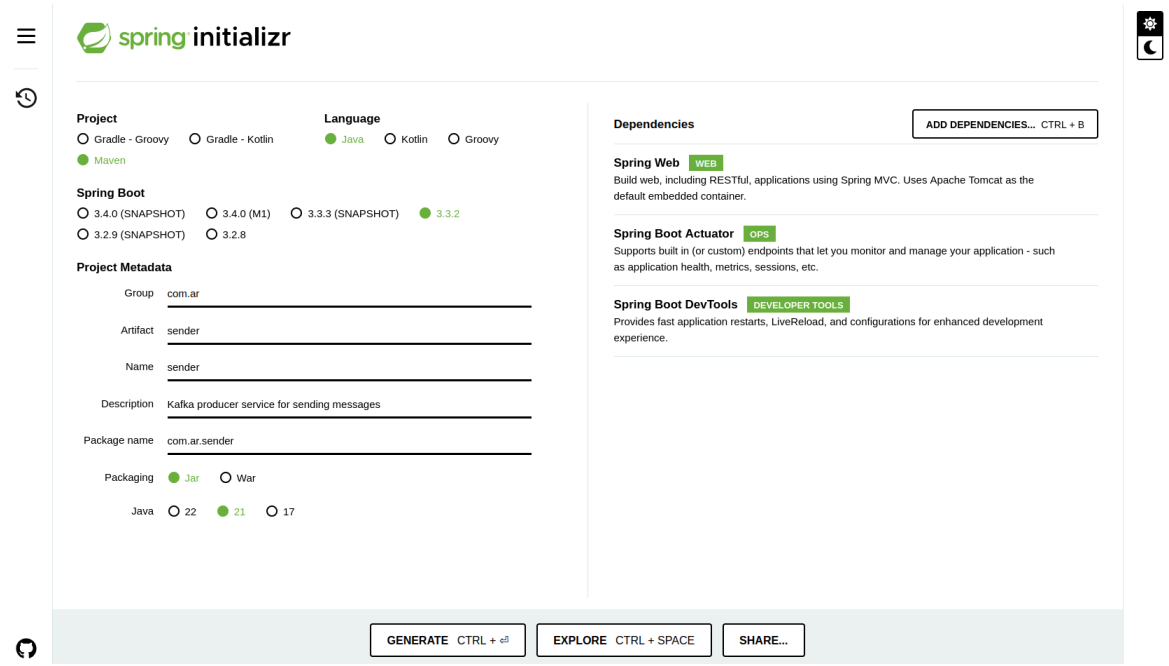
Project: Maven
Language: Java
Spring boot: 3.3.2

Group: com.ar
Artifact: sender
Name: sender
Description: Kafka producer service for sending messages
Package name: com.ar.sender

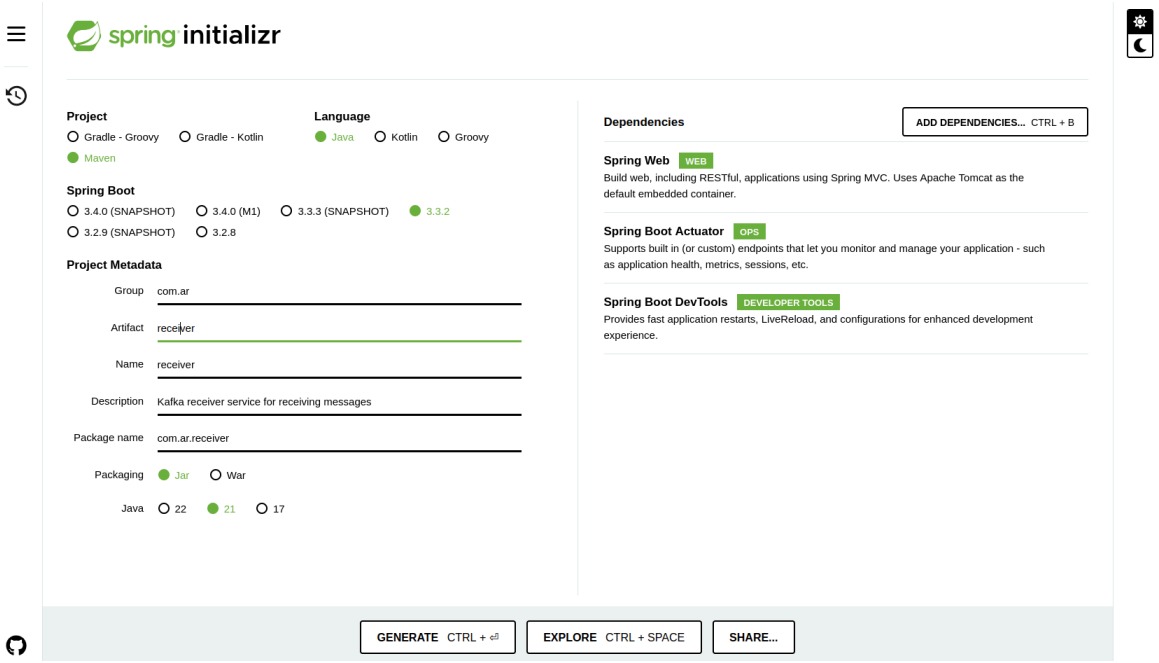
Packaging: Jar
Java: 21

Dependencies:

