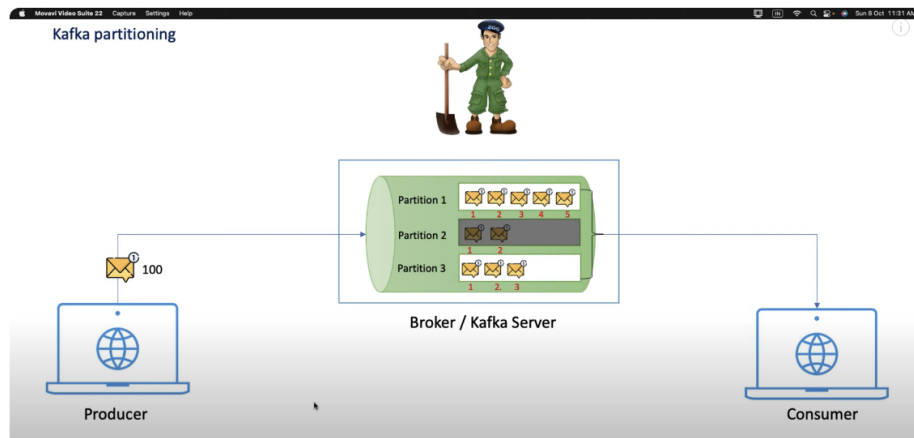


Mastering Message Routing with Specific Partitions

How to make producer to send messages to only one particular topic ?



PLAY WITH KAFKA PARTITION

A topic with 5 partitions

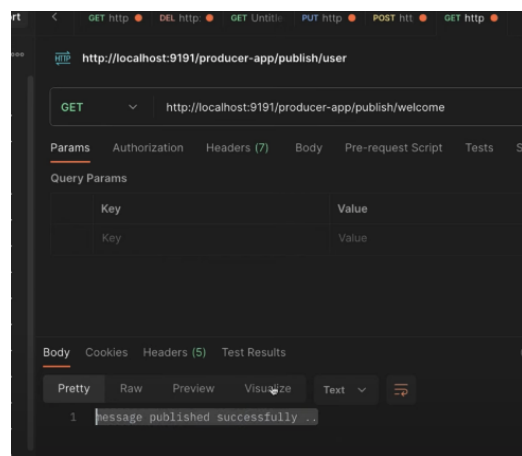
```
void consumeEvents(Customer customer) {  
    .info("con  
    kafka_2.12-3.4.0 --zsh -- 80x24  
    javatechie@Basantas-iMac kafka_2.12-3.4.0 % sh bin/kafka-topics.sh --bootstrap-s  
    er localhost:9092 --create --topic javatechie-topic --partitions 5 --replicat  
    ion-factor 1  
    Created topic javatechie-topic.  
    javatechie@Basantas-iMac kafka_2.12-3.4.0 %  
    onsumerExam  
    .kafka.cli
```

```
EventController.java x KafkaMessagePublisher.java x KafkaProducerExampleApplication.java x  
11 @RequestMapping("/producer-app")  
12 public class EventController {  
13  
14     @Autowired  
15     private KafkaMessagePublisher publisher;  
16  
17     @GetMapping("/publish/{message}")  
18     public ResponseEntity<?> publishMessage(@PathVariable String message) {  
19  
20         try {  
21             for (int i = 0; i <= 10000; i++) {  
22                 publisher.sendMessageToTopic(message + " : " + i);  
23             }  
24             return ResponseEntity.ok("message published successfully ..");  
25         } catch (Exception ex) {  
26             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)  
27                 .build();  
28         }  
29     }  
30 }
```

```

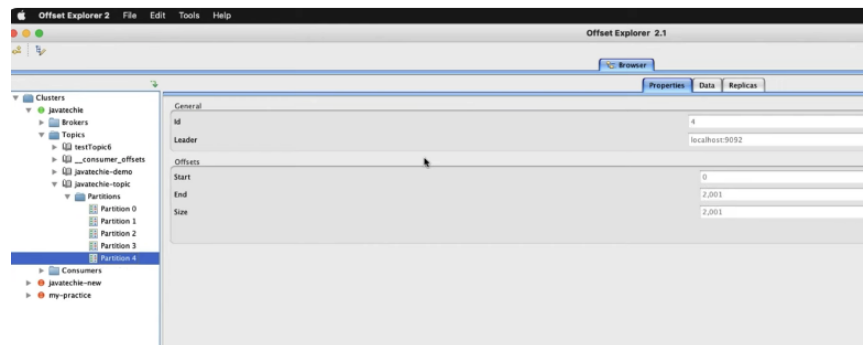
10
11 @Service
12 public class KafkaMessagePublisher {
13
14     @Autowired
15     private KafkaTemplate<String, Object> template;
16
17     public void sendMessageToTopic(String message){
18         CompletableFuture<SendResult<String, Object>> future = template.send(topic: "javatechie-topic", message);
19         future.whenComplete((result, ex)->{
20             if (ex == null) {
21                 System.out.println("Sent message=[" + message +
22                     "] with offset=[" + result.getRecordMetadata().offset() + "]");
23             } else {
24                 System.out.println("Unable to send message=[" +
25                     message + "] due to : " + ex.getMessage());
26             }
27         });
28     }
29 }
30

