# Kafka Chatbox Application

This project is a practice implementation of Apache Kafka using a Spring Boot Java-based application.

It consists of two servers communicating with each other via Kafka, designed as a chatbox application.

# Features

Two servers (user-side and client-side) capable of producing and consuming messages using Kafka.

Dockerized Kafka setup for seamless integration.

Chatbox functionality with a basic front-end for message exchange.

Separate pages for user ("Abhi") and client communication.

# How It Works

Kafka Setup:

Kafka is installed and run using Docker.

Both servers communicate over a single Kafka topic.

# Front-End Interaction:

The front-end is built using HTML, CSS, and JavaScript with assistance from ChatGPT.

Users can open the respective URLs in separate browser tabs to simulate chat functionality.

# Usage:

# Abhi (User-side):

Open the following URL in your browser:

http://localhost:63342/kafkaproducer/kafkaproducer/kafkaproducer/frontend/index.html?

\*if not working simply click on top right side of html file.\*

# Client-side:

Open the following URL in another browser tab:

http://localhost:63342/kafkaconsumer/kafkaconsumer/kafkaconsumer/frontend/client.html?

\*if not working simply click on top right side of html file.\*

In each tab, you will see a chatbox.

Write a message and click Send to produce messages.

Use GetReply to consume messages from the other side.

Technologies Used

Java (Spring Boot)

Apache Kafka

Docker (Kafka setup)

HTML, CSS, and JavaScript (Front-end)

Setup and Installation

Pre-requisites:

Install Docker.

Set up Kafka using Docker.

Steps to Run the Application:

Clone the repository.

Start Kafka using Docker.

Run both the producer and consumer servers.

Open the provided URLs in separate browser tabs.

Acknowledgments

Front-end design was assisted by ChatGPT.

This project was developed as a learning exercise to explore Kafka and Spring Boot.

\*thank you\*