1. WebApi\_Handson

### 1. Introduction to Kafka

* **Kafka** is a distributed event streaming platform.
* It lets you **send**, **store**, **receive**, and **process** messages (like chat messages).
* It’s used by big companies for real-time data pipelines (like user activity, transactions, etc.).

### 2. Kafka Architecture (Like building blocks)

Main components:

* **Producer**: Sends (produces) data/messages.
* **Consumer**: Receives (consumes) data/messages.
* **Broker**: Kafka server that stores and delivers messages.
* **Topic**: Category or name of a stream (like "chatroom1").
* **Zookeeper**: Helps manage and coordinate Kafka servers.

### 3. Topics

* A **topic** is like a channel or group name in chat.
* Producers send messages to a topic.
* Consumers read messages from a topic.

### 4. Partitions

* Each topic is split into **partitions**.
* Partitions allow Kafka to **scale** (spread load).
* Messages are stored in order within a partition.

### 5. Brokers

* A **broker** is a Kafka server that stores data and handles requests.
* You can have multiple brokers in a Kafka **cluster**.

### 6. Kafka Plugin in .NET

* **Confluent.Kafka** is the most used .NET client for Kafka.

You install it using NuGet:

Install-Package Confluent.Kafka

Hands On:

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

Step 1: Open Command Prompt

cd C:\kafka

Step 2: Start Zookeeper

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

Step 3: Open a Second Command Prompt

cd C:\kafka

Step 4: Start Kafka Broker

.\bin\windows\kafka-server-start.bat .\config\server.properties

Step 5: Open a Third Command Prompt

This one is for commands like creating topics.

Navigate:

cd C:\kafka

Step 6: Create a Kafka Topic

We’ll call it chatroom:

.\bin\windows\kafka-topics.bat --create --topic chatroom --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

We should see:

Created topic chatroom.

Step 7: Start Kafka Producer (Send Message)

.\bin\windows\kafka-console-producer.bat --topic chatroom --bootstrap-server localhost:9092

Now type messages → hit Enter to send.