```

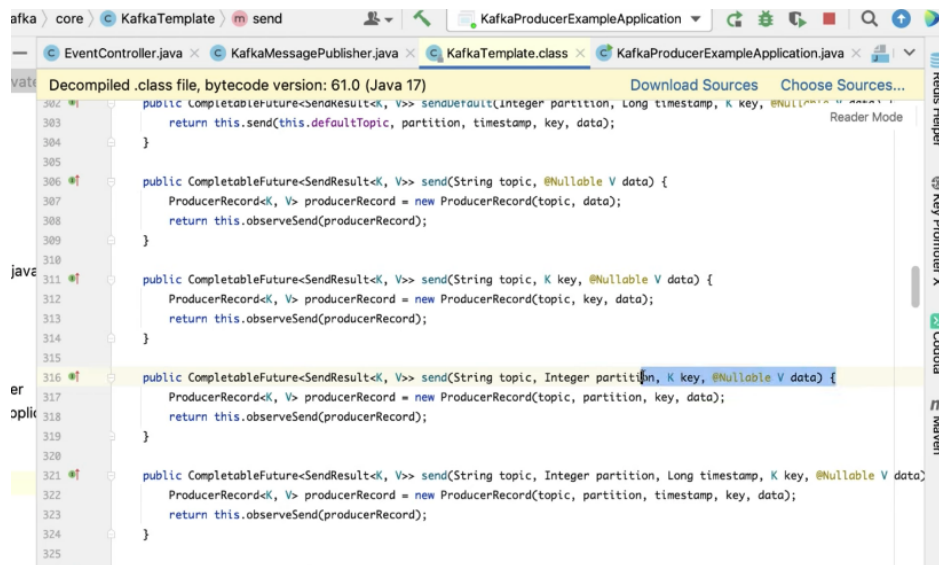


Sent 10K messages

5 partitions in a topic manages to send 10K messages.
In the below screenshot 2001 messages sent via partition 4.



There is an override method on which topic and which partition has to be used.



The screenshot shows the `KafkaTemplate` class with several `send` methods. The methods are as follows:

```
public CompletableFuture<SendResult<K, V>> sendDefault(Integer partition, Long timestamp, K key, @Nullable V data) {
    return this.send(this.defaultTopic, partition, timestamp, key, data);
}

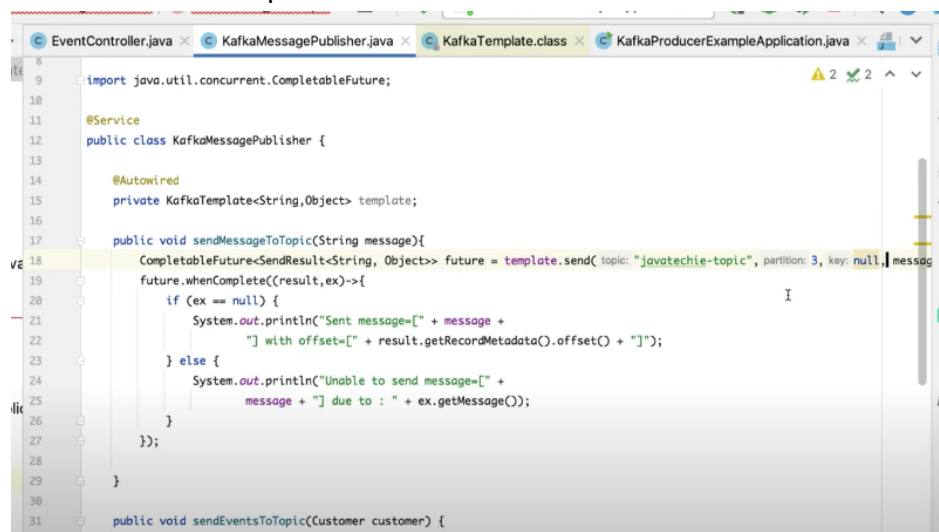
public CompletableFuture<SendResult<K, V>> send(String topic, @Nullable V data) {
    ProducerRecord<K, V> producerRecord = new ProducerRecord(topic, data);
    return this.observeSend(producerRecord);
}

public CompletableFuture<SendResult<K, V>> send(String topic, K key, @Nullable V data) {
    ProducerRecord<K, V> producerRecord = new ProducerRecord(topic, key, data);
    return this.observeSend(producerRecord);
}

public CompletableFuture<SendResult<K, V>> send(String topic, Integer partition, K key, @Nullable V data) {
    ProducerRecord<K, V> producerRecord = new ProducerRecord(topic, partition, key, data);
    return this.observeSend(producerRecord);
}

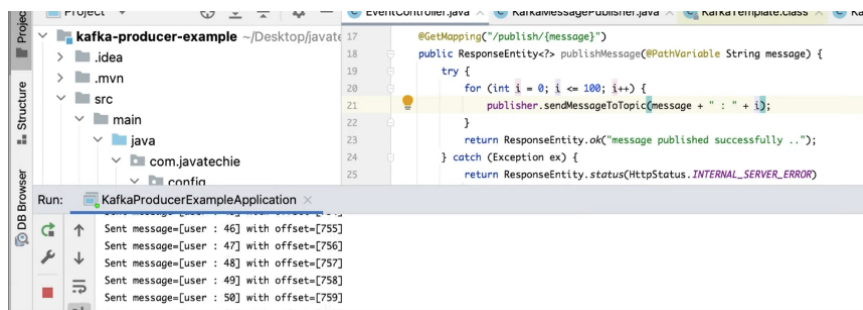
public CompletableFuture<SendResult<K, V>> send(String topic, Integer partition, Long timestamp, K key, @Nullable V data) {
    ProducerRecord<K, V> producerRecord = new ProducerRecord(topic, partition, timestamp, key, data);
    return this.observeSend(producerRecord);
}
```

