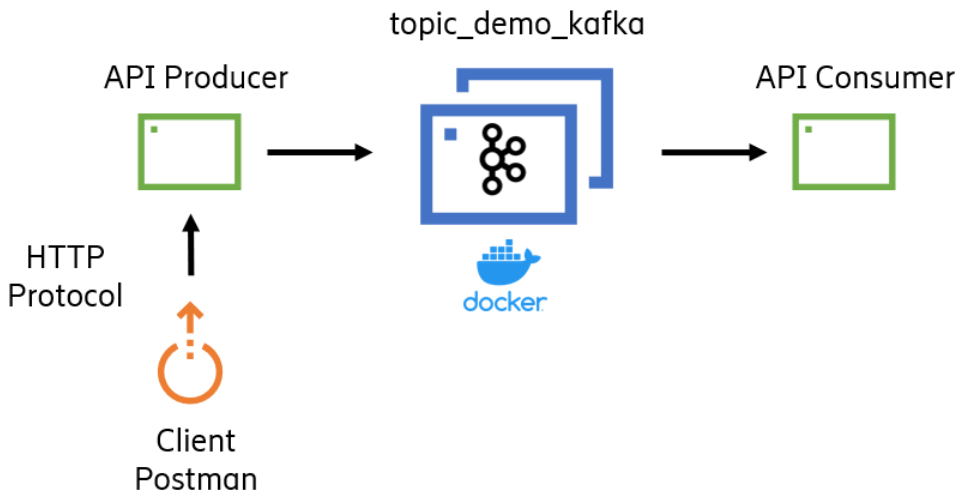
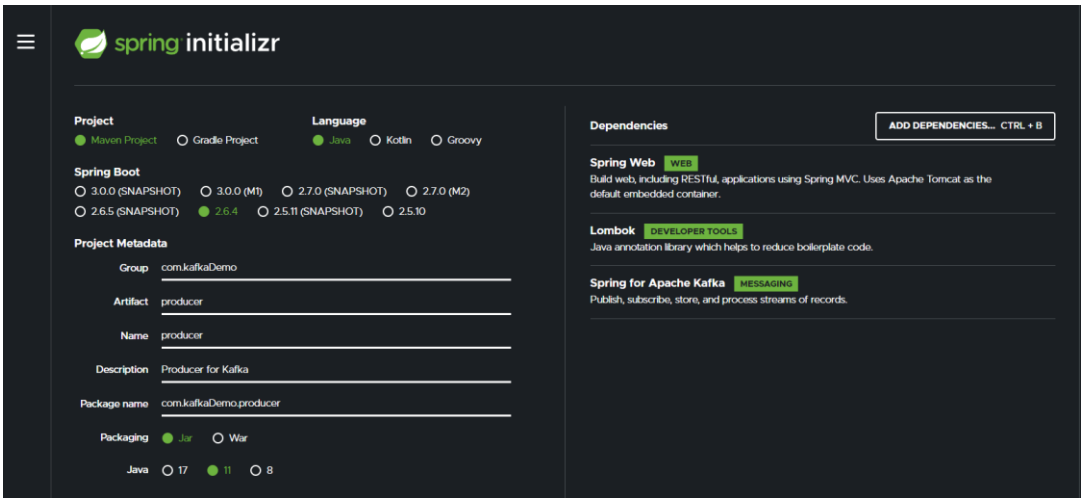
Spring & Kafka & Docker Project – Hello World

