**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(string[] args)

{

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

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

Console.WriteLine("Kafka Chat Producer Started...");

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

while (true)

{

var message = Console.ReadLine();

if (message == "exit") break;

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

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

}

}

}

KafkaConsumerApp2 – *Program.cs*

using Confluent.Kafka;

class Program

{

static void Main(string[] args)

{

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("Kafka Chat Consumer Started. Waiting for messages...");

while (true)

{

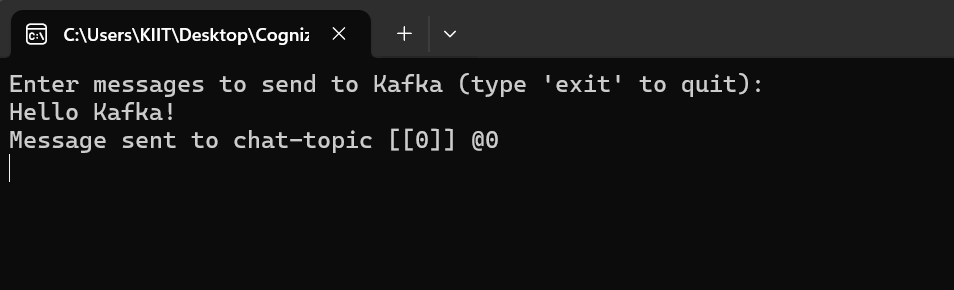
var consumeResult = consumer.Consume();

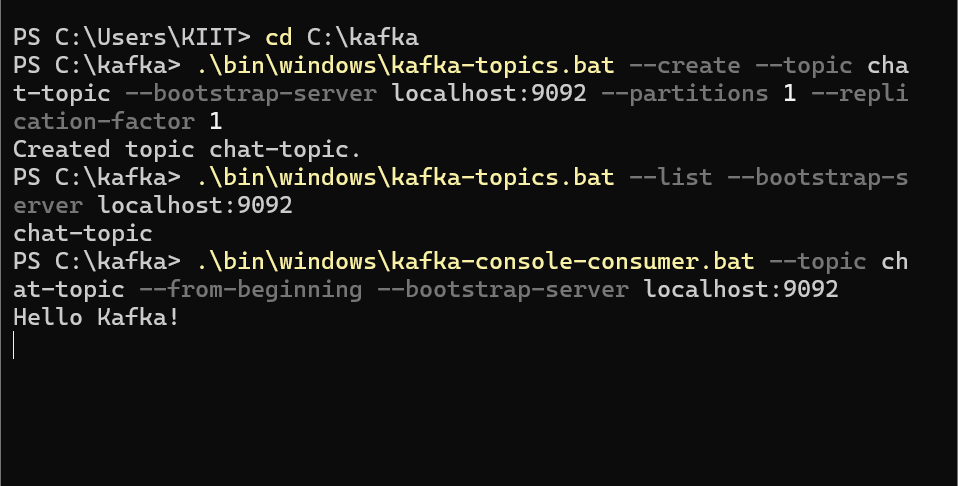
Console.WriteLine($"Received: {consumeResult.Message.Value}");

}

}

}





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

Form1.cs (Windows Form Logic)

using Confluent.Kafka;

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KafkaChatWindowsApp

{

public partial class Form1 : Form

{

private readonly string topic = "chat-topic";

private readonly string bootstrapServers = "localhost:9092";

private CancellationTokenSource cts;

public Form1()

{

InitializeComponent();

StartConsumer();

}

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

{

var config = new ProducerConfig { BootstrapServers = bootstrapServers };

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

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

txtMessage.Clear();

}

private void StartConsumer()

{

cts = new CancellationTokenSource();

Task.Run(() =>

{

var config = new ConsumerConfig

{

BootstrapServers = bootstrapServers,

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

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe(topic);

try

{

while (!cts.Token.IsCancellationRequested)

{

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

Invoke(new Action(() =>

{

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

}));

}

}

catch (OperationCanceledException) { }

});

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

cts.Cancel();

base.OnFormClosing(e);

}

}

}

Producer Code (Send Button Click Handler):

using Confluent.Kafka;

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

{

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

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

{

var message = txtMessage.Text;

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

lstMessages.Items.Add("Sent: " + message);

}

}

Consumer Code (Background Task):

using Confluent.Kafka;

private void StartConsumer()

{

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

GroupId = "chat-group",

AutoOffsetReset = AutoOffsetReset.Earliest

};

Task.Run(() =>

{

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

{

consumer.Subscribe("chat-topic");

while (true)

{

var cr = consumer.Consume();

Invoke((MethodInvoker)(() => lstMessages.Items.Add("Received: " + cr.Message.Value)));

}

}

});

}

