Lab 11

Part 1

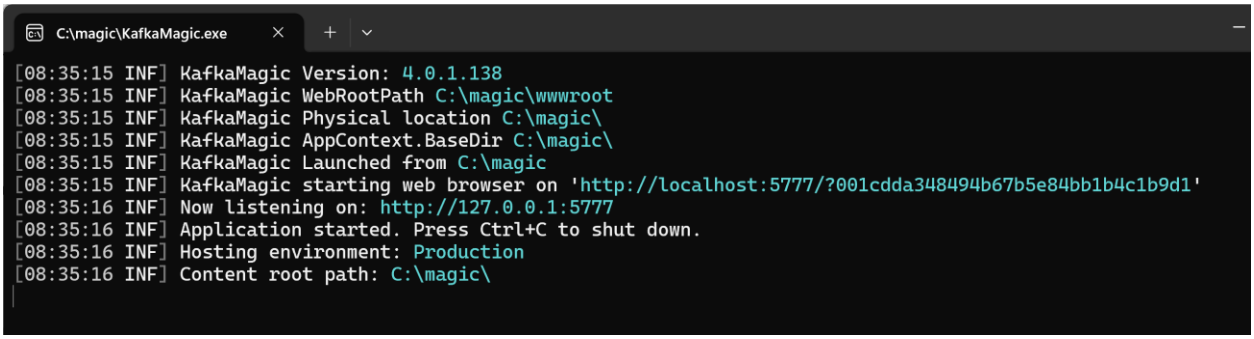
First install and run Kafka with the docker command:

docker run -d --name=kafka -p 9092:9092 apache/kafka

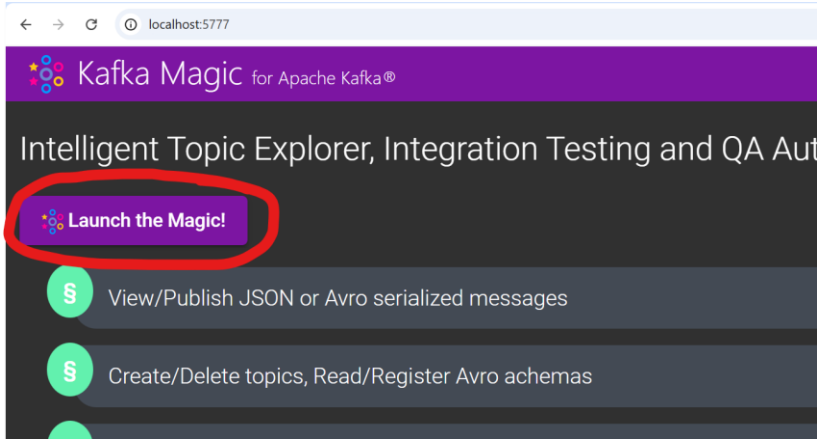
Installing KafkaMagic:

Download (https://www.kafkamagic.com/download/) and install KafkaMagic on your computer

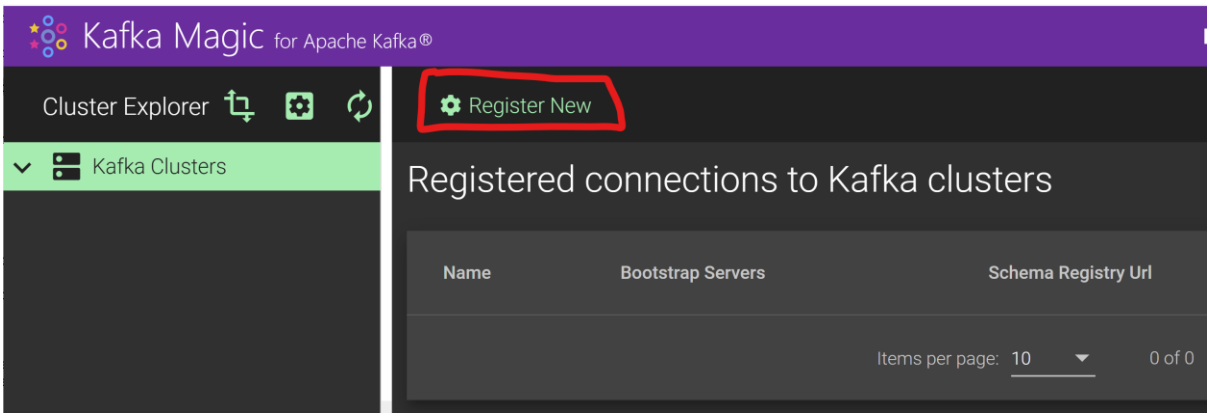
Now start KafkaMagic (run **KafkaMagic.exe** in Windows**)**



