# System Overview

The system is designed to ensure reliable message delivery from a queue to an API endpoint. It consists of two microservices:

## Microservice-1:

* Acts as a consumer of messages from the queue.
* Forwards the messages to Microservice-2 via a POST API.
* Implements a retry mechanism to ensure no message is lost in case of delivery failures.

## Microservice-2:

* Exposes a POST API to receive messages from Microservice-1.
* Processes and stores the received data.

## Flow Diagram

**Key Components:**

1. **Queue (Kafka)**:

* Stores messages in **FIFO** order.
* Acts as a temporary buffer to decouple message producers from Microservice-1.

1. **Microservice-1**:

* Consumes messages from the queue.
* Posts messages to Microservice-2.
* Retries indefinitely every 10 seconds if delivery fails.

1. **Microservice-2**:

* Processes messages received via the POST API.
* Stores data in the PostgreSQL database.

## System Design Flow

**High-Level Flow:**

1. A **producer** sends messages to the queue (my-topic).
2. **Microservice-1**:

* Consumes messages from the queue.
* Sends the message payload to the POST API of Microservice-2.
* If the API call fails, retries every 10 seconds until successful.

1. **Microservice-2**:

* Receives the message via the API.
* Validates and stores the message in the PostgreSQL database.

## Detailed Design

**Microservice-1 (Queue Consumer and Forwarder)**

1. **Queue Consumption**:

* Listens to the Kafka topic (my-topic).
* Processes messages in the order they are received.

1. **Message Forwarding**:

* Sends the message payload as a POST request to Microservice-2.

1. **Retry Mechanism**:

* If the POST API fails:
  + The message is not acknowledged or Not marked as "done."
  + A retry is scheduled after a 10-second delay.
* Retries continue indefinitely until the message is successfully delivered.

1. **Failure Handling**:

* Logs failed attempts for debugging.
* Persists messages in a local database (optional) for enhanced reliability.

**Microservice-2 (API Receiver)**

1. **POST API**:

* Endpoint: /api/data.
* Validates the JSON payload.

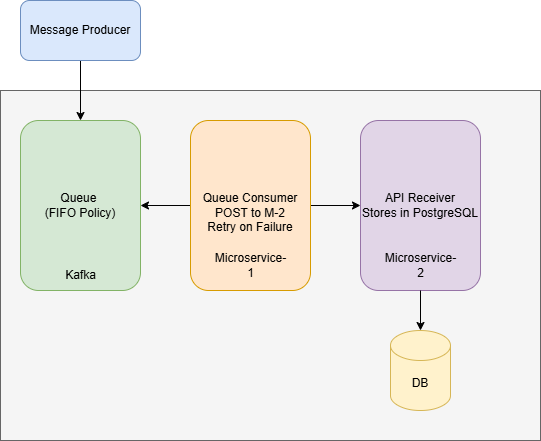
1. **Data Processing**:

* Inserts the validated data into the PostgreSQL table received\_messages.

1. **Response Codes**:

* **200 OK**: Message received and stored successfully.
* **400 Bad Request**: Invalid payload.
* **500 Internal Server Error**: Issues with data processing or database.

## Flow Diagram



**Key Features**

1. **Retry Mechanism**:

* Ensures no messages are lost.
* Retries every 10 seconds indefinitely.

1. **FIFO Processing**:

* Ensures messages are processed in the order they are received from the queue.

1. **Data Persistence**:

* Ensures all processed data is stored reliably in PostgreSQL.

**Technologies**

| **Component** | **Technology** |
| --- | --- |
| **Queue** | Kafka |
| **Microservice-1** | Golang (Kafka Consumer) |
| **Microservice-2** | Golang (REST API) |
| **Database** | PostgreSQL |
| **Retry Logic** | Custom retry mechanism in Golang |

**Benefits of the Design**

* **Reliability**: No data loss due to the retry mechanism.
* **Scalability**: Kafka and Microservices can scale horizontally.
* **Extensibility**: Supports multiple queue systems (Kafka).
* **Modularity**: Decouples message production, consumption, and processing.