Scenario: You are building a simplified backend system for an e-commerce platform. When a customer places an order, the order details must be sent to multiple downstream services (e.g., inventory management, billing, and shipping) via Kafka. Each service processes the updates independently, but the updates for any given order must be processed in the correct sequence to ensure consistency across the system.

Requirements:

1. Producer Application (Cart Service):
    - Implement an API endpoint (/create-order) that accepts a request to create a new order.
      This is the same API function from ex1.
    - Implement API endpoint (/update-order) that receives orderId and status fields. The function will update the order with the new status (if the order exists) and notify of the change to all the relevant consumers.

      Order Object Structure:

```
{
  "orderId": "string",          // Unique identifier for the order
  "customerId": "string",       // Unique identifier for the customer
  "orderDate": "ISO8601",       // Timestamp in ISO 8601 format
  "items": [                    // Array of order items, with each item having:
    {
      "itemId": "string",       // Unique identifier for the item
      "quantity": "integer",    // Quantity of the item (must be greater than 0)
      "price": "decimal"        // Price of a single unit of the item
    }
  ],
  "totalAmount": "decimal",     // Total cost of the order (must equal the sum of item prices * quantit
  "currency": "string",         // Currency code (e.g., USD, EUR)
  "status": "string"            // Status of the order (e.g., 'pending', 'confirmed')
}
```

Consumer Application (Order Service):

- The consumer should connect to Kafka and receive **all** order events and handle them according to the correct order they happened.
- Create a new API function (/getAllOrderIdsFromTopic) that receives a topic name as a string and prints all the orderId that were received from that topic
- Upon receiving an order event, the consumer will perform **the same actions as ex1.**

- Implement an API endpoint (/order-details) that gets an orderId input and returns the order details and shipping costs (**same actions as ex1**).

Important instructions:

1. You must handle different types of errors that can be caused by connection issues, brokers not available and return the correct response in the API.
   Keep on mind to use the correct function type in the API (meaning GET/POST/PUT)
2. You must handle the errors with all the tools and approaches we learned.
3. You can write the applications in any language that supports Kafka. you need to use docker container and submit all the applications as described below.
4. for simplicity, you can use the message value format as string and use Jsonconverter locally.
   There will be 10 bonus points if you will use a schema registry with Avro format.
5. Attach a readme file, where you will answer the following questions:
   1. Your full name and ID number
   2. all the topic names you used in the program (producer and consumer) and state their purpose.
   3. What is the key you used in the message and why?
   4. detail the different ways you handled errors and why you choose that way