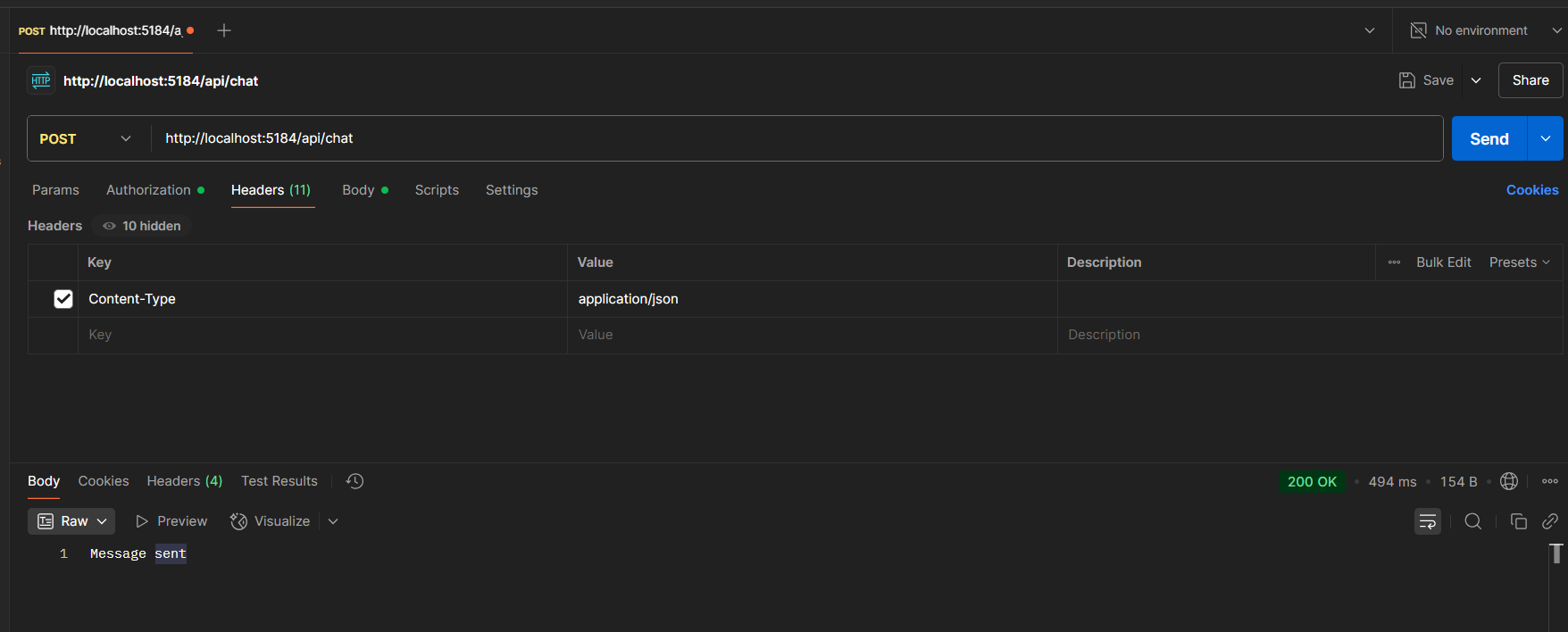
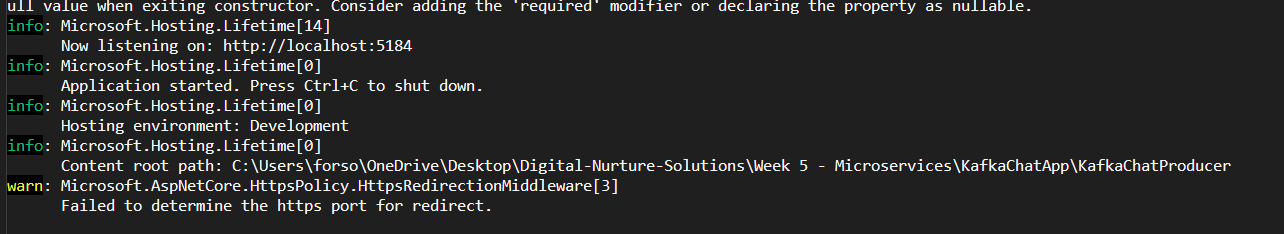
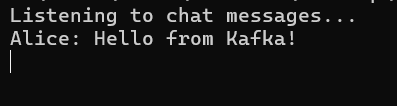
WebApi\_HandsOn

1. Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prompt.
2. Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.

Output:







Code:

ChatController.cs

using Confluent.Kafka;

using KafkaChatProducer.Models;

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

namespace KafkaChatProducer.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class ChatController : ControllerBase

{

[HttpPost]

public IActionResult SendMessage([FromBody] ChatMessage message)

{

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

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

{

var jsonMessage = JsonSerializer.Serialize(message);

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

producer.Flush(TimeSpan.FromSeconds(5));

}

return Ok("Message sent");

}

}

}

ChatMessage.cs

namespace KafkaChatProducer.Models

{

public class ChatMessage

{

public string Sender { get; set; }

public string Message { get; set; }

}

}

Program.cs

namespace KafkaChatProducer

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}