**Week-5 (Integrating MyWebAPI with Apache Kafka ( Hands-On 6)**

*AuthController.cs :-*

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using MyFirstWebAPI.Models;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace MyFirstWebAPI.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_config;

public AuthController(IConfiguration config)

{

\_config = config;

}

[AllowAnonymous]

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (model.Username == "admin" && model.Password == "password")

{

var token = GenerateJWTToken(1, "Admin");

return Ok(new { token });

}

return Unauthorized("Invalid credentials");

}

private string GenerateJWTToken(int userId, string role)

{

var key = Encoding.UTF8.GetBytes("mysuperdupersecurekeythatishardtoguess123!");

var creds = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256);

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, role),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.UtcNow.AddMinutes(30),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

*EmployeeController.cs:-*

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyFirstWebAPI.Models;

using MyFirstWebAPI.Services;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace MyFirstWebAPI.Controllers

{

[ApiController]

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

[Authorize(Roles = "Admin")]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

var employees = new List<string> { "Alice", "Bob", "Charlie" };

return Ok(employees);

}

[HttpPost("send")]

public async Task<IActionResult> SendToKafka(

[FromServices] KafkaProducerService kafka,

[FromBody] KafkaMessage input)

{

await kafka.SendMessageAsync("chat-topic", input.Message);

return Ok("Message sent to Kafka");

}

}

}

**Models:-**

*Employee.cs :-*

using System;

using System.Collections.Generic;

namespace MyFirstWebAPI.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; } = new Department();

public List<Skill> Skills { get; set; } = new List<Skill>();

public DateTime DateOfBirth { get; set; }

}

}

*KafkaMessage.cs :-*

namespace MyFirstWebAPI.Models

{

public class KafkaMessage

{

public string Message { get; set; } = string.Empty;

}

}

*LoginModel.cs :-*

namespace MyFirstWebAPI.Models

{

public class LoginModel

{

public string Username { get; set; } = string.Empty;

public string Password { get; set; } = string.Empty;

}

}

**Services :-**

*KafkaProducerService.cs :-*

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

namespace MyFirstWebAPI.Services

{

public class KafkaProducerService

{

private readonly IProducer<Null, string> \_producer;

public KafkaProducerService()

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

}

public async Task SendMessageAsync(string topic, string message)

{

try

{

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

Console.WriteLine($"✅ Message delivered to {deliveryResult.TopicPartitionOffset}: {message}");

}

catch (ProduceException<Null, string> ex)

{

Console.WriteLine($" Kafka Produce Error: {ex.Error.Reason}");

}

}

}

}

***KafkaConsumerApp(Integrated Project) :-***

KafkaConsumerApp

*Program.cs :-*

using Confluent.Kafka;

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

GroupId = "test-consumer-group",

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("chat-topic");

Console.WriteLine("🎧 Listening to Kafka topic 'chat-topic'...");

try

{

while (true)

{

var cr = consumer.Consume();

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

}

}

catch (OperationCanceledException)

