**Create Topics**:

delivery\_requests for customer shipment requests.

delivery\_responses for responses from logistics to clients.

kafka-topics.sh --create --topic delivery\_requests --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1 kafka-topics.sh --create --topic delivery\_responses --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

use a cloud-based MongoDB service.

**Define Collections**:

customers (to store customer details)

* + shipments (to store shipment requests and statuses)

*// Example schema for customers* { "firstName": "John", "lastName": "Doe", "contactNumber": "123-456-7890" }

*// Example schema for shipments* { "customerId": "customer\_id", "type": "express", "pickupLocation": "location1", "deliveryLocation": "location2", "preferredTimeSlot": "2023-10-01T10:00:00", "status": "pending" }

**Implement Ballerina Services**

**Create Central Logistics Service**:

Listens to delivery\_requests, processes requests, and forwards them to the appropriate delivery service.

**Create Delivery Services** (Standard, Express, International):

Each service listens to messages pertinent to their type and processes them accordingly.

*// Example Ballerina service for Central Logistics*

import ballerina/http;

import ballerina/kafka;

import ballerina/mongodb;

service /logistics on new http:Listener(8080) { *// Kafka config* kafka:Consumer deliveryConsumer = check new (kafka:ConsumerConfig { groupId: "logistics\_group", bootstrapServers: "localhost:9092", topics: ["delivery\_requests"] });

*// MongoDB config* mongodb:Client dbClient = check new ("mongodb://localhost:27017", "logistics");

*// Function to process incoming delivery requests* function processRequests() returns error? { *// Consume messages and process requests* *// Interact with MongoDB to store the shipment* *// Publish to specific delivery services* } }

**Define the Delivery Services** in a similar manner with specific logic based on shipment type.

**Step 4: Dockerization**

**Create Dockerfile for Each Service**

1. *# Dockerfile for each Ballerina service*
2. FROM ballerina/jdk8
3. COPY . /home/ballerina/
4. WORKDIR /home/ballerina/ CMD ["ballerina", "run", "logistics\_service.bal"]**vice**:

**Create a Docker Compose File**:

version: '3' services: kafka: image: wurstmeister/kafka ports: - "9092:9092" environment: KAFKA\_ADVERTISED\_LISTENERS: INSIDE://kafka:9092,OUTSIDE://localhost:9092 KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT KAFKA\_LISTENERS: INSIDE://0.0.0.0:9092,OUTSIDE://0.0.0.0:9092 depends\_on: - zookeeper zookeeper: image: wurstmeister/zookeeper ports: - "2181:2181" mongo: image: mongo ports: - "27017:27017" logistics\_service: build: context: ./logistics ports: - "8080:8080" standard\_service: build: context: ./standard ports: - "8081:8080" express\_service: build: context: ./express ports: - "8082:8080" international\_service: build: context: ./international ports: - "8083:8080"

**Start the Docker Containers**:

docker-compose up –build

ballerina use the components and how can we use kafka to comminicate through messages and store them in mongo

*# Creating Kafka topics using the command line* kafka-topics.sh --create --topic requestTopic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 *# Repeat for other topics...*

**Ballerina Service to Produce Messages to Kafka**

import ballerina/io; import ballerina/kafka; service /logisticsService { kafka:Producer producer = new (kafka:ProducerConfig { bootstrapServers: "localhost:9092" }); *// Sample resource for incoming requests* resource function post request(json requestDetails) returns error? { *// Send request to Kafka* var result = producer->send("requestTopic", requestDetails); if (result is error) { return result; *// Handle error appropriately* } io:println("Request sent to Kafka successfully: ", requestDetails); } }

### **Ballerina Service to Consume Messages from Kafka**

import ballerina/io; import ballerina/kafka; import ballerina/mongo; service /deliveryService { kafka:Consumer consumer = new (kafka:ConsumerConfig { bootstrapServers: "localhost:9092", groupId: "deliveryGroup", autoOffsetReset: "earliest" }); mongo:Client mongoClient = new ("mongodb://localhost:27017", "logisticsDB"); *// Listen for incoming messages in the 'requestTopic'* resource function onMessage(kafka:ConsumerRecord record) returns error? { json receivedRequest = check <json> record.value; io:println("Received delivery request: ", receivedRequest); *// Store the request in MongoDB* var insertResult = mongoClient->insert("deliveryRequests", receivedRequest); if (insertResult is error) { io:println("Error inserting into MongoDB: ", insertResult); } else { io:println("Request stored in MongoDB successfully."); } *// Process request (implement logic to interact with delivery services)* } }

You will need to configure MongoDB and create a suitable schema. In the Ballerina service above, we use a MongoDB client to store incoming request details directly into the deliveryRequests collection.

Make sure your MongoDB instance is properly set up and running:

docker run -d -p 27017:27017 --name mongodb mongo

### Deployment

* **Dockerize Each Service**: Create a Dockerfile for each Ballerina service to run them in containers.
* **Use Docker Compose**: You can create a docker-compose.yml file to define how to run the entire stack, including Kafka, MongoDB, and the Ballerina services.

FROM ballerina/ballerina:latest COPY ./your\_service.bal /home/ballerina/ CMD ["ballerina", "run", "/home/ballerina/your\_service.bal"]