**Name: Deepanshu Rathore**

**Superset ID: 6358199**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Week 5  
**6. WebApi\_Handson**

Producer  
Program.cs  
using System;

using System.Threading.Tasks;

using Confluent.Kafka;

namespace KafkaChat.Console.Producer

{

class Program

{

static async Task Main(string[] args)

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

Console.WriteLine("Type messages to send to Kafka. Enter blank line to exit.");

while (true)

{

var line = Console.ReadLine();

if (string.IsNullOrWhiteSpace(line)) break;

try

{

var result = await producer.ProduceAsync(

topic: "chat",

new Message<Null, string> { Value = line });

Console.WriteLine($"Delivered '{result.Value}' to {result.TopicPartitionOffset}");

}

catch (ProduceException<Null, string> e)

{

Console.WriteLine($"Delivery failed: {e.Error.Reason}");

}

}

}

}

}  
  
Consumer  
Program.cs:  
using System;

using Confluent.Kafka;

namespace KafkaChat.Console.Consumer

{

class Program

{

static void Main(string[] args)

{

var config = new ConsumerConfig

{

GroupId = "chat-consumers",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("chat");

Console.WriteLine("Listening for chat messages. Press Ctrl+C to quit.");

try

{

while (true)

{

var cr = consumer.Consume();

Console.WriteLine($"[{cr.TopicPartitionOffset}] {cr.Message.Value}");

}

}

catch (OperationCanceledException)

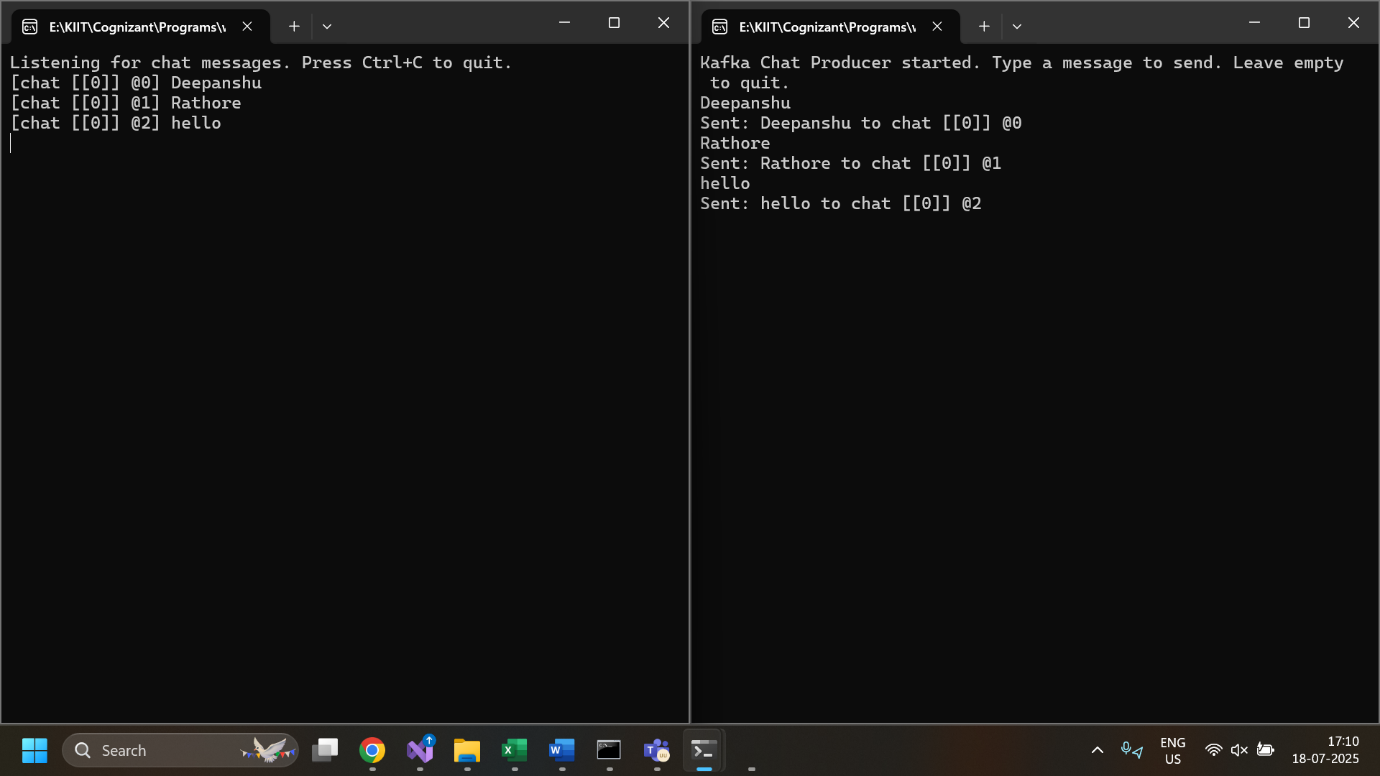
{

consumer.Close();

}

}

}

}  
**Output:**   
ProducerForm.cs:

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

using Confluent.Kafka;

using KafkaChat.Common;

public partial class ProducerForm : Form

{

private readonly IProducer<Null, string> \_producer;

public ProducerForm()

{

InitializeComponent();

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

}

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

{

var msg = txtMessage.Text;

if (string.IsNullOrWhiteSpace(msg)) return;

await \_producer.ProduceAsync("chat", new Message<Null, string> { Value = msg });

txtMessage.Clear();

}

}  
ConsumerForm.cs:  
using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using Confluent.Kafka;

using KafkaChat.Common;

public partial class ConsumerForm : Form

{

private readonly IConsumer<Ignore, string> \_consumer;

private CancellationTokenSource \_cts = new();

public ConsumerForm()

{

InitializeComponent();

\_consumer = new ConsumerBuilder<Ignore, string>(KafkaHelper.ConsumerConfig).Build();

\_consumer.Subscribe("chat");

Task.Run(() => ConsumeLoop(\_cts.Token));

}

private void ConsumeLoop(CancellationToken token)

{

try

{

while (!token.IsCancellationRequested)

{

var cr = \_consumer.Consume(token);

Invoke((Action)(() =>

lstMessages.Items.Add($"{cr.Message.Value}")));

}

}

catch (OperationCanceledException) { }

}

private void ConsumerForm\_FormClosing(object sender, FormClosingEventArgs e)

{

\_cts.Cancel();

\_consumer.Close();

}

}

A screenshot of a computer

AI-generated content may be incorrect.