Spring Web
Spring Boot Actuator
Spring Boot DevTools



Step 2: Initialization consumer



Step 3: Writing producer service

Add dependencies of kafka clients and spring kafka:

```
<!-- Kafka -->
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka-clients</artifactId>
            <version>3.8.0</version>
        </dependency>

        <!--
https://mvnrepository.com/artifact/org.springframework.kafka/spring-kaf
ka -->
        <dependency>
            <groupId>org.springframework.kafka</groupId>
```

```
            <artifactId>spring-kafka</artifactId>
            <version>3.2.2</version>
        </dependency>
```

Write kafka producer configuration file and service file:

```
ChannelMessageProducerConfig.java
```

```
package com.ar.sender.kafka.producer.channel.message;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.producer.ProducerConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;

@Configuration
public class ChannelMessageProducerConfig {
   @Value("${kafka.bootstrap-servers}")
   private String bootstrapServers;

   @Value("${kafka.producer.key-serializer}")
   private String keySerializer;

   @Value("${kafka.producer.value-serializer}")
   private String valueSerializer;

   @Bean
   public ProducerFactory<String, String> producerFactory() {
       Map<String, Object> configProps = new HashMap<>();
       configProps.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
bootstrapServers);
       configProps.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
keySerializer);
```

```
        configProps.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
valueSerializer);
        return new DefaultKafkaProducerFactory<>(configProps);
    }

    @Bean
    public KafkaTemplate<String, String> kafkaTemplate() {
        return new KafkaTemplate<>(producerFactory());
    }
}
```

ChannelMessageProducerService.java

```
package com.ar.sender.kafka.producer.channel.message;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;

@Service
public class ChannelMessageProducerService {
    private static final Logger LOGGER =
LoggerFactory.getLogger(ChannelMessageProducerService.class);

    private final KafkaTemplate<String, String> kafkaTemplate;


    @Value("${kafka.topic}")
    private String topic;

    @Autowired
    public ChannelMessageProducerService(KafkaTemplate<String, String>
kafkaTemplate) {
        this.kafkaTemplate = kafkaTemplate;
```

```
    }

    public void sendMessage(String value) {
        kafkaTemplate.send(topic, "message", value);
        LOGGER.info("Sent message: key=message, value=" + value);
    }

    public void sendMessage(String key, String value) {
        kafkaTemplate.send(topic, key, value);
        LOGGER.info("Sent message: key=" + key + ", value=" + value);
    }

}
```

Specify properties details in application.yml:

```
kafka:
 bootstrap-servers: localhost:9092
 topic: channel-messages
 producer:
   key-serializer:
org.apache.kafka.common.serialization.StringSerializer
   value-serializer:
org.apache.kafka.common.serialization.StringSerializer
```

Use any scheduler or class at the starting of application to push to kafka:

Using scheduler:

```
KafkaMessageScheduler.java
```

```java
package com.ar.sender.scheduler;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import
com.ar.sender.kafka.producer.channel.message.ChannelMessageProducerServ
ice;

@Component
public class KafkaMessageScheduler {

   private static final Logger LOGGER =
LoggerFactory.getLogger(KafkaMessageScheduler.class);

   private final ChannelMessageProducerService
channelMessageProducerService;
   private int counter = 0; // Counter variable


   @Autowired
   public KafkaMessageScheduler(ChannelMessageProducerService
channelMessageProducerService) {
       this.channelMessageProducerService =
channelMessageProducerService;
   }

   @Scheduled(fixedRate = 5000) // Run every 5 seconds
   public void sendMessage() {
       counter++; // Increment the counter
       String key = "key" + System.currentTimeMillis();
       String value = "Scheduled message at " +
System.currentTimeMillis();
       channelMessageProducerService.sendMessage("Test "+counter);
       printMessageWithCurrentTime("Sent scheduled message: key=" + key
+ ", value=" + value);
   }
```

```
    private void printMessageWithCurrentTime(String message) {
        // formate current timestamp in dd/MM/yyyy HH:mm:ss
        SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
        Date date = new Date();
        String formattedDate = formatter.format(date);

        LOGGER.info("At "+ formattedDate + ": " + message);
    }
}
```

Make sure to add `@EnableScheduling` in spring boot application starting application class.

```
package com.ar.sender;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableScheduling
public class SenderApplication {

    public static void main(String[] args) {
        SpringApplication.run(SenderApplication.class, args);
    }

}
```

It'll start pushing into kafka topics as specified in application.yml.

