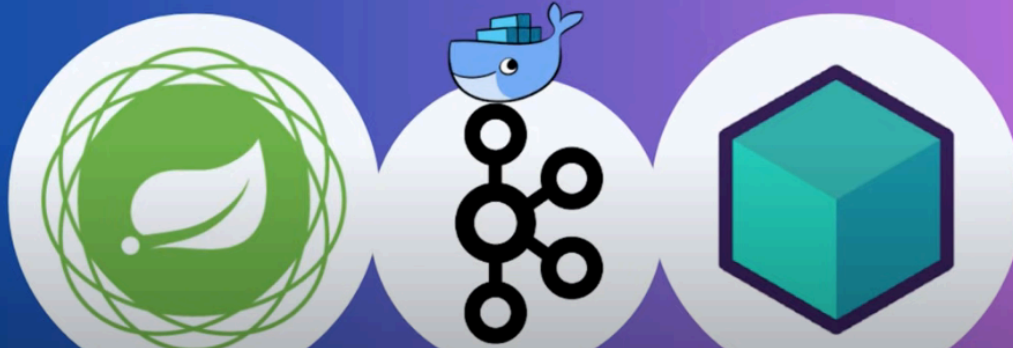


KAFKA E2E INTEGRATION TEST

TESTCONTAINERS



The four dependencies we need

```
45 <groupId>org.testcontainers</groupId>
46 <artifactId>testcontainers</artifactId>
47 <version>1.18.1</version>
48 <scope>test</scope>
49 </dependency>
50 <dependency>
51 <groupId>org.testcontainers</groupId>
52 <artifactId>kafka</artifactId>
53 <version>1.18.1</version>
54 <scope>test</scope>
55 </dependency>
56 <dependency>
57 <groupId>org.testcontainers</groupId>
58 <artifactId>junit-jupiter</artifactId>
59 <version>1.18.1</version>
60 <scope>test</scope>
61 </dependency>
62 <dependency>
63 <groupId>org.awaitility</groupId>
64 <artifactId>awaitility</artifactId>
65 <version>4.2.0</version>
66 <scope>test</scope>
67 </dependency>
```

project > dependencies

```
KafkaProducerExampleApplicationTests.java
11
12 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
13 @Testcontainers
14 public class KafkaProducerExampleApplicationTests {
15
16     @Container
17     KafkaContainer kafka = new KafkaContainer(DockerImageName.parse("confluentinc/cp-kafka:latest"))
18
19     @DynamicPropertySource
20     public void initKafkaProperties(DynamicPropertyRegistry registry){
21
22     }
23 }
```

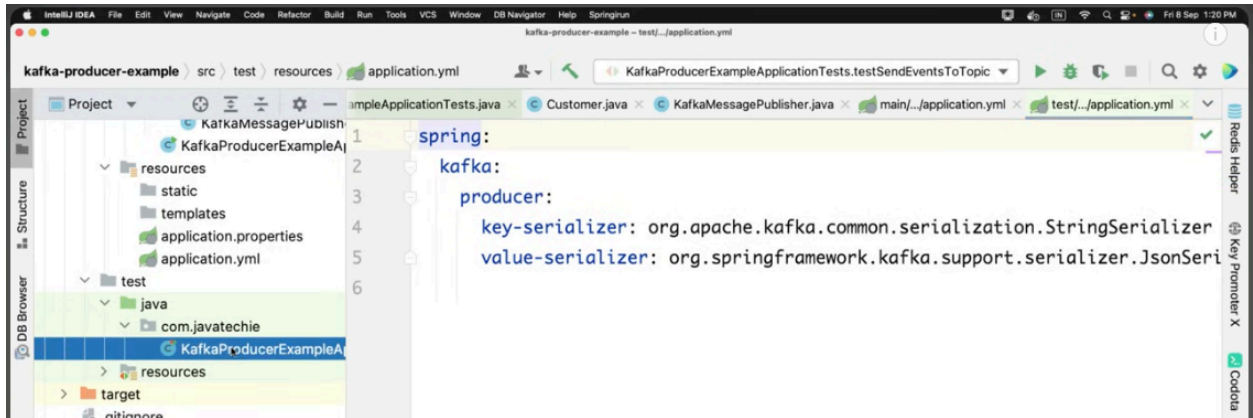
```

16     @Container
17     KafkaContainer kafka = new KafkaContainer(DockerImageName.parse("confluentinc/cp-kafka:latest"))
18
19     @DynamicPropertySource
20     public void initKafkaProperties(DynamicPropertyRegistry registry){
21         registry.add(name: "spring.kafka.bootstrap-servers", kafka::getBootstrapServers);
22     }
23 }
```