Call this method for partition 3.



The screenshot shows the `KafkaMessagePublisher` class with the `sendMessageToTopic` method. The method is as follows:

```
public void sendMessageToTopic(String message) {
    CompletableFuture<SendResult<String, Object>> future = template.send(topic: "javatechie-topic", partition: 3, key: null, message);
    future.whenComplete((result, ex) -> {
        if (ex == null) {
            System.out.println("Sent message=[" + message +
                "] with offset=[" + result.getRecordMetadata().offset() + "]);
        } else {
            System.out.println("Unable to send message=[" +
                message + " due to : " + ex.getMessage());
        }
    });
}
```



The screenshot shows the `KafkaProducerExampleApplication` class with the `publishMessage` method. The method is as follows:

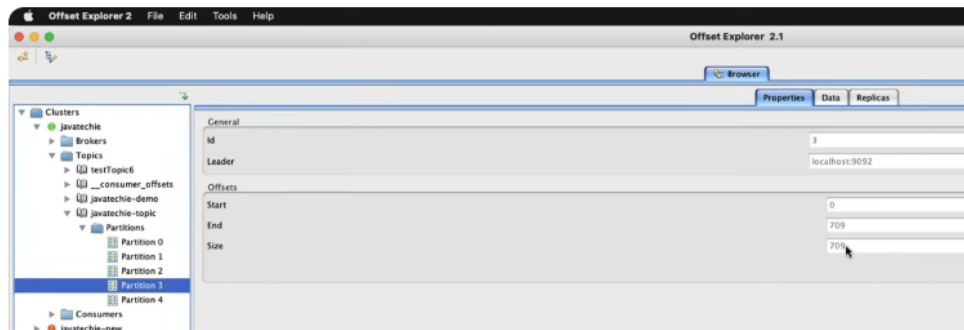
```
@GetMapping("/publish/{message}")
public ResponseEntity<> publishMessage(@PathVariable String message) {
    try {
        for (int i = 0; i <= 100; i++) {
            publisher.sendMessageToTopic(message + " : " + i);
        }
        return ResponseEntity.ok("message published successfully ..");
    } catch (Exception ex) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

The output of the application is shown in the console:

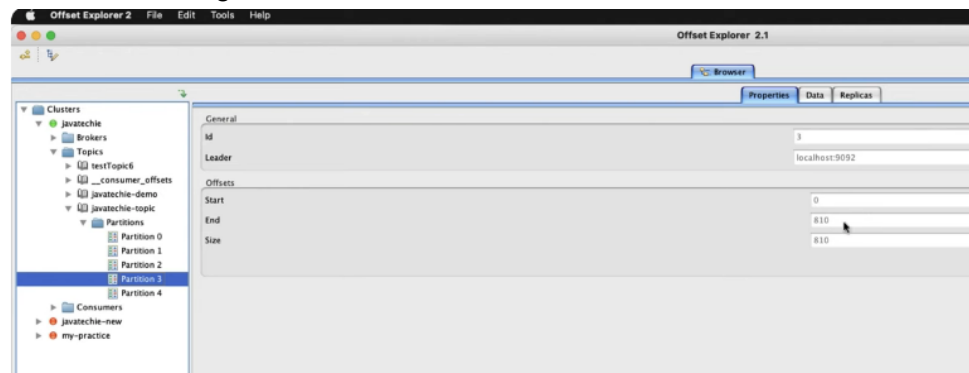
```
Run: KafkaProducerExampleApplication
Sent message=[user : 46] with offset=[755]
Sent message=[user : 47] with offset=[756]
Sent message=[user : 48] with offset=[757]