Consumer service can be viewed in command as well:

```
a-r-danish@a-r-danish-HP-Laptop-15s-eq2xxx:~/Documents/Software/kafka_2.13-3.8.0$ bin/kafka-console-consumer.sh --topic channel-messages --from-beginning --bootstrap-server localhost:9092
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test
Test 1
Test 2
Test 3
Test 4
Test 5
Test 6
Test 7
```

That’s it.

Step 4: Writing consumer service

Add the dependency:

```
<dependency>
            <groupId>org.springframework.kafka</groupId>
            <artifactId>spring-kafka</artifactId>
            <version>3.2.2</version>
        </dependency>
```

Write consumer config and consumer service:

```
ChannelMessageConsumerConfig.java
```

```
package com.ar.receiver.kafka.consumer.channel.message;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.common.serialization.StringDeserializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.annotation.EnableKafka;
import
org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactor
y;
import org.springframework.kafka.core.ConsumerFactory;
import org.springframework.kafka.core.DefaultKafkaConsumerFactory;

@EnableKafka
@Configuration
public class ChannelMessageConsumerConfig {
```

```
    @Value("${kafka.bootstrap-servers}")
    private String bootstrapServers;

    @Value("${kafka.consumer.group-id}")
    private String groupId;

    @Bean
    public ConsumerFactory<String, String> consumerFactory() {
        Map<String, Object> props = new HashMap<>();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
bootstrapServers);
        props.put(ConsumerConfig.GROUP_ID_CONFIG, groupId);
        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>
kafkaListenerContainerFactory() {
        ConcurrentKafkaListenerContainerFactory<String, String> factory
= new ConcurrentKafkaListenerContainerFactory<>();
        factory.setConsumerFactory(consumerFactory());
        return factory;
    }
}
```

```
ChannelMessageConsumerService.java
```

```java
package com.ar.receiver.kafka.consumer.channel.message;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Service;

@Service
public class ChannelMessageConsumerService {

   private static final Logger LOGGER =
LoggerFactory.getLogger(ChannelMessageConsumerService.class);

   @KafkaListener(topics = "${kafka.topic}", groupId =
"${kafka.consumer.group-id}")
   public void listen(String message) {
       LOGGER.info("Received message: " + message);
   }
}
```

Add properties in application.yml:

```yaml
kafka:
 bootstrap-servers: localhost:9092
 topic: channel-messages
 consumer:
   group-id: channel-group
```

Give any topic name we like, it doesn’t necessarily need to be there if not here the kafka will create a default group id for this topic subscriber.

Run the application and it will start listening from the producer:

```
receiver > src > main > resources > application.yml
spring:
  application:


server:
  port: 8102

kafka:
  bootstrap-servers: localhost:9092
  topic: channel-messages
  consumer:
    group-id: channel-group
    # The group ID is a string that uniquely identifies the group of consumer processes to which this consumer belongs.
    # By setting the group ID, we can ensure that the consumer is part of a group of consumers that share the load of consuming messages
```

```
PROBLEMS 1K+   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   COMMENTS

2024-07-30T18:51:11.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 48
2024-07-30T18:51:16.798+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 49
2024-07-30T18:51:21.798+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 50
2024-07-30T18:51:26.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 51
2024-07-30T18:51:31.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 52
2024-07-30T18:51:36.798+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 53
2024-07-30T18:51:41.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 54
2024-07-30T18:51:46.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 55
2024-07-30T18:51:51.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 56
2024-07-30T18:51:56.798+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 57
2024-07-30T18:52:01.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 58
2024-07-30T18:52:06.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 59
2024-07-30T18:52:11.798+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 60
2024-07-30T18:52:16.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 61
2024-07-30T18:52:21.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 62
2024-07-30T18:52:26.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 63
2024-07-30T18:52:31.798+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 64
2024-07-30T18:52:36.797+05:30  INFO 544824 --- [receiver] [ntainer#0-0-C-1] .r.k.c.c.m.ChannelMessageConsumerService : Received message: Test 65

bash
Run: Receiv...
Run: Sende...

Ln 10, Col 1 (109 selected)   Spaces: 2   UTF-8   LF   YAML
```

We can also verify that Group will be created once a single message pushed to the topic:

```
^CProcessed a total of 27 messages
a-r-danish@a-r-danish-HP-Laptop-15s-eq2xxx:~/Documents/Software/kafka_2.13-3.8.0$ kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
kafka-consumer-groups.sh: command not found
a-r-danish@a-r-danish-HP-Laptop-15s-eq2xxx:~/Documents/Software/kafka_2.13-3.8.0$ bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
console-consumer-15783
a-r-danish@a-r-danish-HP-Laptop-15s-eq2xxx:~/Documents/Software/kafka_2.13-3.8.0$ bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
channel-group
console-consumer-15783
a-r-danish@a-r-danish-HP-Laptop-15s-eq2xxx:~/Documents/Software/kafka_2.13-3.8.0$
```

That’s it. We have a producer producing the messages and a consumer which is subscribed to the topic of producer. This is the complete service.

Now, we have to look advancement of topic by partitioning and other more information.