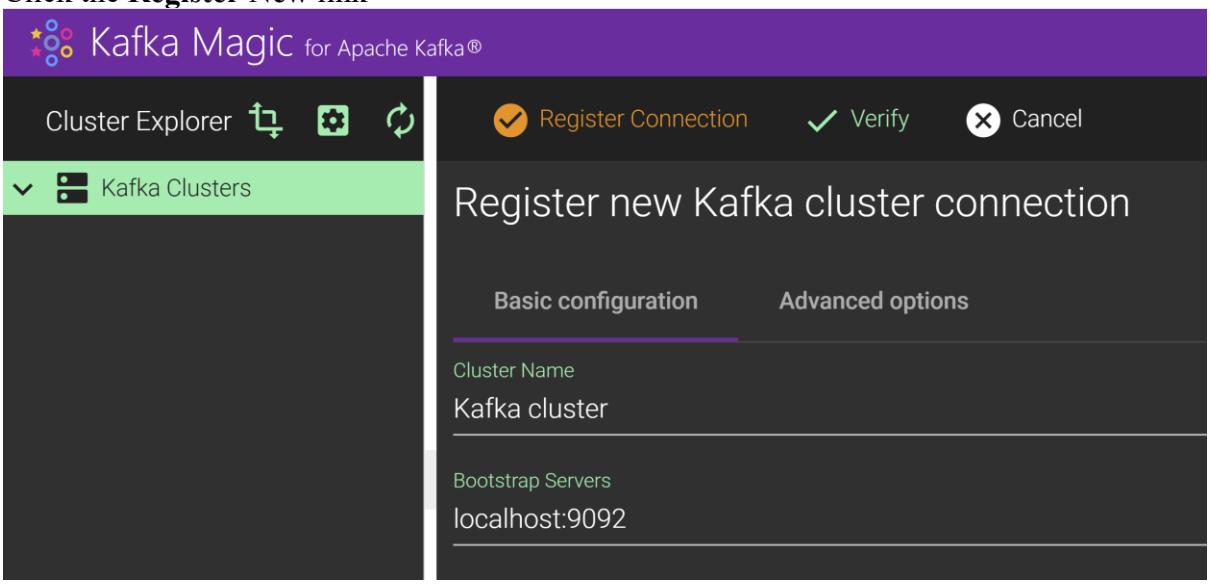
Open in a browser the following URL: **http://localhost:5777**



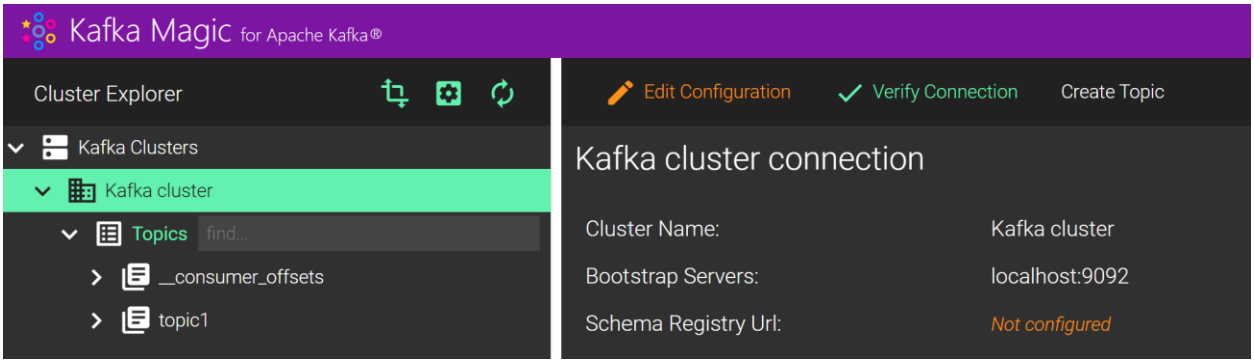
Click the **Launch the Magic!** Button



Click the **Register New** link



Fill in the **Cluser Name** (**Kafka cluster**) and the **Bootstrap Servers (localhost:9092)** and click the **Verify** icon to verify the connection
Then click on the **Register Connection** icon to connect to Kafka



You are now connected to Kafka

a. Write a producer that produces 5 Orders with the fields ordernumber, customername, customercountry, amount and status. Write a consumer that consumes these Order objects
b. Write a new consumer that consumes all published Order objects starting from the first published Order. Let the consumer also print the offset and the groupid to the console.
c. Create a consumer group of 2 Order consumers and test how the Orders are distributed between the consumers. Let the consumer also print the offset and the groupid to the console. Notice that it is always the same consumer who gets the messages.