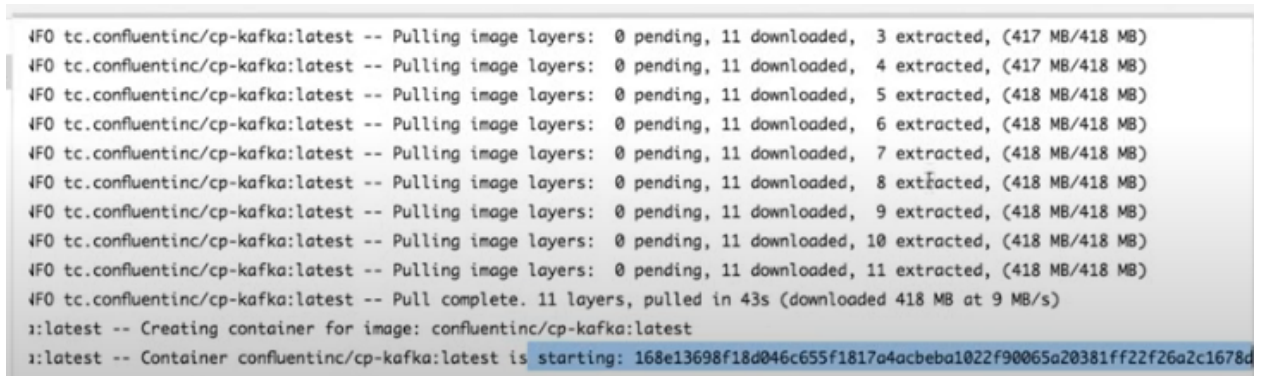
Create kafka as a static variable and initKafkaProperty as static method. Everything will be taken care of by the above two lines. Our focus is only on writing test cases which are given in the below screenshot.

```
KafkaProducerExampleApplicationTests
testSendEventsToTopic
KafkaProducerExampleApplicationTests.testSendEventsToTopic
33 @Autowired
34 private KafkaMessagePublisher publisher;
35
36 @Test
37 public void testSendEventsToTopic() {
38     publisher.sendEventsToTopic(new Customer(id: 263, name: "test user", email: "test@gmail.com"));
39     await().pollInterval(Duration.ofSeconds(3))
40         .atMost(timeout: 10, TimeUnit.SECONDS).untilAsserted() -> {
41         // assert statement
42     };
43 }
44
45 }
46 }
```

Create test Application properties



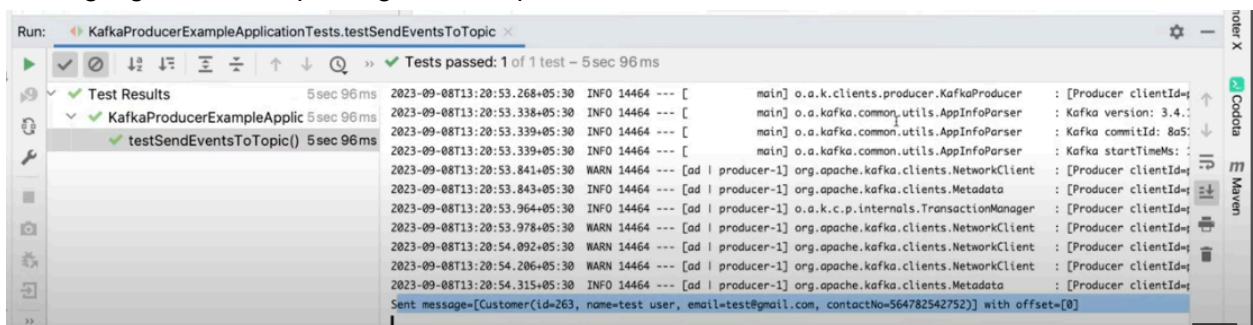
Run the test. We can see test container docker pull others in the console