Project Architecture Design

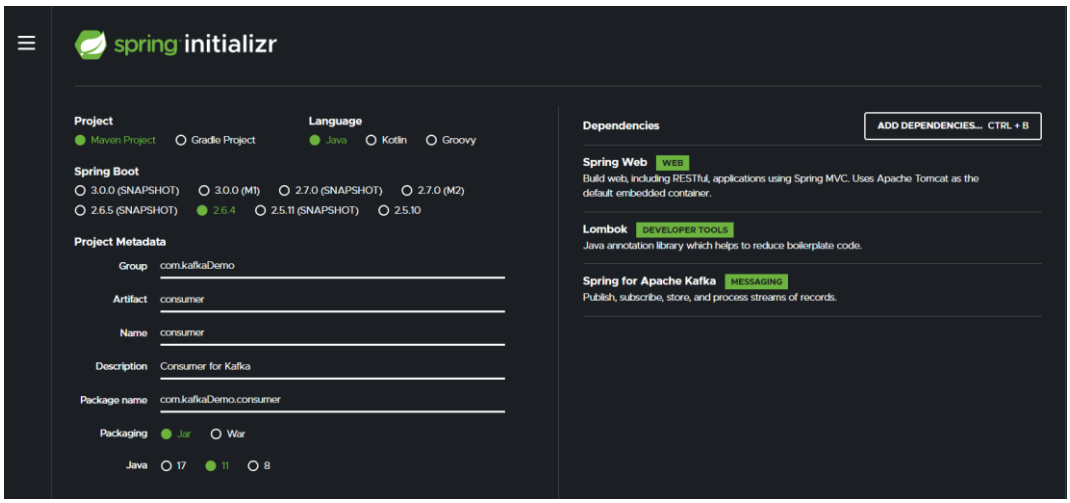


Creating Dependency Packages

1. Access [Spring Initializr](https://start.spring.io/) to generate two packages containing all the necessary dependencies for the project: Spring Web, Lombok and Spring for Apache Kafka.
2. On the main page, create the first package (producer) using the configs as the image below. Generate the zip folder.



3. For generating the second package (consumer), go again to the main page, create the package using the configs as the image below (same properties minus Artifact, Name and Description). Generate the zip folder.



4. Open your preferred IDE and import both unzipped folders as Maven projects.

Bringing up Docker with Kafka application.

1. Inside your workspace, with the producer and consumer folders, create a new folder and name it “docker”. The project should now have 3 folders: producer, consumer and docker.
2. Inside docker folder create the file docker-compose.yml and copy the script below. This file contains all the set up to initialize 3 containers on docker: zookeeper, kafka and kafdrop.

```yaml
version: '3'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    networks:
      - broker-kafka
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  kafka:
    image: confluentinc/cp-kafka:latest
    networks:
      - broker-kafka
    depends_on:
      - zookeeper
    ports:
      - 9090:9090
    environment:
```

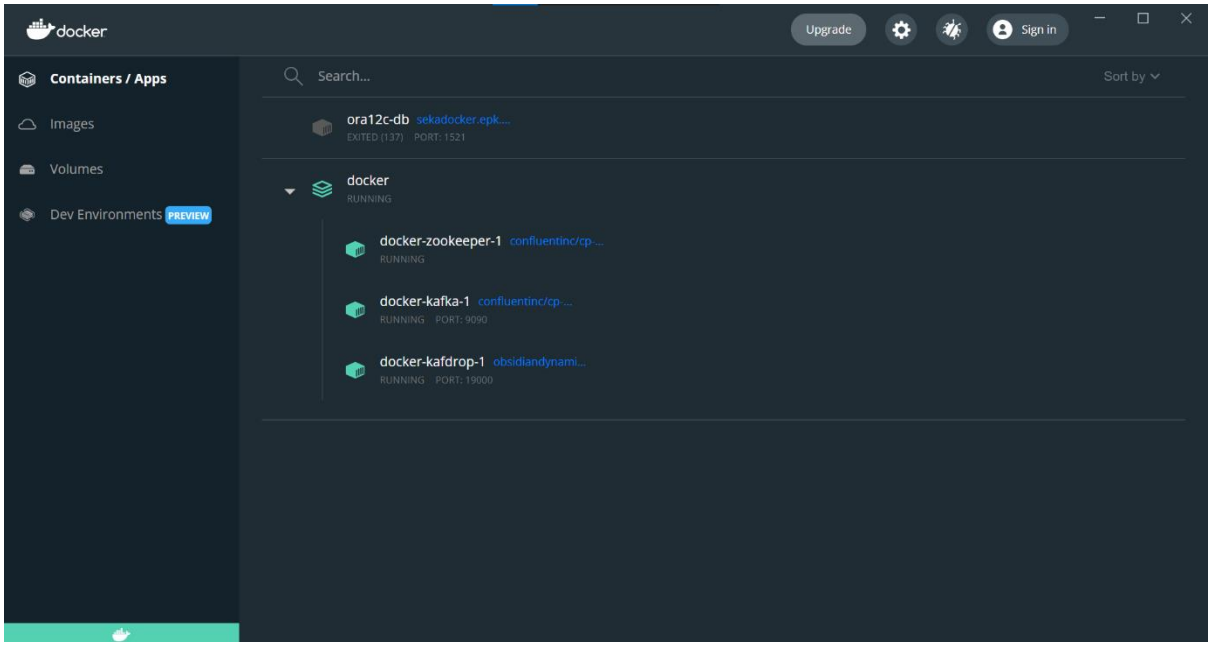
```
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS:
PLAINTEXT://kafka:29090,PLAINTEXT_HOST://localhost:9090
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

  kafdrop:
    image: obsidiandynamics/kafdrop:latest
    networks:
      - broker-kafka
    depends_on:
      - kafka
    ports:
      - 19000:9000
    environment:
      KAFKA_BROKERCONNECT: kafka:29090

networks:
  broker-kafka:
    driver: bridge
```

