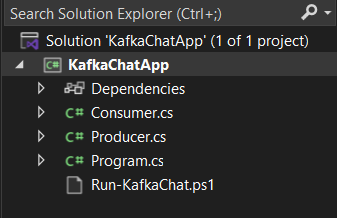
**WebAPI\_HandsOn-6**

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

**File structure:**



**Consumer.cs**

using Confluent.Kafka;

using System;

using System.Threading;

using System.Threading.Tasks;

class Consumer

{

public static void StartConsumer()

{

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("Listening for messages...");

while (true)

{

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

Console.WriteLine($"[Received] {result.Message.Value}");

}

}

}

**Producer.cs**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class Producer

{

public static async Task ProduceMessage(string message)

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

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

Console.WriteLine($"Message sent to {result.TopicPartitionOffset}");

}

}

**Program.cs**

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

Console.WriteLine("Kafka Chat Application");

Console.WriteLine("Type 'receive' to listen, or type message to send:");

while (true)

{

var input = Console.ReadLine();

if (input.ToLower() == "receive")

{

Consumer.StartConsumer();

}

else

{

await Producer.ProduceMessage(input);

}

}

}

}

