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

**Producer.cs**

**using Confluent.Kafka;**

**using System;**

**using System.Threading.Tasks;**

**public class Producer**

**{**

**private readonly IProducer<Null, string> \_producer;**

**public Producer(string bootstrapServers)**

**{**

**var config = new ProducerConfig { BootstrapServers = bootstrapServers };**

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

**}**

**public async Task SendMessageAsync(string topic, string message)**

**{**

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

**}**

**}**

**Consumer.cs**

**using Confluent.Kafka;**

**using System;**

**using System.Threading;**

**using System.Threading.Tasks;**

**public class Consumer**

**{**

**private readonly string \_topic;**

**private readonly IConsumer<Ignore, string> \_consumer;**

**public Consumer(string bootstrapServers, string topic)**

**{**

**\_topic = topic;**

**var config = new ConsumerConfig**

**{**

**BootstrapServers = bootstrapServers,**

**GroupId = "chat-consumer-group",**

**AutoOffsetReset = AutoOffsetReset.Earliest**

**};**

**\_consumer = new ConsumerBuilder<Ignore, string>(config).Build();**

**}**

**public void StartConsuming(CancellationToken token)**

**{**

**\_consumer.Subscribe(\_topic);**

**Task.Run(() =>**

**{**

**try**

**{**

**while (!token.IsCancellationRequested)**

**{**

**var cr = \_consumer.Consume(token);**

**Console.WriteLine($"\n👤 Friend: {cr.Value}");**

**}**

**}**

**catch (OperationCanceledException)**

**{**

**\_consumer.Close();**

**}**

**}, token);**

**}**

**}**

**Program.cs**

**using System;**

**using System.Threading;**

**class Program**

**{**

**static async Task Main(string[] args)**

**{**

**string bootstrapServers = "localhost:9092";**

**string topic = "chat-topic";**

**var producer = new Producer(bootstrapServers);**

**var consumer = new Consumer(bootstrapServers, topic);**

**using var cts = new CancellationTokenSource();**

**consumer.StartConsuming(cts.Token);**

**Console.WriteLine("💬 Chat started. Type messages below:");**

**Console.WriteLine("Type 'exit' to quit.\n");**

**while (true)**

**{**

**Console.Write("You: ");**

**var input = Console.ReadLine();**

**if (input?.ToLower() == "exit")**

**break;**

**await producer.SendMessageAsync(topic, input);**

**}**

**cts.Cancel();**

**}**

**}**

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

**using Confluent.Kafka;**

**using System;**

**using System.Threading;**

**using System.Threading.Tasks;**

**using System.Windows.Forms;**

**namespace KafkaWinChatApp**

**{**

**public partial class Form1 : Form**

**{**

**private IProducer<Null, string> \_producer;**

**private IConsumer<Ignore, string> \_consumer;**

**private CancellationTokenSource \_cts;**

**private readonly string topic = "winchat-topic";**

**public Form1()**

**{**

**InitializeComponent();**

**InitKafka();**

**StartConsumer();**

**}**

**private void InitKafka()**

**{**

**var producerConfig = new ProducerConfig**

**{**

**BootstrapServers = "localhost:9092"**

**};**

**var consumerConfig = new ConsumerConfig**

**{**

**BootstrapServers = "localhost:9092",**

**GroupId = $"chat-group-{Guid.NewGuid()}",**

**AutoOffsetReset = AutoOffsetReset.Earliest**

**};**

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

**\_consumer = new ConsumerBuilder<Ignore, string>(consumerConfig).Build();**

**\_consumer.Subscribe(topic);**

**}**

**private void StartConsumer()**

**{**

**\_cts = new CancellationTokenSource();**

**Task.Run(() =>**

**{**

**try**

**{**

**while (!\_cts.Token.IsCancellationRequested)**

**{**

**var cr = \_consumer.Consume(\_cts.Token);**

**AppendChat($"👤 Friend: {cr.Value}");**

**}**

**}**

**catch (OperationCanceledException)**

**{**

**\_consumer.Close();**

**}**

**}, \_cts.Token);**

**}**

**private void AppendChat(string message)**

**{**

**if (txtChatLog.InvokeRequired)**

**{**

**txtChatLog.Invoke(new Action(() => txtChatLog.AppendText($"{message}{Environment.NewLine}")));**

**}**

**else**

**{**

**txtChatLog.AppendText($"{message}{Environment.NewLine}");**

**}**

**}**

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

**{**

**var message = txtMessage.Text.Trim();**

**if (!string.IsNullOrEmpty(message))**

**{**

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

**AppendChat($"🧑 You: {message}");**

**txtMessage.Clear();**

**}**

**}**

**protected override void OnFormClosing(FormClosingEventArgs e)**

**{**

**\_cts?.Cancel();**

**\_producer?.Dispose();**

**\_consumer?.Dispose();**

**base.OnFormClosing(e);**

**}**

**}**

**}**