Pub-Sub System Using Kafka

💻 OVERVIEW

Building a scalable Publisher Subscriber system using **GoLang** and **Apache Kafka** using **Docker**

📋 Why Kafka

1. Low Latency:
2. High Throughput:
3. Fault tolerance:
4. Durability:
5. Scalability:
6. Append Only Logs system:

🔍 BACKGROUND

Pub-Sub is a concept where you want to broadcast the data to N number of consumers at the same time.

Consumers can use these data to facilitate their needs.

Example => In a Payment Gateway System once a transaction is completed you might want to send notifications via Email/SMS to the customer that the transaction is completed.
So for the above use case, we have 2 different consumers.

1. Sending Email
2. Sending SMS

📋 PRODUCT SPECIFICATIONS

The application will consist of the following **modules:**

- **Producer Service**: Producing messages to different topics inside the system.

- **Consumer Service:** Consumers who will consume messages from a given topic.
- **Email Client:** Sending Emails to customers
- **SMS Client:** Sending SMS to customers

🖇 API's

```
POST => /api/v1/produce
```

Description: Build an API Inside Producer service also, this API publishes a message to a given topic.

RequestBody:

```
{
  "request_id": "string",
  "topic_name": "string",
  "message_body": "string",
  "transaction_id": "string",
  "email": "string",
  "phone": "string",
  "customer_id": "string",
  "key": "string",
}
```

💎 FEATURES

1. Create a go lang service using gin-gonic as a framework.
2. The service should have a logging feature that should write logs to file and rotate logs on some logic.
3. Use producer API to produce messages to a given topic in Kafka.
4. Handle subscription of all different consumer groups to a topic, also make sure the logic is written in such a way that it happens during service bootup.
5. There will be 2 consumer groups that will serve different features.

a. Once the first consumer will get the message it will send a **Transaction Successful** email to the email present in the message.
b. The second consumer will send a **Transaction Successful** SMS to the number present in the message.

6. Fault-tolerant system :
    a. Producers will keep on producing the message in cases where the consumers are down. You should be able to process all those pending messages as soon as the consumer is up and running.

🛠️ TECHNICAL SPECIFICATIONS

Technologies to be used to create this application: **GoLang**,**Gin-Gonic**,**Zap**,**Lumberjack**, **Apache Kafka**

- Use **Visual Studio Code** IDE to create this project.
- Plugins for VSCode
    - `Name: Beautify`