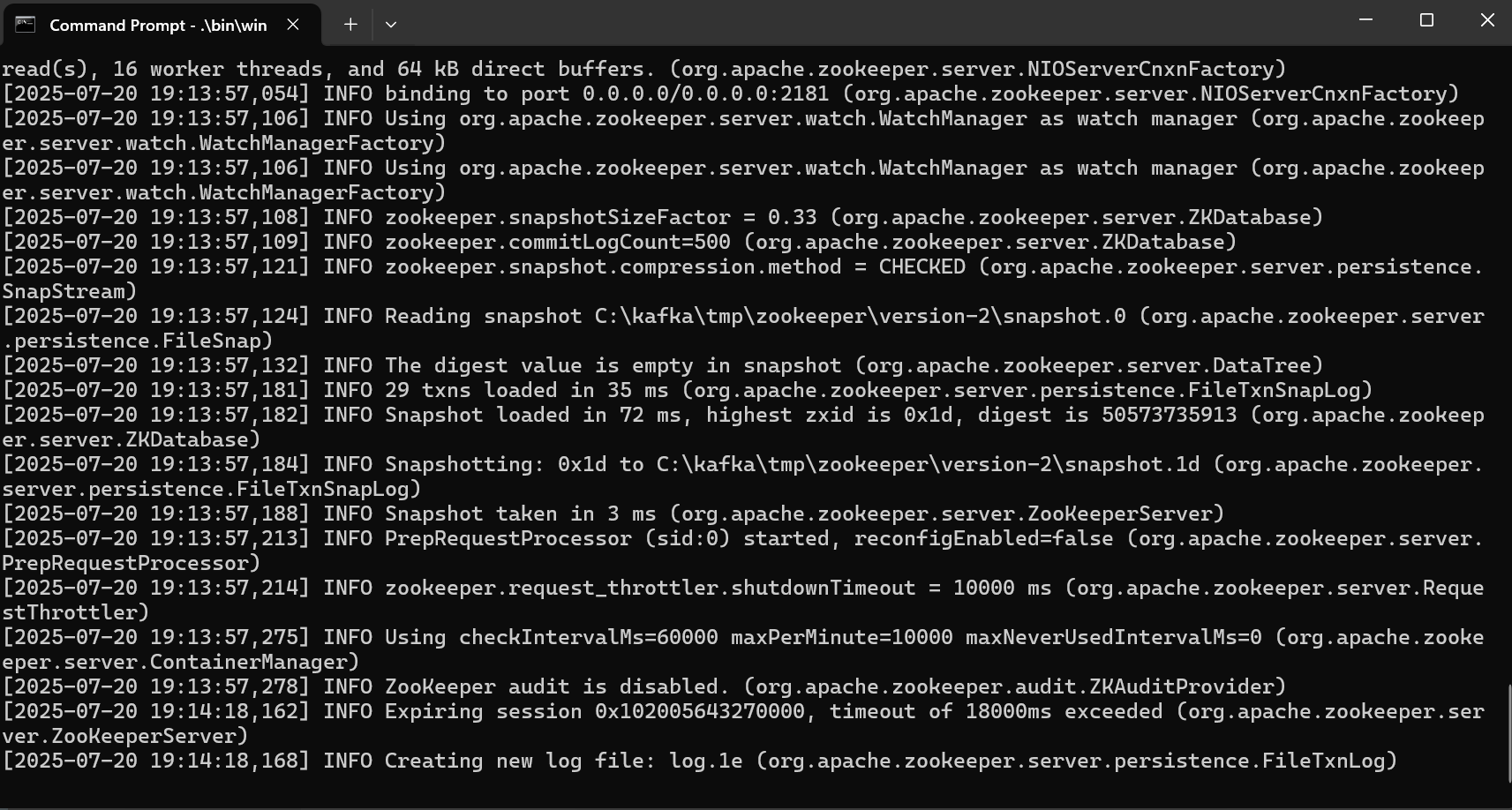
Step 8: Start Kafka Consumer (Receive Message)