3. Now to run the docker-compose.yml and bring up the 3 containers, open a command prompt inside docker folder and run:

   docker-compose -f docker-compose.yml up -d

4. If your containers initialize and run without errors, the message below should appear on your command prompt. It is possible to open docker desktop and check the 3 containers created and running.

```
[+] Running 3/3
 - Container docker-zookeeper-1  Started
 - Container docker-kafka-1      Started
 - Container docker-kafdrop-1    Started
```



Creating Producer API

1. Inside the producer package, on the path producer/src/main/resources delete the file application.properties and create a new one called application.yml and copy the script below:

```
spring:
  kafka:
    producer:
      bootstrap-servers: localhost:9090
      key-serializer:
org.apache.kafka.common.serialization.StringSerializer
      value-serializer:
org.apache.kafka.common.serialization.StringSerializer

topic:
  demo-topic: topic_demo_kafka
```

2. On the path producer/src/main/java/com.kafkaDemo.producer create a new package “service” and inside that package create a java file. Name it “ProducerService.java” and copy the script below:

```
package com.kafkaDemo.producer.service;

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;
```

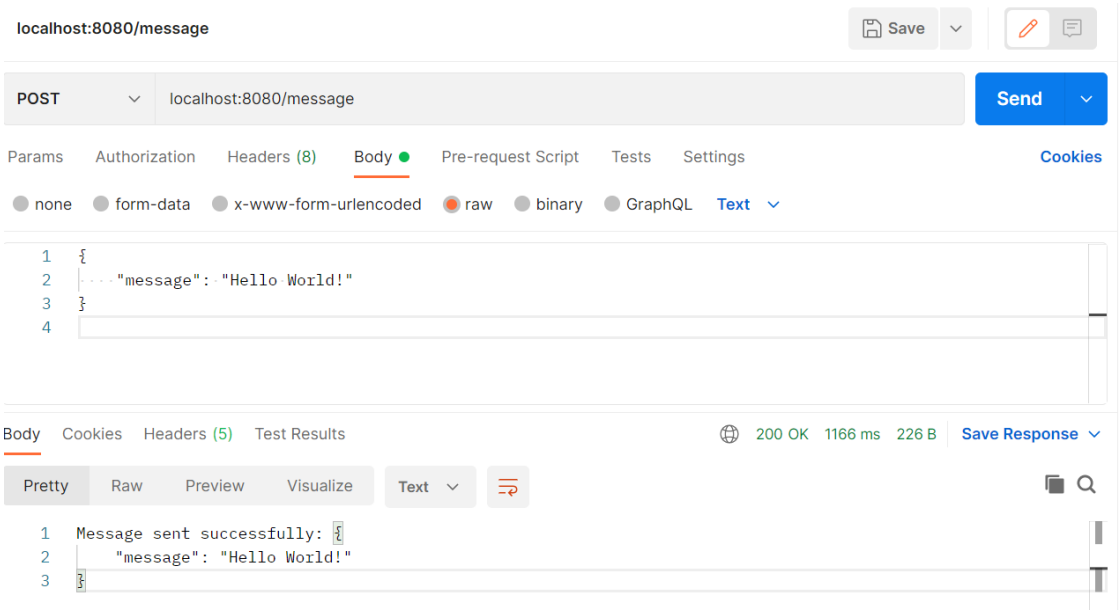
```
@Service
public class ProducerService {

    private static final Logger logger =
LoggerFactory.getLogger(ProducerService.class);

    @Value("${topic.demo-topic}")
    private String topicDemoTopic;

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    public void sendMessageToKafka(String message) {
        logger.info("Message -> {}", message); //Message in
terminal
        this.kafkaTemplate.send(topicDemoTopic, message);
    }
}
```