We can notice that under docker images

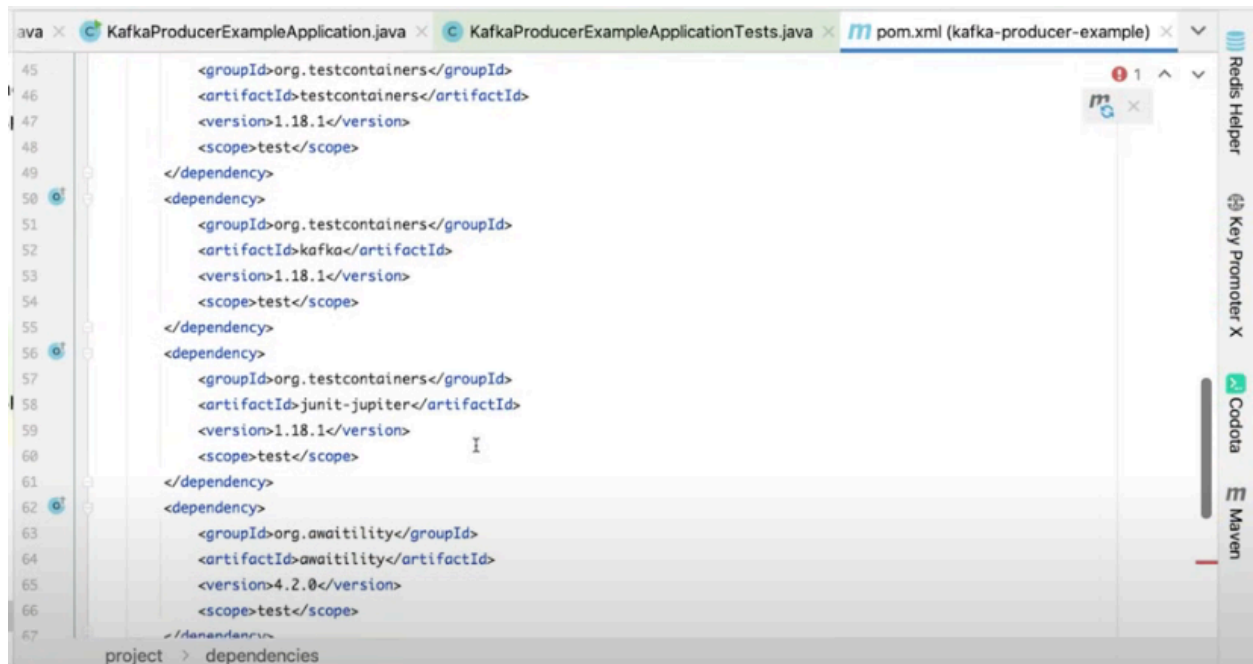


The highlighted one is printing from the publisher

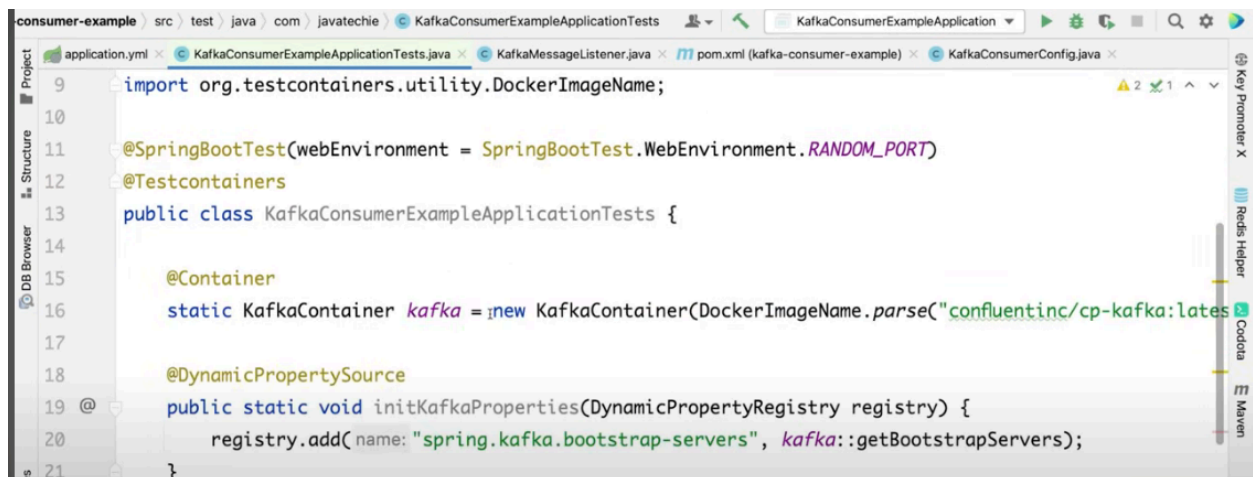


Let's do same thing for consumer

Add this dependencies

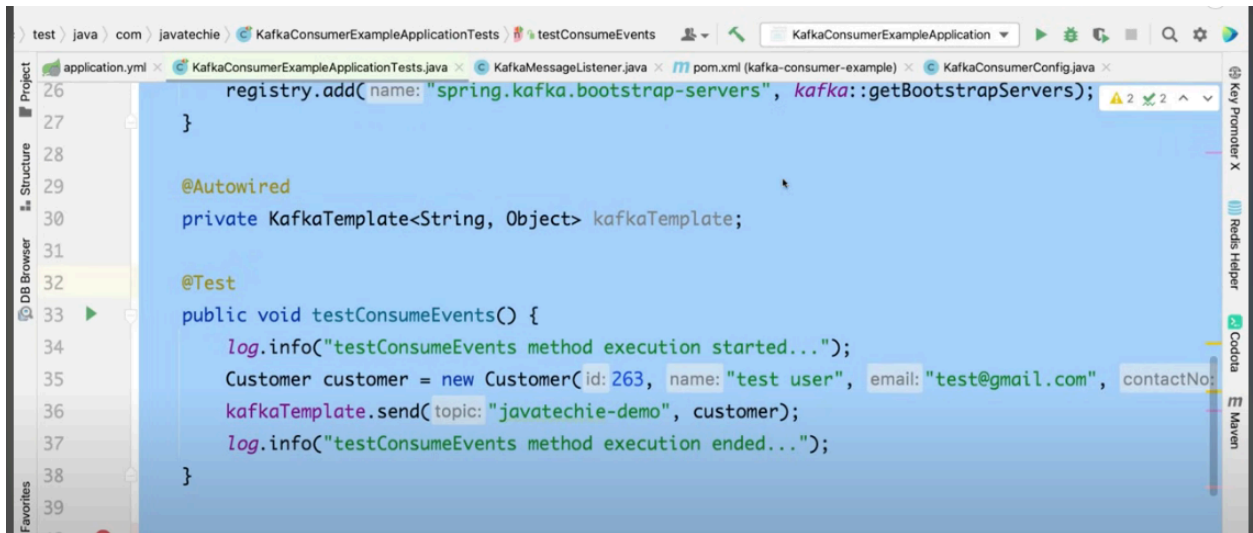


```
45     <groupId>org.testcontainers</groupId>
46     <artifactId>testcontainers</artifactId>
47     <version>1.18.1</version>
48     <scope>test</scope>
49 </dependency>
50 <dependency>
51     <groupId>org.testcontainers</groupId>
52     <artifactId>kafka</artifactId>
53     <version>1.18.1</version>
54     <scope>test</scope>
55 </dependency>
56 <dependency>
57     <groupId>org.testcontainers</groupId>
58     <artifactId>junit-jupiter</artifactId>
59     <version>1.18.1</version>
60     <scope>test</scope>
61 </dependency>
62 <dependency>
63     <groupId>org.awaitility</groupId>
64     <artifactId>awaitility</artifactId>
65     <version>4.2.0</version>
66     <scope>test</scope>
67 </dependency>
```



