Prueba de Escalabilidad y Paralelismo con Kafka

Objetivo

Demostrar cómo Kafka distribuye automáticamente la carga de mensajes entre múltiples consumidores en un mismo **consumer group**, usando un topic con varias particiones.

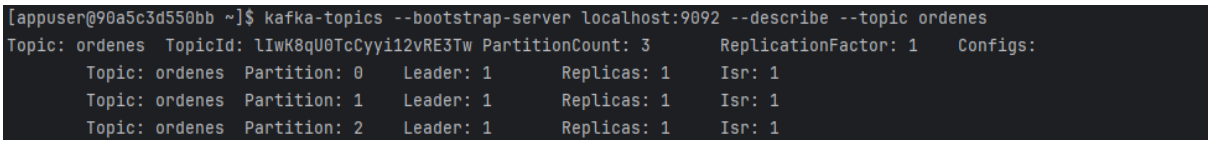
1. Preparación del Topic

1. Crear un topic `ordenes` con 3 particiones:

```
kafka-topics --bootstrap-server localhost:9092 --create --topic ordenes --partitions 3 --replication-factor 1
```

2. Verificar particiones:

```
kafka-topics --bootstrap-server localhost:9092 --describe --topic ordenes
```



2. Configuración de los Consumidores

- Crear dos instancias de consumidor (`kafka-consumer`) corriendo en **mismo consumer group** `grupo-ordenes`:

```
# Instancia 1
./mvnw spring-boot:run
-Dspring-boot.run.arguments="--server.port=8081"
```

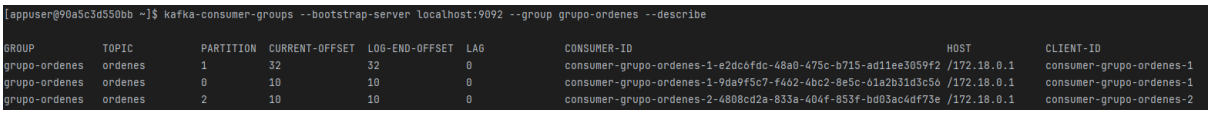
```
# Instancia 2
./mvnw spring-boot:run
-Dspring-boot.run.arguments="--server.port=8083"
```

3. Verificación de la Distribución de Particiones

- Listar grupos y particiones:

```
kafka-consumer-groups --bootstrap-server localhost:9092 --group
grupo-ordenes --describe
```

- Resultado esperado:
  Cada consumidor tiene asignadas diferentes particiones.



```
[appuser@90a5c3d550bb ~]$ kafka-consumer-groups --bootstrap-server localhost:9092 --group grupo-ordenes --describe

GROUP          TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG             CONSUMER-ID                                           HOST            CLIENT-ID
grupo-ordenes  ordenes        1          32              32              0               consumer-grupo-ordenes-1-e2dc6fdc-48a0-475c-b715-ad11ee3059f2 /172.18.0.1     consumer-grupo-ordenes-1
grupo-ordenes  ordenes        0          10              10              0               consumer-grupo-ordenes-1-9da9f5c7-f462-4bc2-8e5c-61a2b31d3c56 /172.18.0.1     consumer-grupo-ordenes-1
grupo-ordenes  ordenes        2          10              10              0               consumer-grupo-ordenes-2-4808cd2a-833a-404f-853f-bd03ac4df73e /172.18.0.1     consumer-grupo-ordenes-2
```

4. Envío Masivo de Mensajes con Postman

1. Crear un **request POST** en Postman:

```
POST http://localhost:8080/orders
Content-Type: application/json
Body:
{
  "id": {{id}},
  "cliente": "Producto-{{id}}",
  "monto": {{id}}
}
```

2. Guardar request en una **Collection** `Ordenes`.

Usando Collection Runner

1. Abrir Collection Runner → seleccionar la colección `Ordenes`.
2. Configurar:
    - **Iterations:** número de mensajes a enviar (ej. 50)
    - **Delay:** opcional entre requests (ej. 100 ms)
    - **Data File (CSV/JSON):** si querés enviar `orderId` diferentes automáticamente.
3. Ejecutar y observar cómo los mensajes se reparten entre los consumidores.

5. Observación

- En la consola de los consumidores se verá algo como:

```
None
2025-09-23T12:00:07.710-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@75250657
2025-09-23T12:00:07.710-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@6a2d6c91
2025-09-23T12:00:07.730-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@7fee32fc
2025-09-23T12:00:07.730-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@1ddd1758
2025-09-23T12:00:07.739-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@4c4c0b8
2025-09-23T12:00:07.760-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@31b28687
```

```
2025-09-23T12:00:07.929-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@4e09fb04
2025-09-23T12:00:08.009-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@6229b478
2025-09-23T12:00:08.195-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@80c6e34
2025-09-23T12:00:08.290-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@68dcb613
2025-09-23T12:00:08.451-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@6fbc41b
2025-09-23T12:00:08.510-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@1dce0b87
2025-09-23T12:00:08.695-03:00  INFO 20884 --- [ntainer#0-0-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-0-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@2498157c
2025-09-23T12:00:08.787-03:00  INFO 20884 --- [ntainer#0-1-C-1]
c.c.kafka.consumer.OrderConsumer         : Mensaje recibido en
[org.springframework.kafka.KafkaListenerEndpointContainer#0-1-C-1]:
com.capacitacion.kafka.common.model.OrderEvent@386d3007
```

Demostrando **paralelismo y escalabilidad automática** de Kafka.

20386114553

3984-0800