Part 2

Create a topic with 3 different partitions.

a. Modify the producer from Exercise 1 so that we send the Orders to a topic with 3 partitions. Give all orders the same key. Create 3 different Order consumers (listeners), one for every partition. Let every listener print out the offset and the partition for the messages received.
b. Now give every Order an unique key (the orderNumber) and notice that the Orders are distributed over the partitions

Part 3

a. Using a DefaultErrorHandler write an Order consumer that does the following when an exception occurs in the listener method:
    1. Retry 2 times
    2. Send the message that caused the exception to the DLT and write a listener for this DLT
b. Using the @Retryable annotation write an Order consumer that does the following when an exception occurs in the listener method:
    1. Retry 2 times
    2. Send the message that caused the exception to the DLT and write a listener for this DLT

Part 4

a. Write an Order producer that produces a new Order every 2 seconds. Batch all produced messages for 6 seconds

    Write an Order consumer that consumes the messages and notice that the messages arrive every 6 seconds

b. Write an Order producer that produces a new Order every 2 seconds. (No batch)

    Write an Order consumer that consumes the messages in batches of 6 seconds

Part 5

a. Write a testcase for the Order producer and Order consumer using embedded Kafka
b. Write a testcase for the Order producer and Order consumer using a test container

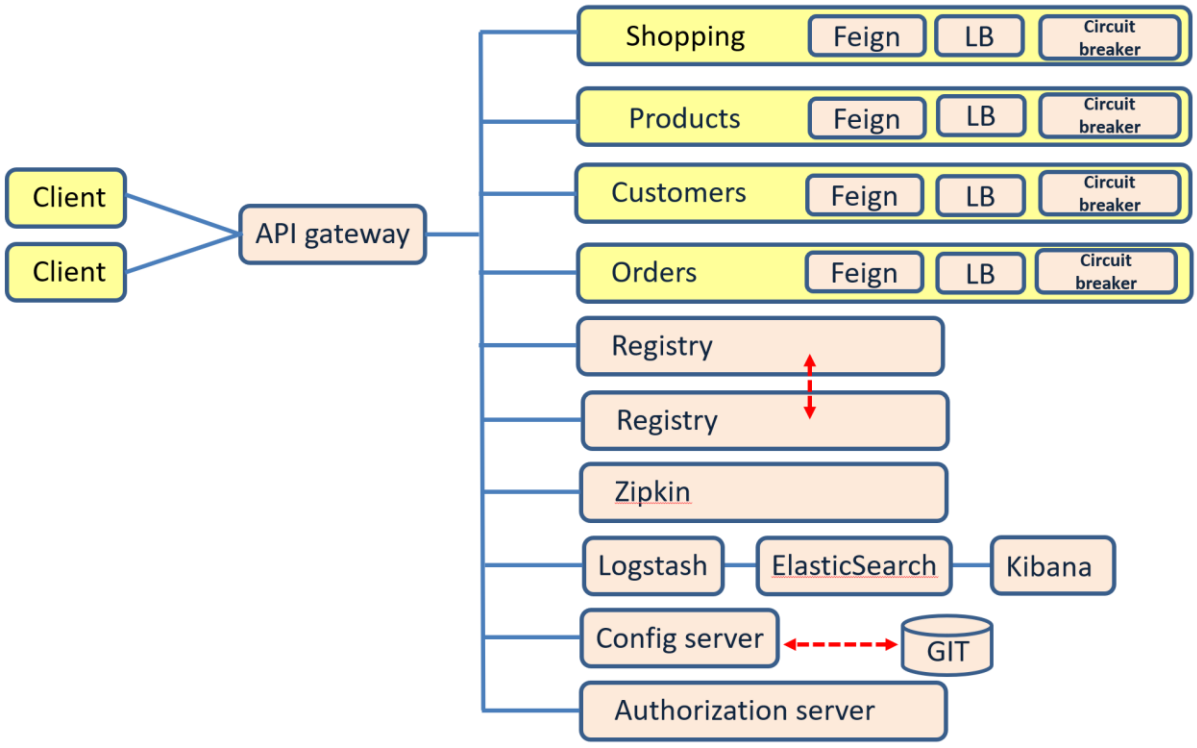
Part 6

Write an Order producer that sends 5 Orders in a transaction. This means all 5 Orders are send to Kafka, or nothing is send in case of an exception.

Write an Order consumer that receives only committed messages.

Part 7

Suppose we have a complete microservice architecture like this:



a. Draw a sequence diagram showing all calls between the different services for the scenario where we start up the different microservices. Let your diagram show in which

order we have to start the microservices, and what other services are called when we startup a certain microservice.

b. Draw a sequence diagram showing all calls between the different applications/services for the following scenario:
   - The client application call the method placeOrder() on the Orders service.
   - The Orders service calls the Products service to update the available products on stock amount.