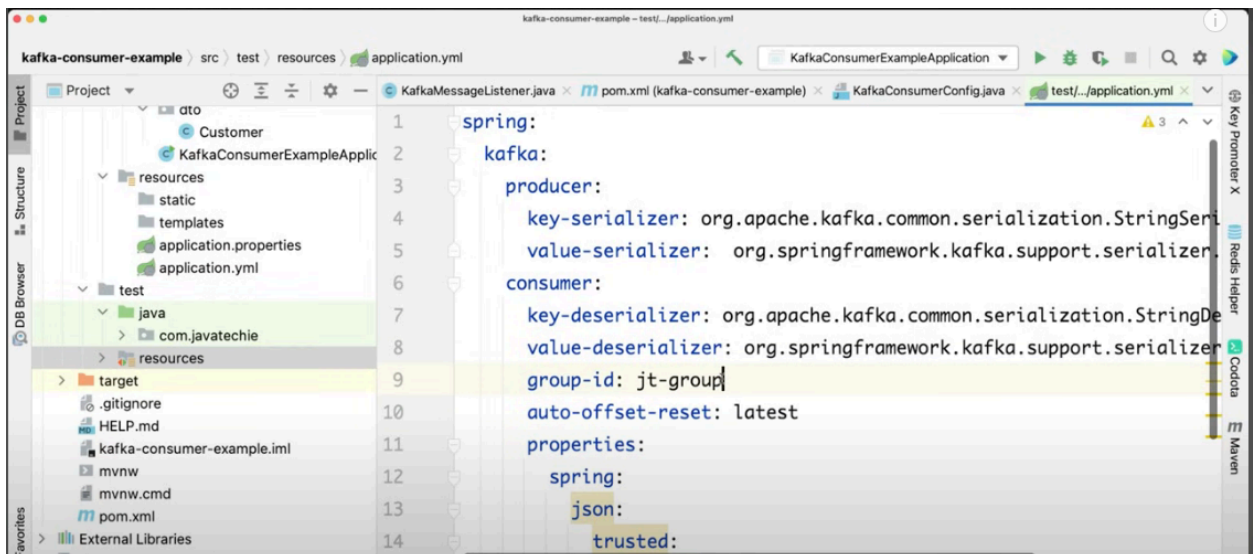
```
9 import org.testcontainers.utility.DockerImageName;
10
11 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
12 @Testcontainers
13 public class KafkaConsumerExampleApplicationTests {
14
15     @Container
16     static KafkaContainer kafka = new KafkaContainer(DockerImageName.parse("confluentinc/cp-kafka:latest"));
17
18     @DynamicPropertySource
19     @org.springframework.test.context.DynamicPropertyRegistry
20     public static void initKafkaProperties(DynamicPropertyRegistry registry) {
21         registry.add(name: "spring.kafka.bootstrap-servers", kafka::getBootstrapServers);
22     }
23 }
```

Above is config and actual test is below

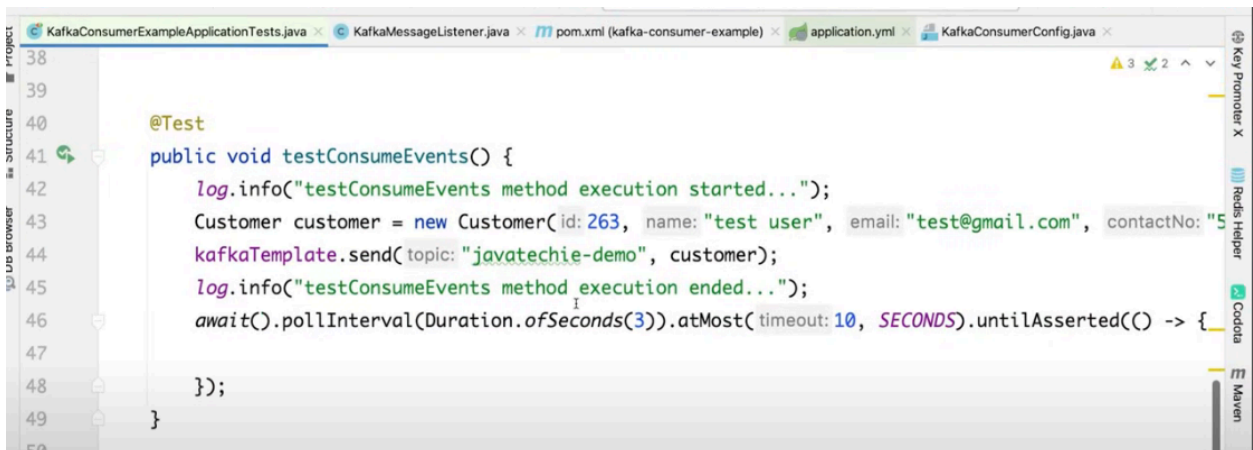


```
26 registry.add(name: "spring.kafka.bootstrap-servers", kafka::getBootstrapServers);
27 }
28
29 @Autowired
30 private KafkaTemplate<String, Object> kafkaTemplate;
31
32 @Test
33 public void testConsumeEvents() {
34     log.info("testConsumeEvents method execution started...");
35     Customer customer = new Customer(id: 263, name: "test user", email: "test@gmail.com", contactNo: "555-555-5555");
36     kafkaTemplate.send(topic: "javatechie-demo", customer);
37     log.info("testConsumeEvents method execution ended...");
38 }
39
```