Sent message=[user : 49] with offset=[758]
Sent message=[user : 50] with offset=[759]
```

Just publish 101 messages.

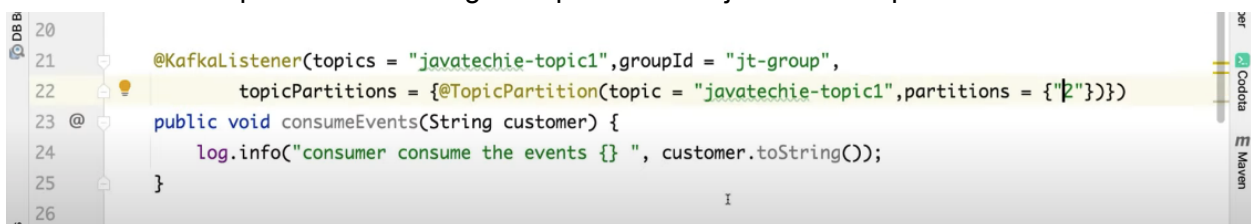
Now notice the partition 3 message count alone. Previously it was 709



After 101 messages it is 810.



The above is from the producer point of view. The below one is from the consumer point of view. In the consumer also we have provision from which topic and from which partition we want read. In the below example we are reading from partition 2 of javatechie-topic. Line number 22



From the producer we are sending so many messages from different partitions of topic "javatechie-topic1". But my consumer will consume only from partition 2.

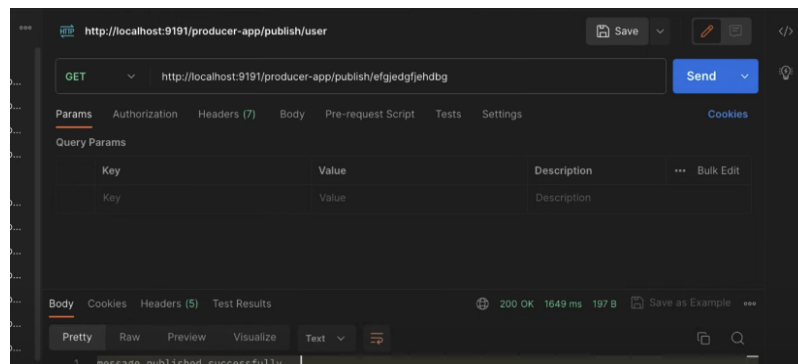


Create a topic with the above name

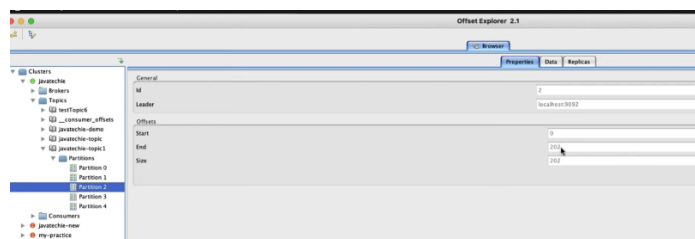
```
partition-factor 1
Created topic javatechie-topic.
23-javatechie@Basantas-iMac kafka_2.12-3.4.0 % sh bin/kafka-topics.sh --bootstrap-s
for server localhost:9092 --create --topic javatechie-topic1 --partitions 5 --replica
partition-factor 1
23-javatechie@Basantas-iMac kafka_2.12-3.4.0 %
```

When the consumer application starts - we can see it read it from 2.

```
sumer clientId=consumer-jt-group-1, groupId=jt-group] Resetting offset for partition javatechie-topic1-2 to position Fet
```



The above controller will call send method 101 times. And our consumer will listen only to partition 2 of the topic.



Notice only welcome and youtube got consumed by partition 2.