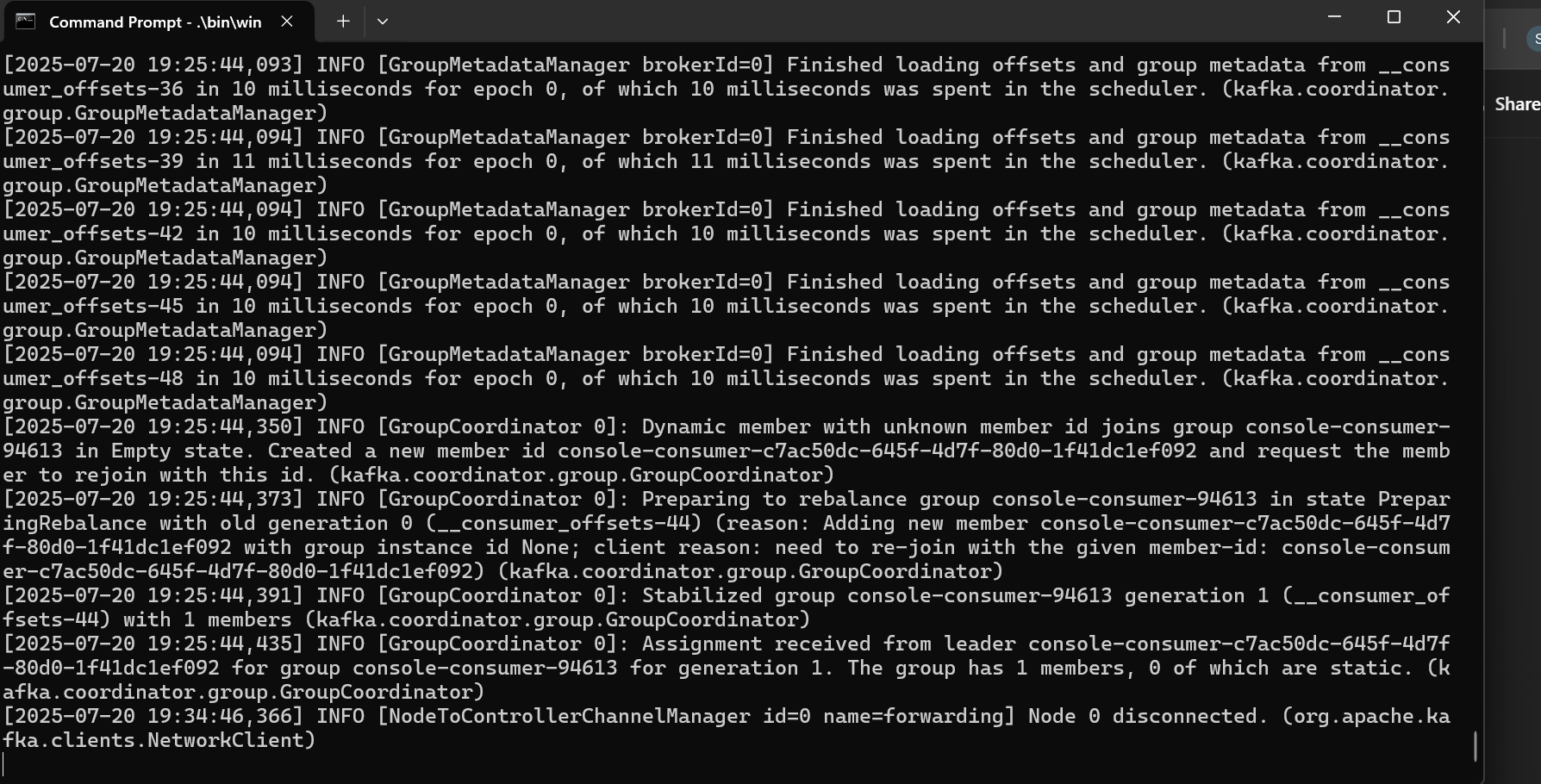
Open a new terminal. Navigate to Kafka:

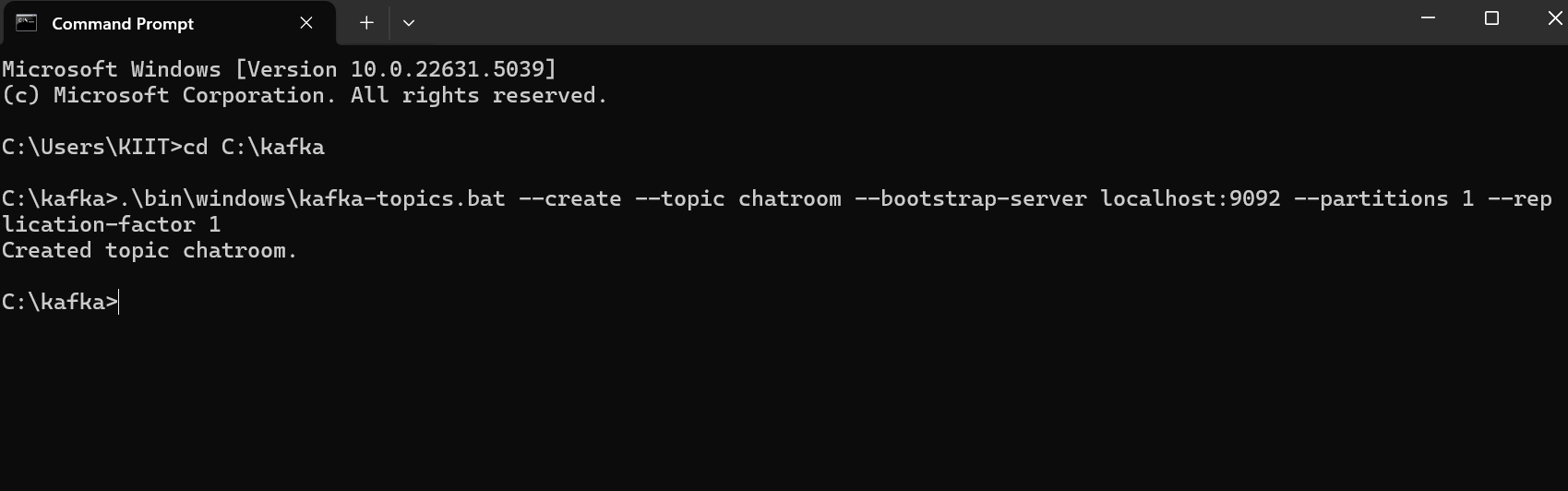
Then:

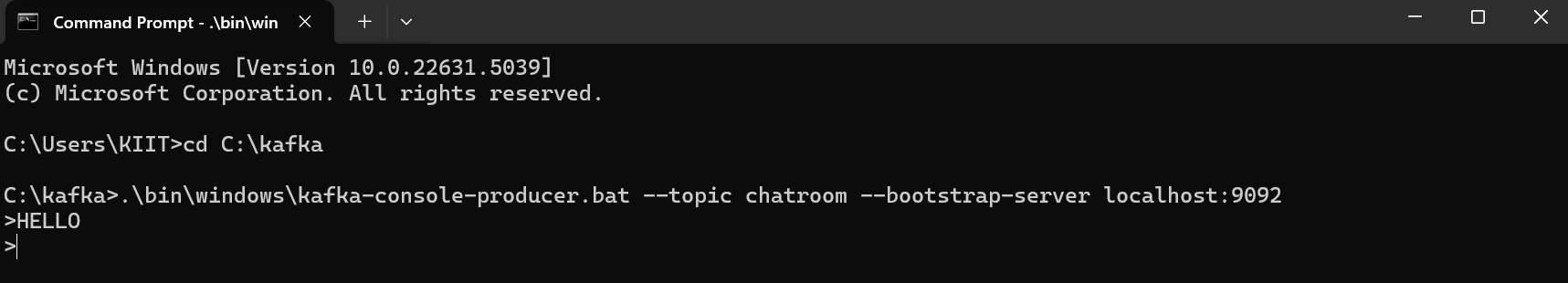
.\bin\windows\kafka-console-consumer.bat --topic chatroom --from-beginning --bootstrap-server localhost:9092

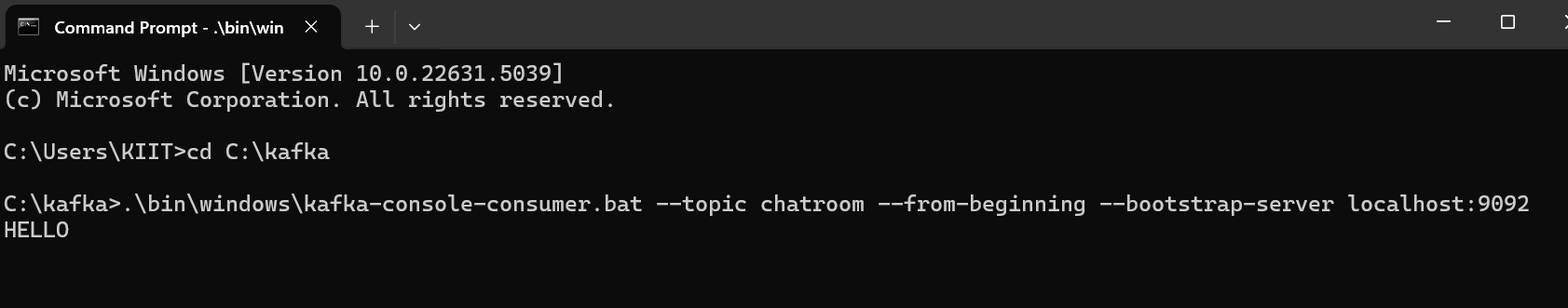
We’ll now see the messages sent from the producer appear here.