Create test properties

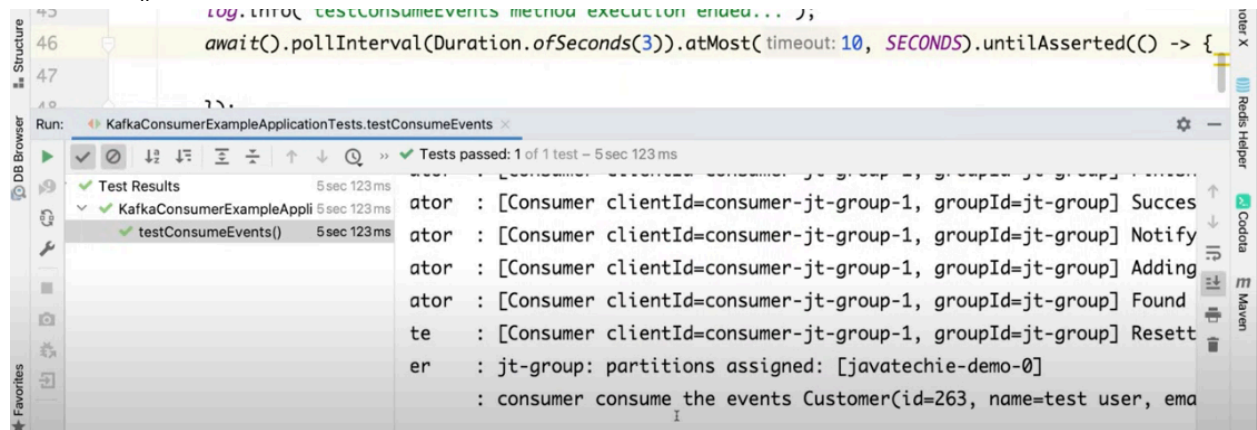


```
1 spring:
2   kafka:
3     producer:
4       key-serializer: org.apache.kafka.common.serialization.StringSerializer
5       value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
6     consumer:
7       key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
8       value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer
9       group-id: jt-group
10      auto-offset-reset: latest
11      properties:
12        spring:
13          json:
14            trusted: true
```



```
38
39
40 @Test
41 public void testConsumeEvents() {
42     log.info("testConsumeEvents method execution started...");
43     Customer customer = new Customer(id: 263, name: "test user", email: "test@gmail.com", contactNo: "555-555-5555");
44     kafkaTemplate.send(topic: "javatechie-demo", customer);
45     log.info("testConsumeEvents method execution ended...");
46     await().pollInterval(Duration.ofSeconds(3)).atMost(timeout: 10, SECONDS).untilAsserted(() -> {
47
48     });
49 }
50
```

Add await() method and run the test.



Refer [Kafka Error Handling Retry Strategies & Dead Letter Topics.pdf](#)

Refer [Kafka Schema Registry.pdf](#)