      `Id: hookyqr.beautify`
    - `Name: Bracket Pair Colorizer 2`

      `Id: coenraads.bracket-pair-colorizer-2`
    - `Name: Go`

      `Id: ms-vscode.go`
    - `Name: Visual Studio IntelliCode`

      `Id: visualstudioexptteam.vscodeintellicode`

- Build and deploy the service on **Heroku** or somewhere else
- Use the following docker Image to start Kafka on your local

```
---version: "2"
services:
 zk1:
   image: confluentinc/cp-zookeeper:3.0.1
   ports:
     - "22181:22181"
   environment:
     ZOOKEEPER_SERVER_ID: 1
     ZOOKEEPER_CLIENT_PORT: 22181
     ZOOKEEPER_TICK_TIME: 2000
     ZOOKEEPER_INIT_LIMIT: 5
     ZOOKEEPER_SYNC_LIMIT: 2
     ZOOKEEPER_SERVERS: zk1:22888:23888;zk2:32888:33888;zk3:42888:43888

 zk2:
   image: confluentinc/cp-zookeeper:3.0.1
   ports:
     - "32181:32181"
   environment:
     ZOOKEEPER_SERVER_ID: 2
     ZOOKEEPER_CLIENT_PORT: 32181
     ZOOKEEPER_TICK_TIME: 2000
     ZOOKEEPER_INIT_LIMIT: 5
     ZOOKEEPER_SYNC_LIMIT: 2
     ZOOKEEPER_SERVERS: zk1:22888:23888;zk2:32888:33888;zk3:42888:43888

 zk3:
   image: confluentinc/cp-zookeeper:3.0.1
   ports:
     - "42181:42181"
   environment:
     ZOOKEEPER_SERVER_ID: 3
     ZOOKEEPER_CLIENT_PORT: 42181
     ZOOKEEPER_TICK_TIME: 2000
     ZOOKEEPER_INIT_LIMIT: 5
     ZOOKEEPER_SYNC_LIMIT: 2
     ZOOKEEPER_SERVERS: zk1:22888:23888;zk2:32888:33888;zk3:42888:43888

 kafka-1:
   image: confluentinc/cp-kafka:3.0.1
   ports:
     - "19092:19092"
   depends_on:
     - zk1
     - zk2
     - zk3
```

```
  environment:
    KAFKA_BROKER_ID: 1
    KAFKA_ZOOKEEPER_CONNECT: ${MY_IP}:22181,${MY_IP}:32181,${MY_IP}:42181
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://${MY_IP}:19092

kafka-2:
  image: confluentinc/cp-kafka:3.0.1
  ports:
    - "29092:29092"
  depends_on:
    - zk1
    - zk2
    - zk3
  environment:
    KAFKA_BROKER_ID: 2
    KAFKA_ZOOKEEPER_CONNECT: ${MY_IP}:22181,${MY_IP}:32181,${MY_IP}:42181
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://${MY_IP}:29092

kafka-3:
  image: confluentinc/cp-kafka:3.0.1
  ports:
    - "39092:39092"
  depends_on:
    - zk1
    - zk2
    - zk3
  environment:
    KAFKA_BROKER_ID: 3
    KAFKA_ZOOKEEPER_CONNECT: ${MY_IP}:22181,${MY_IP}:32181,${MY_IP}:42181
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://${MY_IP}:39092
```

- You can use any Go library to send SMS/Email.
- Basic Docker Commands and links to read about Kafka
  https://medium.com/@yusufs/getting-started-with-kafka-in-golang-14ccab5fa26

  https://docs.docker.com/compose/reference/overview/

🌟GOOD TO HAVE FEATURES

List of features that are not mandatory but will give you extra points if built.

- Writing Unit test Cases with the test coverage of more than 85%
- Using Design Principles while writing code
- Using Mocks for testing wherever needed
- Reading Configuration using **viper**
- How well you are running your consumers, good if you are running as a go-routine

🏆 MILESTONES & EVALUATION

This problem statement carries a total of **100 points**.

1. Learn the basics of Go-Lang, Apache Kafka, and its usage **(10 pts)**
2. Learn the basics of docker and some useful commands **(5 pts)**
3. Use the above-mentioned docker-compose file to start Kafka instance **(5 pts)**
4. Create a hello world go lang service and deploy on Heroku/AWS **(10 pts)**
5. Creating Producer service
    a. Publishing message to a given topic using key **(10 pts)**
    b. Publishing message on without any key **(10 pts)**
    c. Publishing message to a topic using partition number **(10 pts)**
6. Creating consumer service
    a. Consuming message and sending email to customers **(10 pts)**
    b. Consuming message and sending SMS to customers **(10 pts)**
7. Making a fault-tolerant system
    a. Handling message when the Mail server is down **(10 pts)**
    b. Handling message when SMS client is down **(10 pts)**
    c. Handling message when the entire consumer is down **(10 pts)**

👨💻 ADVISE FROM PROFESSOR

Spend the first 2-3 days in learning and brushing up your GoLang skillset. If you are already proficient in, you can directly start working on the problem.

Links to read

1. https://medium.com/@yusufs/getting-started-with-kafka-in-golang-14ccab5fa26
2. https://medium.com/@dablu/our-first-microservice-in-golang-using-gin-gonic-as-framework-4db155e46fc6

📨 QUESTIONS

In case of any concerns, reach out to me at **professor@zestmoney.in**