We have now completed the **Kafka CLI Chat Demo**

Part B: C# Chat Application Using Kafka (.NET)

Step 1: Create a new C# Console App

dotnet new console -n KafkaConsoleChat

cd KafkaConsoleChat

Step 2: Add Kafka Library (Confluent.Kafka)

dotnet add package Confluent.Kafka

Step 3: Create **Producer.cs**

**using Confluent.Kafka;**

**using System;**

**using System.Threading.Tasks;**

**class Producer**

**{**

**public async Task SendMessages()**

**{**

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

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

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

**string message;**

**while ((message = Console.ReadLine()) != "exit")**

**{**

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

**}**

**}**

**}**

Step 4: Create **Consumer.cs**

**using Confluent.Kafka;**

**using System;**

**class Consumer**

**{**

**public void ReceiveMessages()**

**{**

**var config = new ConsumerConfig**

**{**

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

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

**AutoOffsetReset = AutoOffsetReset.Earliest**

**};**

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

**consumer.Subscribe("chatroom");**

**Console.WriteLine("Listening for messages...");**

**while (true)**

**{**

**var msg = consumer.Consume();**

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

**}**

**}**

**}**

Step 5: Modify **Program.cs**

**using System;**

**using System.Threading.Tasks;**

**class Program**

**{**

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

**{**

**Console.WriteLine("Kafka Chat App");**

**Console.WriteLine("1. Send Messages");**

**Console.WriteLine("2. Receive Messages");**

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

**if (choice == "1")**

**{**

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

**await producer.SendMessages();**

**}**

**else if (choice == "2")**

**{**

**var consumer = new Consumer();**

**consumer.ReceiveMessages();**

**}**

**}**

**}**

Step 6: Run the App

Open two terminals:

