Kafka Chat Application - Hands-On

# 1. Kafka & Zookeeper Setup

Make sure Java is installed. Then download and extract Kafka. Open two terminals:

* Start Zookeeper:  
  > .\zookeeper-server-start.bat ..\..\config\zookeeper.properties
* Start Kafka:  
  > .\kafka-server-start.bat ..\..\config\server.properties

# 2. Create Kafka Topic

* In a new terminal:  
  > .\kafka-topics.bat --create --topic chat-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

# 3. Visual Studio Code Setup

Create a new console project and install the Kafka NuGet package:

> dotnet new console -n KafkaChatApp  
> cd KafkaChatApp  
> dotnet add package Confluent.Kafka

# 4. Program.cs (Combined Sender & Receiver)

using System;  
using System.Threading;  
using System.Threading.Tasks;  
using Confluent.Kafka;  
  
class Program  
{  
 static async Task Main(string[] args)  
 {  
 Console.WriteLine("Kafka Chat App");  
 Console.WriteLine("Choose mode: 1 = Sender, 2 = Receiver");  
 var choice = Console.ReadLine();  
  
 if (choice == "1")  
 {  
 await RunProducer();  
 }  
 else if (choice == "2")  
 {  
 RunConsumer();  
 }  
 else  
 {  
 Console.WriteLine("Invalid choice. Exiting.");  
 }  
 }  
  
 static async Task RunProducer()  
 {  
 var config = new ProducerConfig { BootstrapServers = "localhost:9092" };  
 using var producer = new ProducerBuilder<Null, string>(config).Build();  
  
 Console.WriteLine("You are now sending messages. Type and press Enter:");  
  
 while (true)  
 {  
 Console.Write("You: ");  
 var message = Console.ReadLine();  
 if (message == "exit") break;  
 await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = message });  
 }  
 }  
  
 static void RunConsumer()  
 {  
 var config = new ConsumerConfig  
 {  
 GroupId = "chat-group",  
 BootstrapServers = "localhost:9092",  
 AutoOffsetReset = AutoOffsetReset.Earliest  
 };  
  
 using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();  
 consumer.Subscribe("chat-topic");  
  
 Console.WriteLine("Listening for messages (Press Ctrl+C to stop)...");  
  
 var cts = new CancellationTokenSource();  
 Console.CancelKeyPress += (\_, e) => {  
 e.Cancel = true;  
 cts.Cancel();  
 };  
  
 try  
 {  
 while (!cts.Token.IsCancellationRequested)  
 {  
 var msg = consumer.Consume(cts.Token);  
 Console.WriteLine($"Friend: {msg.Message.Value}");  
 }  
 }  
 catch (OperationCanceledException)  
 {  
 Console.WriteLine("Exiting consumer...");  
 }  
 finally  
 {  
 consumer.Close();  
 }  
 }  
}

# 5. Running the App

Open two terminals in VS Code:

* Terminal 1:  
  > dotnet run  
  Choose: 1
* Terminal 2:  
  > dotnet run  
  Choose: 2

# 6. Output:



