**WEEK 5  
  
ASP.NET Core 8.0 Web API  
  
CODE->  
  
Install Kafka & Zookeeper-->**   
  
# Start Zookeeper

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

# Start Kafka Server

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

## Create Kafka Topic--> # Create topic named "chat-topic"

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

## # List topics

## .\bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092 **Publisher Code-->**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class Producer

{

public static async Task Main(string[] args)

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

{

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

string input;

do

{

input = Console.ReadLine();

if (input != "exit")

{

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

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

}

} while (input != "exit");

}

}

}

**Subscriber Code-->**using Confluent.Kafka;

using System;

using System.Threading;

class Consumer

{

public static void Main(string[] args)

{

var config = new ConsumerConfig

{

GroupId = "chat-consumer-group",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

{

consumer.Subscribe("chat-topic");

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

try

{

while (true)

{

var cr = consumer.Consume(CancellationToken.None);

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

}

}

catch (OperationCanceledException)

{

consumer.Close();

}

}

}

}

**Form Code (Producer + Consumer in UI)-->**using System;

using System.Windows.Forms;

using Confluent.Kafka;

using System.Threading.Tasks;

namespace KafkaChatApp

{

public partial class ChatForm : Form

{

private readonly ProducerConfig \_producerConfig = new ProducerConfig { BootstrapServers = "localhost:9092" };

private readonly ConsumerConfig \_consumerConfig = new ConsumerConfig

{

GroupId = "chat-ui-group",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

public ChatForm()

{

InitializeComponent();

StartConsumer();

}

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

{

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

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

txtChat.AppendText($"You: {txtMessage.Text}\n");

txtMessage.Clear();

}

private void StartConsumer()

{

Task.Run(() =>

{

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

consumer.Subscribe("chat-topic");

try

{

while (true)

{

var cr = consumer.Consume();

Invoke(new Action(() =>

{

txtChat.AppendText($"Friend: {cr.Message.Value}\n");

}));

}

}

catch

{

consumer.Close();