#### Terminal A – Send Messages

dotnet run

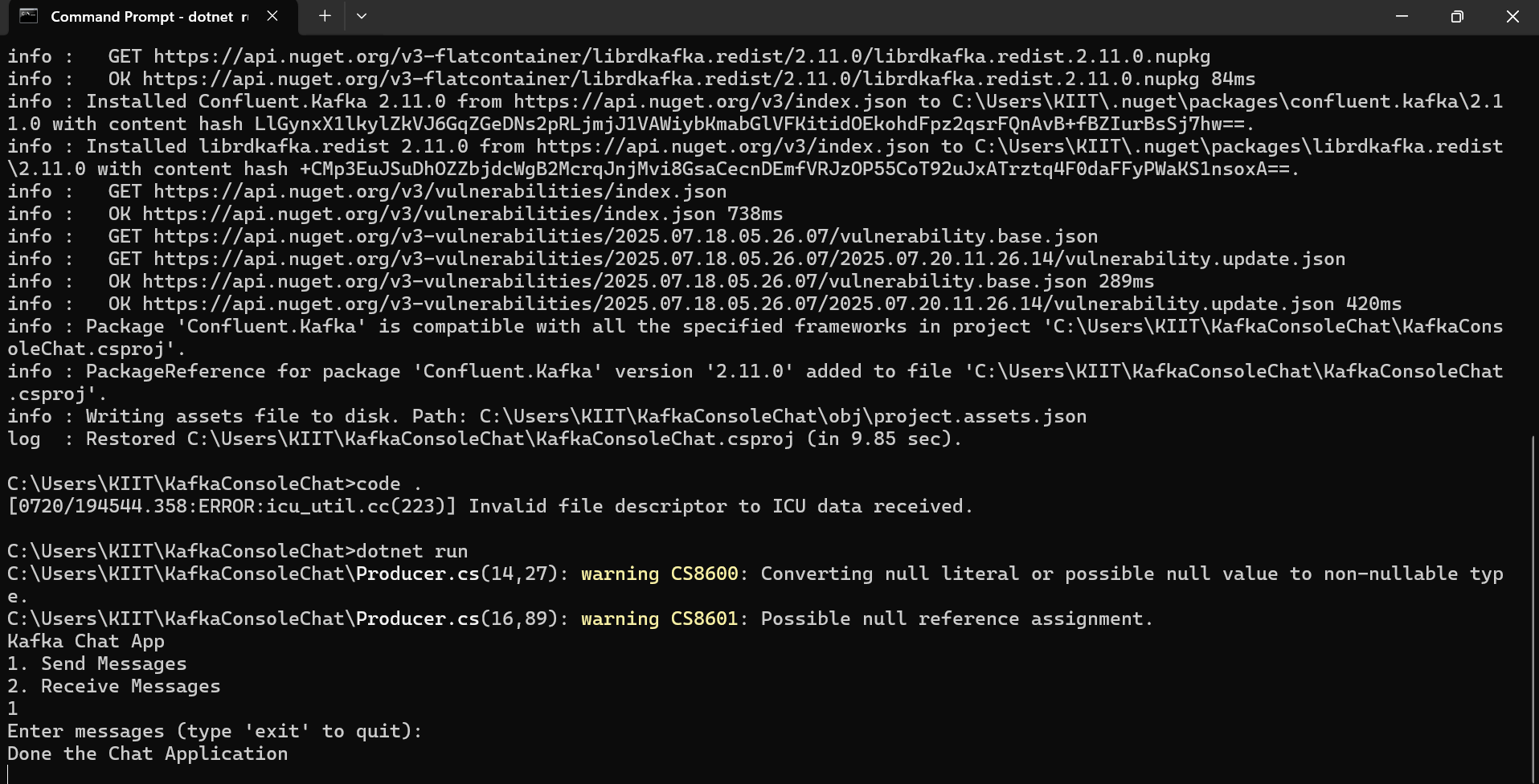
Choose option 1: Send Messages

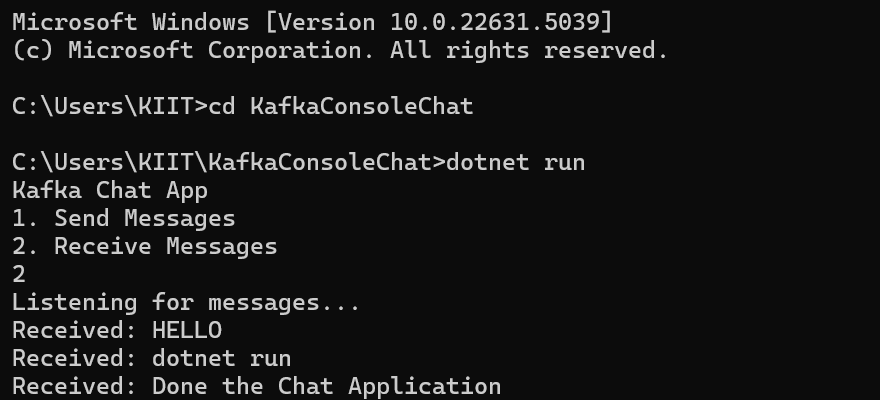
Terminal B – Receive Messages

dotnet run

Choose option 2: Receive Messages

Now we have created a **Kafka Chat App using .NET Console**

****

****

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

Step 1: Create a Windows Forms App

Step 2: Install Kafka NuGet Package

1. **Design the UI Layout**

In Form1.cs [Design]:

Drag the following controls onto the form:

TextBox → name: txtMessage (for user to type message)

Button → name: btnSend, Text: Send

ListBox → name: lstChat (for showing received messages)

Set Multiline = true for txtMessage and Dock property for lstMessages to Bottom.

