WEB API HANDSON

KafkaChatApp.csproj

<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>

    <OutputType>Exe</OutputType>

    <TargetFramework>net8.0</TargetFramework>

  </PropertyGroup>

  <ItemGroup>

    <PackageReference Include="Confluent.Kafka" Version="1.9.2" />

  </ItemGroup>

</Project>

Program.cs

using System;

using System.Threading;

using System.Threading.Tasks;

using Confluent.Kafka;

class Program

{

    static async Task Main(string[] args)

    {

        Console.WriteLine("Choose mode: (1) Producer, (2) Consumer");

        var input = Console.ReadLine();

        if (input == "1")

        {

            await RunProducer();

        }

        else if (input == "2")

        {

            RunConsumer();

        }

        else

        {

            Console.WriteLine("Invalid choice");

        }

    }

    static async Task RunProducer()

    {

        var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

        using var producer = new ProducerBuilder<Null, string>(config).Build();

        Console.WriteLine("Enter messages to send to Kafka (type 'exit' to quit):");

        string? message;

        while ((message = Console.ReadLine()) != "exit")

        {

            var result = await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = message });

            Console.WriteLine($"Sent: {message} to Partition: {result.Partition}");

        }

    }

    static void RunConsumer()

    {

        var config = new ConsumerConfig

        {

            BootstrapServers = "localhost:9092",

            GroupId = "chat-consumer-group",

            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 exit)...");

        CancellationTokenSource cts = new();

        Console.CancelKeyPress += (\_, e) => { e.Cancel = true; cts.Cancel(); };

       try

        {

            while (true)

            {

                var cr = consumer.Consume(cts.Token);

                Console.WriteLine($"Received: {cr.Message.Value}");

            }

        }

        catch (OperationCanceledException)

        {

            Console.WriteLine("Closing consumer...");

            consumer.Close();

        }

    }

}

OUTPUT:



