**Kafka Integration with C#:**

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

KafkaProducerApp/program.cs:

using Confluent.Kafka;

class Program

{

static async Task Main()

{

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

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

Console.WriteLine("Kafka Producer started. Type messages (type 'exit' to quit):");

while (true)

{

var message = Console.ReadLine();

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

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

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

}

}

}

KafkaConsumerApp/program.cs:

using Confluent.Kafka;

class Program

{

static void Main()

{

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-room");

Console.WriteLine("Kafka Consumer started. Listening for messages...");

while (true)

{

var cr = consumer.Consume();

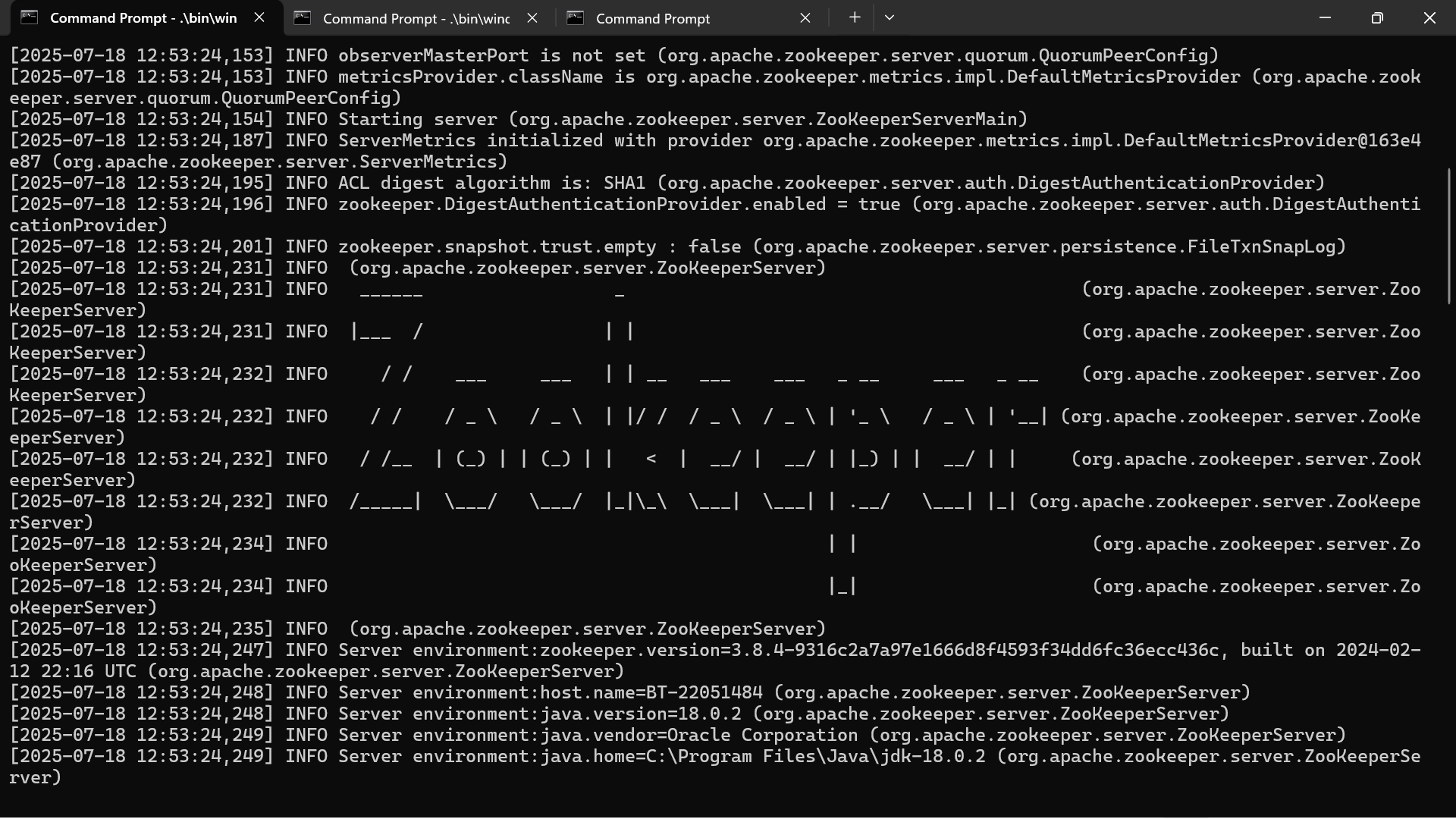
Console.WriteLine($"Received: {cr.Message.Value}");

}

}

}

OUTPUT:

A screenshot of a computer screen

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

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

Form1.cs:

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KafkaChatWinApp

{

public partial class Form1 : Form

{

private IProducer<Null, string>? producer;

private IConsumer<Null, string>? consumer;

private readonly string topic = "chat-topic";

private readonly string clientId = "User-" + Guid.NewGuid().ToString().Substring(0, 4);

private bool isRunning = true;

public Form1()

{

InitializeComponent();

this.Text = $"Kafka Chat - {clientId}";

InitializeKafka();

}

private void InitializeKafka()

{

try

{

var producerConfig = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

producer = new ProducerBuilder<Null, string>(producerConfig).Build();

var consumerConfig = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

GroupId = Guid.NewGuid().ToString(),

AutoOffsetReset = AutoOffsetReset.Latest

};

consumer = new ConsumerBuilder<Null, string>(consumerConfig).Build();

consumer.Subscribe(topic);

Task.Run(() => ListenForMessages());

}

catch (Exception ex)

{

MessageBox.Show("Kafka initialization failed: " + ex.Message);

}

}

private void ListenForMessages()

{

try

{

while (isRunning)

{

try

{

var consumeResult = consumer?.Consume();

if (consumeResult?.Message?.Value != null)

{

AppendMessageToList(consumeResult.Message.Value);

}

}

catch (ConsumeException ex)

{

AppendMessageToList($"⚠ Kafka error: {ex.Error.Reason}");

}

}

}

catch (Exception ex)

{

AppendMessageToList("Error in consumer thread: " + ex.Message);

}

}

private void AppendMessageToList(string message)

{

if (lstMessages.InvokeRequired)

{

lstMessages.Invoke(new Action(() => lstMessages.Items.Add(message)));

}

else

{

lstMessages.Items.Add(message);

}

}

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

{

string userMessage = txtMessage.Text.Trim();

if (!string.IsNullOrEmpty(userMessage) && producer != null)

{

string fullMessage = $"{clientId}: {userMessage}";

try

{

await producer.ProduceAsync(topic, new Message<Null, string> { Value = fullMessage });

txtMessage.Clear();

}

catch (ProduceException<Null, string> ex)

{

AppendMessageToList($"⚠ Failed to send: {ex.Message}");

}

}

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

isRunning = false;

consumer?.Close();

consumer?.Dispose();

producer?.Flush(TimeSpan.FromSeconds(5));

producer?.Dispose();

base.OnFormClosing(e);

}

private void lstMessages\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

OUTPUT:

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.