FULL CODE:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Threading;

using Confluent.Kafka;

namespace KafkaWinFormsChats

{

public partial class Form1 : Form

{

private ContextMenuStrip contextMenuStrip1;

private IContainer components;

private ToolStripTextBox toolStripTextBox1;

private TextBox txtMessage;

private ListBox lstChat;

private Button btnSend;

private ToolStripMenuItem txtMessageToolStripMenuItem;

private IProducer<Null, string> producer;

private CancellationTokenSource cancellationTokenSource;

public Form1()

{

InitializeComponent();

}

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.contextMenuStrip1 = new System.Windows.Forms.ContextMenuStrip(this.components);

this.toolStripTextBox1 = new System.Windows.Forms.ToolStripTextBox();

this.txtMessageToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.txtMessage = new System.Windows.Forms.TextBox();

this.lstChat = new System.Windows.Forms.ListBox();

this.btnSend = new System.Windows.Forms.Button();

this.contextMenuStrip1.SuspendLayout();

this.SuspendLayout();

this.contextMenuStrip1.ImageScalingSize = new System.Drawing.Size(24, 24);

this.contextMenuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.toolStripTextBox1,

this.txtMessageToolStripMenuItem});

this.contextMenuStrip1.Name = "contextMenuStrip1";

this.contextMenuStrip1.Size = new System.Drawing.Size(175, 71);

this.toolStripTextBox1.Font = new System.Drawing.Font("Segoe UI", 9F);

this.toolStripTextBox1.Name = "toolStripTextBox1";

this.toolStripTextBox1.Size = new System.Drawing.Size(100, 31);

this.txtMessageToolStripMenuItem.Name = "txtMessageToolStripMenuItem";

this.txtMessageToolStripMenuItem.Size = new System.Drawing.Size(174, 32);

this.txtMessageToolStripMenuItem.Text = "txtMessage";

this.txtMessage.Location = new System.Drawing.Point(12, 184);

this.txtMessage.Multiline = true;

this.txtMessage.Name = "txtMessage";

this.txtMessage.Size = new System.Drawing.Size(222, 26);

this.txtMessage.TabIndex = 2;

this.lstChat.FormattingEnabled = true;

this.lstChat.ItemHeight = 20;

this.lstChat.Location = new System.Drawing.Point(12, 12);

this.lstChat.Name = "lstChat";

this.lstChat.Size = new System.Drawing.Size(174, 84);

this.lstChat.TabIndex = 3;

this.btnSend.Location = new System.Drawing.Point(365, 187);

this.btnSend.Name = "btnSend";

this.btnSend.Size = new System.Drawing.Size(75, 23);

this.btnSend.TabIndex = 4;

this.btnSend.Text = "Send";

this.btnSend.UseVisualStyleBackColor = true;

this.btnSend.Click += new System.EventHandler(this.btnSend\_Click);

this.ClientSize = new System.Drawing.Size(477, 244);

this.Controls.Add(this.btnSend);

this.Controls.Add(this.lstChat);

this.Controls.Add(this.txtMessage);

this.Name = "Form1";

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1\_FormClosing);

this.Load += new System.EventHandler(this.Form1\_Load);

this.contextMenuStrip1.ResumeLayout(false);

this.contextMenuStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

private void Form1\_Load(object sender, EventArgs e)

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

cancellationTokenSource = new CancellationTokenSource();

Task.Run(() => StartConsumer(cancellationTokenSource.Token));

}

private void StartConsumer(CancellationToken token)

{

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

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

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("chatroom");

try

{

while (!token.IsCancellationRequested)

{

var cr = consumer.Consume(token);

AddMessageToListBox($"Friend: {cr.Message.Value}");

}

}

catch (OperationCanceledException) { }

finally

{

consumer.Close();

}

}

private void AddMessageToListBox(string message)

{

if (lstChat.InvokeRequired)

{

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

}

else

{

lstChat.Items.Add(message);

}

}

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

{

string message = txtMessage.Text.Trim();

if (!string.IsNullOrEmpty(message))

{

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

txtMessage.Clear();

}

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

cancellationTokenSource?.Cancel();

producer?.Flush(TimeSpan.FromSeconds(1));

producer?.Dispose();

}

}

}

Run Two Instances