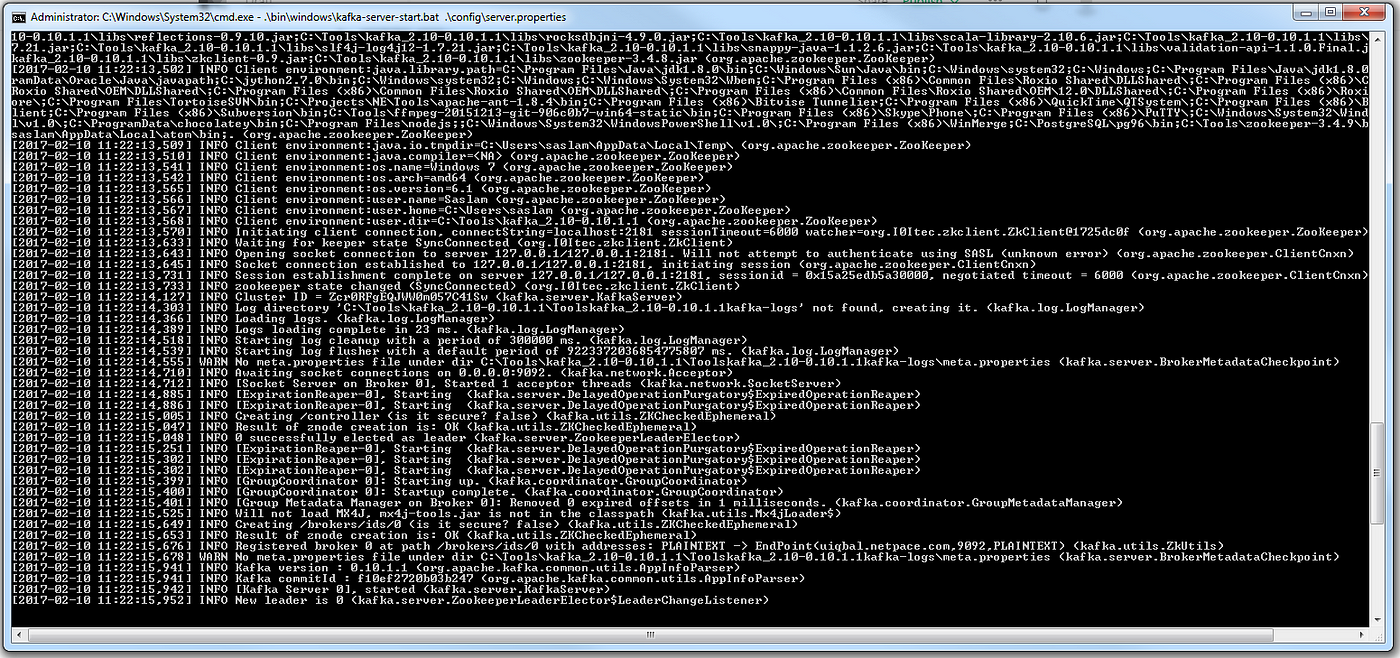
}

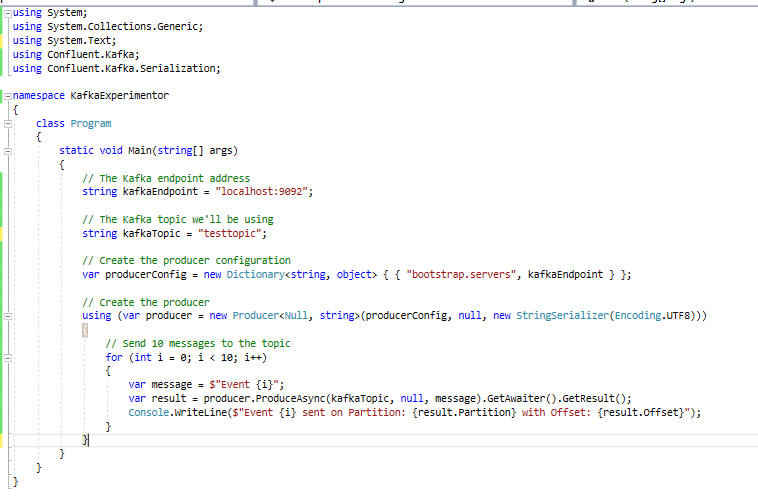
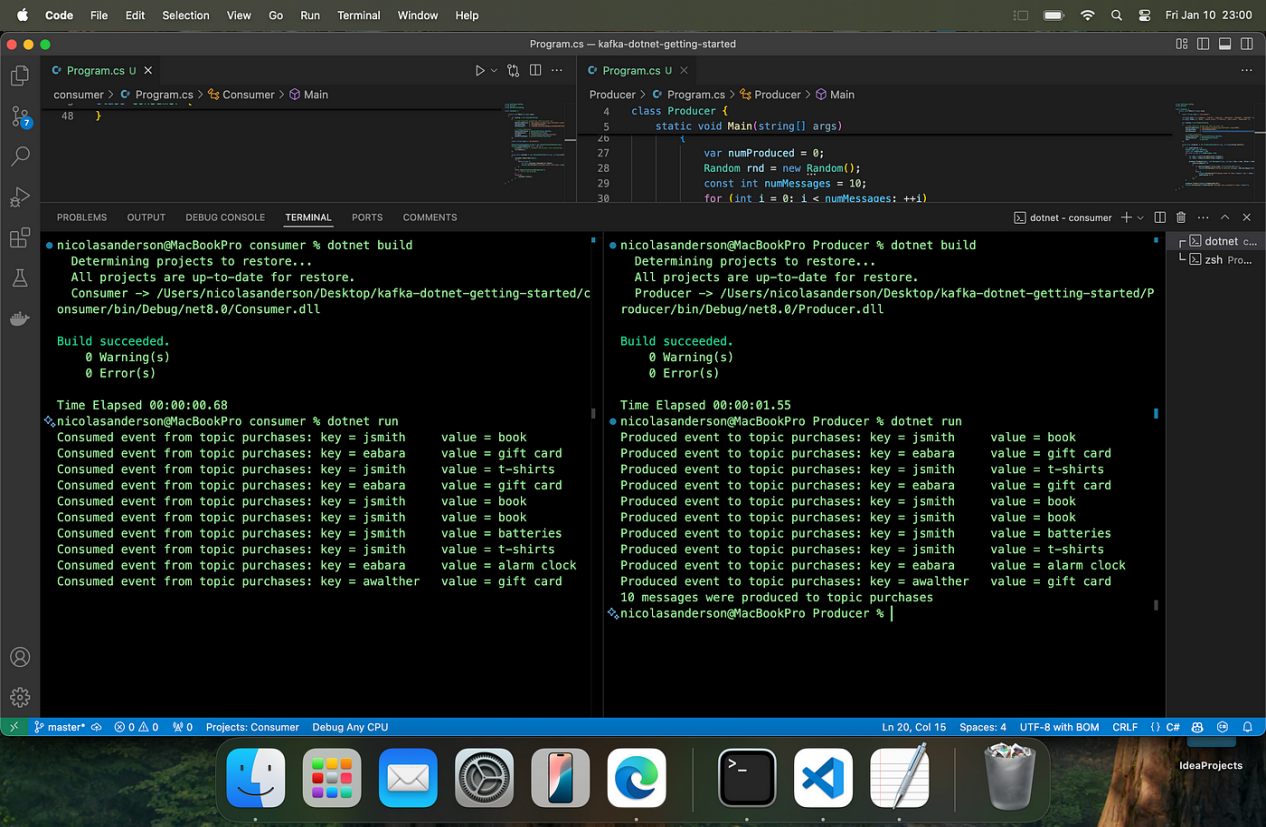
});

}

}

}

**OUTPUT-->(Screenshots)  
  
**

****