1. **WebApi\_Handson**
2. **Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prompt.**

**CODE:**

**Producer.cs**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class Producer

{

public static async Task SendMessagesAsync()

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

Console.WriteLine("Enter messages to send. Type 'exit' to quit.");

while (true)

{

Console.Write("You: ");

var message = Console.ReadLine();

if (message.ToLower() == "exit") break;

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

}

producer.Flush(TimeSpan.FromSeconds(2));

}

}

**Consumer.cs**

using Confluent.Kafka;

using System;

class Consumer

{

public static void StartListening()

{

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

GroupId = "chat-group",

AutoOffsetReset = AutoOffsetReset.Earliest

};

using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();

consumer.Subscribe("chat-topic");

Console.WriteLine("Listening for messages...\n");

while (true)

{

var cr = consumer.Consume();

Console.WriteLine("Received: " + cr.Message.Value);

}

}

}

**OUTPUT:**

1. **Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.**

**CODE:**

// KafkaConfig.cs

public static class KafkaConfig

{

public const string BootstrapServers = "localhost:9092"; // Change if needed

public const string Topic = "chat-topic";

}

**Form1.cs**

using Confluent.Kafka;

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KafkaChatApp

{

public partial class Form1 : Form

{

private IProducer<Null, string> \_producer;

private CancellationTokenSource \_cts;

public Form1()

{

InitializeComponent();

InitializeKafka();

StartConsumer();

}

private void InitializeKafka()

{

var config = new ProducerConfig { BootstrapServers = KafkaConfig.BootstrapServers };

\_producer = new ProducerBuilder<Null, string>(config).Build();

}

private async void btnSend\_Click(object sender, EventArgs e)

{

string user = Environment.UserName;

string message = $"{user}: {txtMessage.Text}";

await \_producer.ProduceAsync(KafkaConfig.Topic, new Message<Null, string> { Value = message });

txtMessage.Clear();

}

private void StartConsumer()

{

\_cts = new CancellationTokenSource();

Task.Run(() =>

{

var consumerConfig = new ConsumerConfig

{

BootstrapServers = KafkaConfig.BootstrapServers,

GroupId = Guid.NewGuid().ToString(), // each client gets its own group

AutoOffsetReset = AutoOffsetReset.Earliest

};

using var consumer = new ConsumerBuilder<Ignore, string>(consumerConfig).Build();

consumer.Subscribe(KafkaConfig.Topic);

try

{

while (!\_cts.Token.IsCancellationRequested)

{

try

{

var cr = consumer.Consume(\_cts.Token);

Invoke(new Action(() =>

{

lstMessages.Items.Add(cr.Message.Value);

}));

}

catch (ConsumeException ex)

{

MessageBox.Show($"Consume error: {ex.Message}");

}

}

}

catch (OperationCanceledException)

{

consumer.Close();

}

}, \_cts.Token);

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

\_cts.Cancel();

\_producer.Dispose();

base.OnFormClosing(e);

}

}

}