3. On the path producer/src/main/java/com.kafkaDemo.producer create a new package “resource” and inside that package create a java file. Name it “ProducerResource.java” and copy the script below:

```
package com.kafkaDemo.producer.resource;


import com.producer.producer.service.ProducerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/message")
public class ProducerResource {

    @Autowired
    ProducerService producerService;

    @PostMapping
    public ResponseEntity<String> sendMessage(@RequestBody String
message){
        producerService.sendMessageToKafka(message);
        return ResponseEntity.ok().body("Message sent
successfully: " + message); //HTTP response (code 200)
    }
}
```

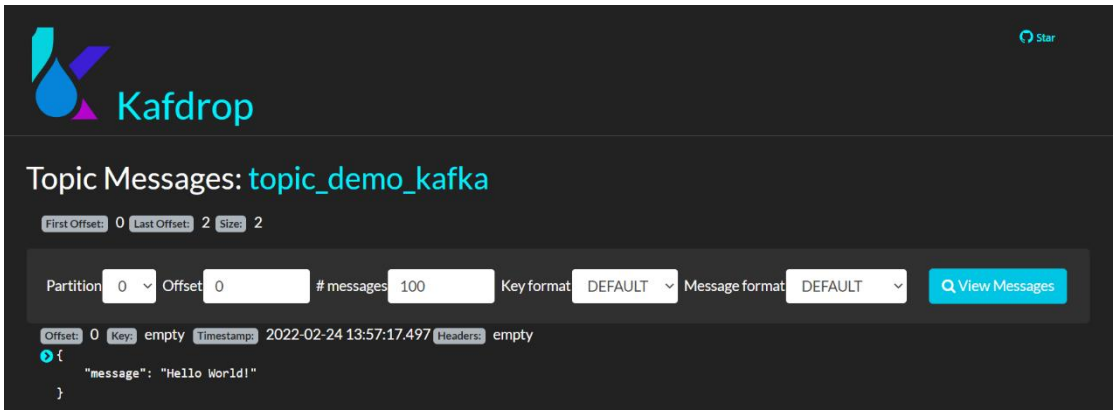
4. Now, on the path producer/src/main/java/com.kafkaDemo.producer, execute the file “ProducerApplication.java. Spring will initialize on port 8080.

5. To verify if the producer is working, use an application to send a HTTP Post command (Postman or Insomnia). Create a HTTP Request to send a Post for the address “localhost:8080/message”. Check the image below where Postman is used to send the post command.



6. After sending the command, check the console where ProducerApplication.java is running to see if your producer API captured the message

7. It is also possible to verify if the producer API captured the message through the kafdrop dashboard. On a web browser, go to the address “localhost:19000”. On the bottom of the page select the topic “topic_demo_kafka”, click on “View Messages” and click again on the “View Messages” button to see all the messages that the producer API captured.



Creating Consumer API

1. Inside the consumer package, on the path producer/src/main/resources delete the file application.properties and create a new one called application.yml and copy the script below. It is important to specify a new port different from 8080, in this example we choose 8081:

```
spring:
  kafka:
    consumer:
      bootstrap-servers: localhost:9090 #Kafka port
      group-id: group_id
      auto-offset-reset: earliest
      key-deserializer:
org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer:
org.apache.kafka.common.serialization.StringDeserializer

topic:
  demo-topic: topic_demo_kafka

server:
  port : 8081 #IMPORTANT
```

2. On the path consumer/src/main/java/com.kafkaDemo.consumer create a new package “listener” and inside that package create a java file. Name it “ConsumerListener.java” and copy the script below. The consumer API will consume the message received and log it on the console.

```
package com.kafkaDemo.consumer.listener;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Service;
import java.io.IOException;
```

```
@Service
public class ConsumerListener {

    private final Logger logger =
LoggerFactory.getLogger(ConsumerListener.class);

    @KafkaListener(topics = "${topic.demo-topic}", groupId =
"group_id")
    public void consume(String message) throws IOException {
        logger.info(String.format("Listening message: %s", message));
    }
}
```

3. Execute the file ConsumerApplication.java, SpringBoot will initialize on the selected port. You should be able to see all the messages sent by postman displayed at the bottom of the console. As planned, these messages have been captured by the producer and consumed by the consumer.

References

1. Project repository
2. Kafka Documentation
3. Tutorial of a Similar project
4. How to Work with Apache Kafka in Your Spring Boot Application

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.3)

2022-02-25 10:09:19.733  INFO 22352 --- [           main] c.k.consumer.ConsumerApplication         : Starting ConsumerApplication using Java 11.0.10 on BR-00002989 with PID 22352
2022-02-25 10:09:19.744  INFO 22352 --- [           main] c.k.consumer.ConsumerApplication         : No active profile set, falling back to default profiles: default
2022-02-25 10:09:26.814  INFO 22352 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2022-02-25 10:09:26.850  INFO 22352 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-02-25 10:09:26.850  INFO 22352 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.56]
2022-02-25 10:09:27.260  INFO 22352 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-02-25 10:09:27.261  INFO 22352 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 7193 ms
2022-02-25 10:09:30.695  INFO 22352 --- [           main] o.a.k.clients.consumer.ConsumerConfig    : ConsumerConfig values:
        allow.auto.create.topics = true
        auto.commit.interval.ms = 5000
        auto.offset.reset = earliest
        bootstrap.servers = [localhost:9090]
        check.crcs = true
        client.dns.lookup = use_all_dns_ips
        client.id = consumer-group_id-1
        client.rack =
        connections.max.idle.ms = 540000
        default.api.timeout.ms = 60000
        enable.auto.commit = false
        exclude.internal.topics = true
        fetch.max.bytes = 52428800
        fetch.max.wait.ms = 500
        fetch.min.bytes = 1
        group.id = group_id
        group.instance.id = null
        heartbeat.interval.ms = 3000
        interceptor.classes = []
        internal.leave.group.on.close = true
        internal.throw.on.fetch.stable.offset.unsupported = false
        isolation.level = read_uncommitted
        key.deserializer = class org.apache.kafka.common.serialization.StringDeserializer
        max.partition.fetch.bytes = 1048576
        max.poll.interval.ms = 300000
        max.poll.records = 500
        metadata.max.age.ms = 300000
        metric.reporters = []
        metrics.num.samples = 2
        metrics.recording.level = INFO
        metrics.sample.window.ms = 30000
        partition.assignment.strategy = [class org.apache.kafka.clients.consumer.RangeAssignor, class org.apache.kafka.clients.consumer.CooperativeStickyAssignor]
        receive.buffer.bytes = 65536
        reconnect.backoff.max.ms = 1000
        reconnect.backoff.ms = 50
        request.timeout.ms = 30000
        retry.backoff.ms = 100
        sasl.client.callback.handler.class = null
        sasl.jaas.config = null
        sasl.kerberos.kinit.cmd = /usr/bin/kinit
        sasl.kerberos.min.time.before.relogin = 60000
        sasl.kerberos.service.name = null
        sasl.kerberos.ticket.renew.jitter = 0.05
        sasl.kerberos.ticket.renew.window.factor = 0.8
        sasl.login.callback.handler.class = null
        sasl.login.class = null
        sasl.login.refresh.buffer.seconds = 300
        sasl.login.refresh.min.period.seconds = 60
        sasl.login.refresh.window.factor = 0.8
        sasl.login.refresh.window.jitter = 0.05
        sasl.mechanism = GSSAPI
        security.protocol = PLAINTEXT
        security.providers = null
        send.buffer.bytes = 131072
        session.timeout.ms = 45000
        socket.connection.setup.timeout.max.ms = 30000
        socket.connection.setup.timeout.ms = 10000
        ssl.cipher.suites = null
        ssl.enabled.protocols = [TLSv1.2, TLSv1.3]
        ssl.endpoint.identification.algorithm = https
        ssl.engine.factory.class = null
        ssl.key.password = null
        ssl.keymanager.algorithm = SunX509
        ssl.keystore.certificate.chain = null
        ssl.keystore.key = null
        ssl.keystore.location = null
        ssl.keystore.password = null
        ssl.keystore.type = JKS
        ssl.protocol = TLSv1.3
        ssl.provider = null
        ssl.secure.random.implementation = null
        ssl.trustmanager.algorithm = PKIX
        ssl.truststore.certificates = null
        ssl.truststore.location = null
        ssl.truststore.password = null
        ssl.truststore.type = JKS
        value.deserializer = class org.apache.kafka.common.serialization.StringDeserializer

2022-02-25 10:09:31.515  INFO 22352 --- [           main] o.a.kafka.common.utils.AppInfoParser     : Kafka version: 3.0.0
2022-02-25 10:09:31.526  INFO 22352 --- [           main] o.a.kafka.common.utils.AppInfoParser     : Kafka commitId: 8cb0a5e9d3441962
2022-02-25 10:09:31.527  INFO 22352 --- [           main] o.a.kafka.common.utils.AppInfoParser     : Kafka startTimeMs: 1645794571505
2022-02-25 10:09:31.541  INFO 22352 --- [           main] o.a.k.clients.consumer.KafkaConsumer     : [Consumer clientId=consumer-group_id-1, groupId=group_id] Subscribed to topic(s): topic_demo_kafka
2022-02-25 10:09:31.637  INFO 22352 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with context path ''
2022-02-25 10:09:31.697  INFO 22352 --- [           main] c.k.consumer.ConsumerApplication         : Started ConsumerApplication in 14.945 seconds (JVM running for 19.076)
2022-02-25 10:09:34.019  INFO 22352 --- [ntainer#0-0-C-1] org.apache.kafka.clients.Metadata        : [Consumer clientId=consumer-group_id-1, groupId=group_id] Cluster ID: nLLeWjLUQ4-UJWD85LzQsA
2022-02-25 10:09:34.025  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Discovered group coordinator localhost:9090 (id: 2147483646 rack: null)
2022-02-25 10:09:34.034  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] (Re-)joining group
2022-02-25 10:09:34.124  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Request joining group due to: need to re-join with the given member-id
2022-02-25 10:09:34.125  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] (Re-)joining group
2022-02-25 10:09:37.194  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Successfully joined group with generation Generation{generationId=22, memberId='consumer-group_id-1-
2022-02-25 10:09:37.209  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Finished assignment for group at generation 22: {consumer-group_id-1-428a1fd8-9fdd-4643-ac35-12d7913
2022-02-25 10:09:37.269  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Successfully synced group in generation Generation{generationId=22, memberId='consumer-group_id-1-42
2022-02-25 10:09:37.271  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Notifying assignor about the new Assignment(partitions=[topic_demo_kafka-0])
2022-02-25 10:09:37.291  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Adding newly assigned partitions: topic_demo_kafka-0
2022-02-25 10:09:37.417  INFO 22352 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : [Consumer clientId=consumer-group_id-1, groupId=group_id] Setting offset for partition topic_demo_kafka-0 to the committed offset FetchPosition{offset=3, offs
2022-02-25 10:09:37.421  INFO 22352 --- [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer    : group_id: partitions assigned: [topic_demo_kafka-0]
2022-02-25 10:09:37.694  INFO 22352 --- [ntainer#0-0-C-1] c.k.consumer.listener.ConsumerListener   : Listening message: {
    "mensagem": "Hello World!"
}
```