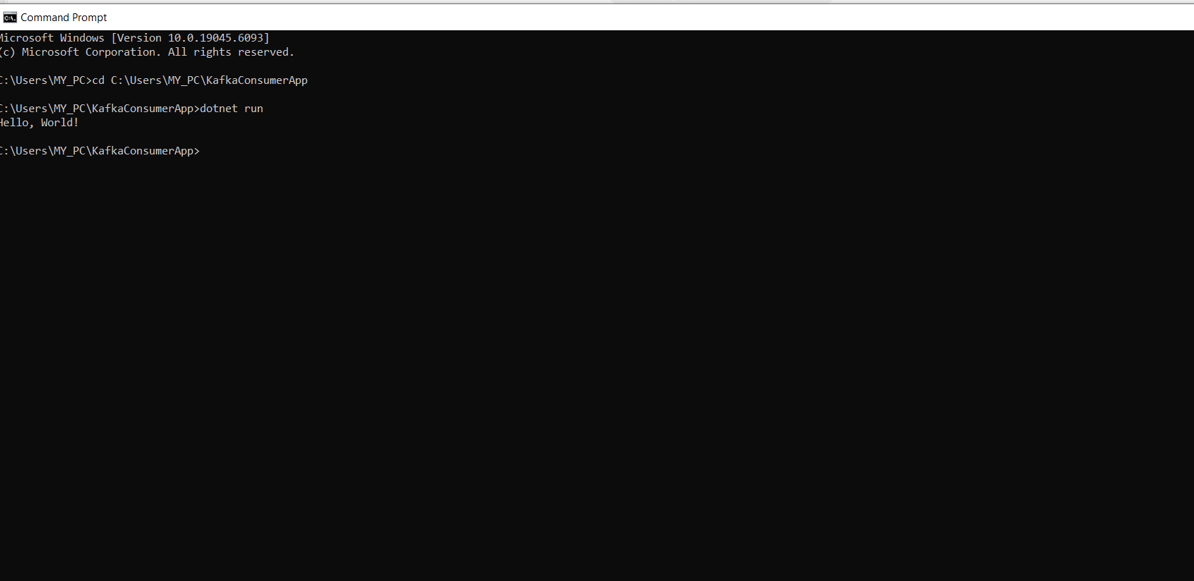
{

consumer.Close();

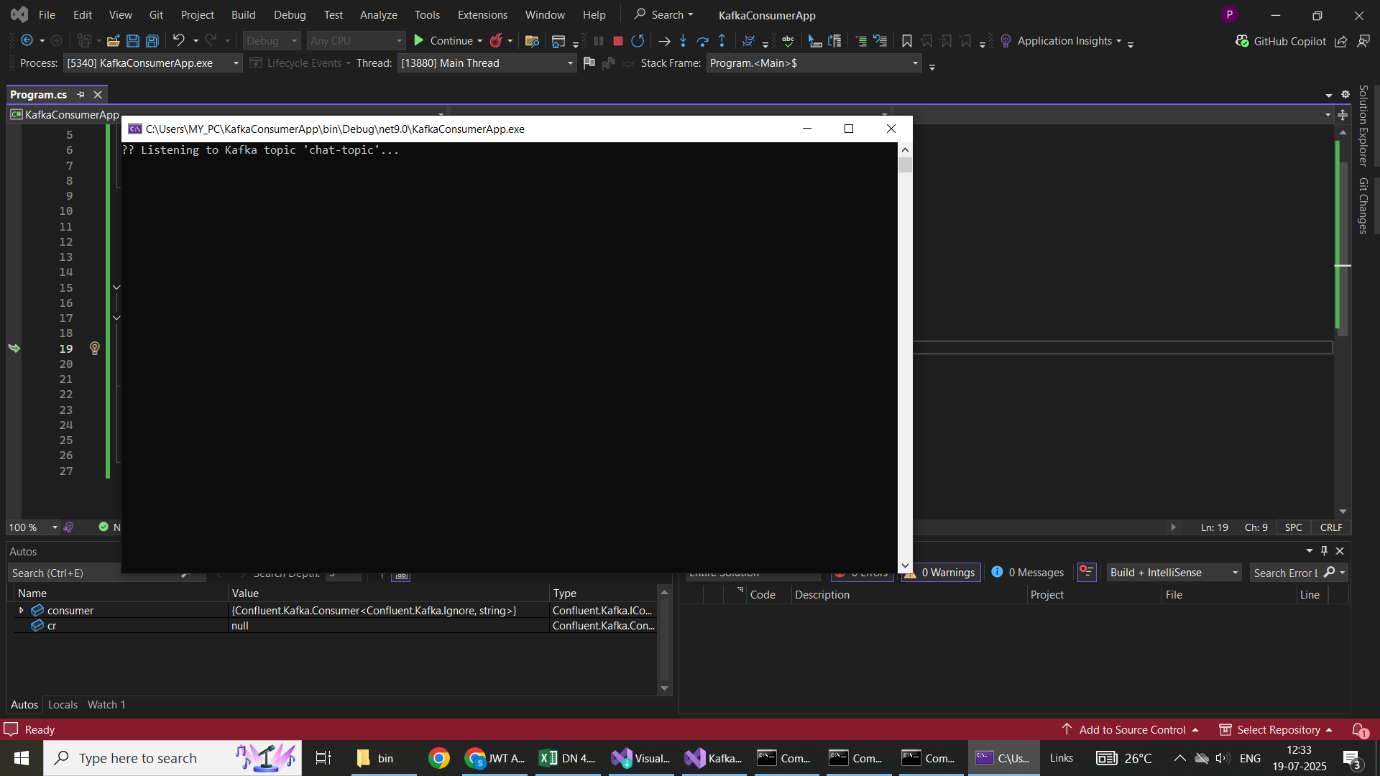
}

**OUTPUT :-**

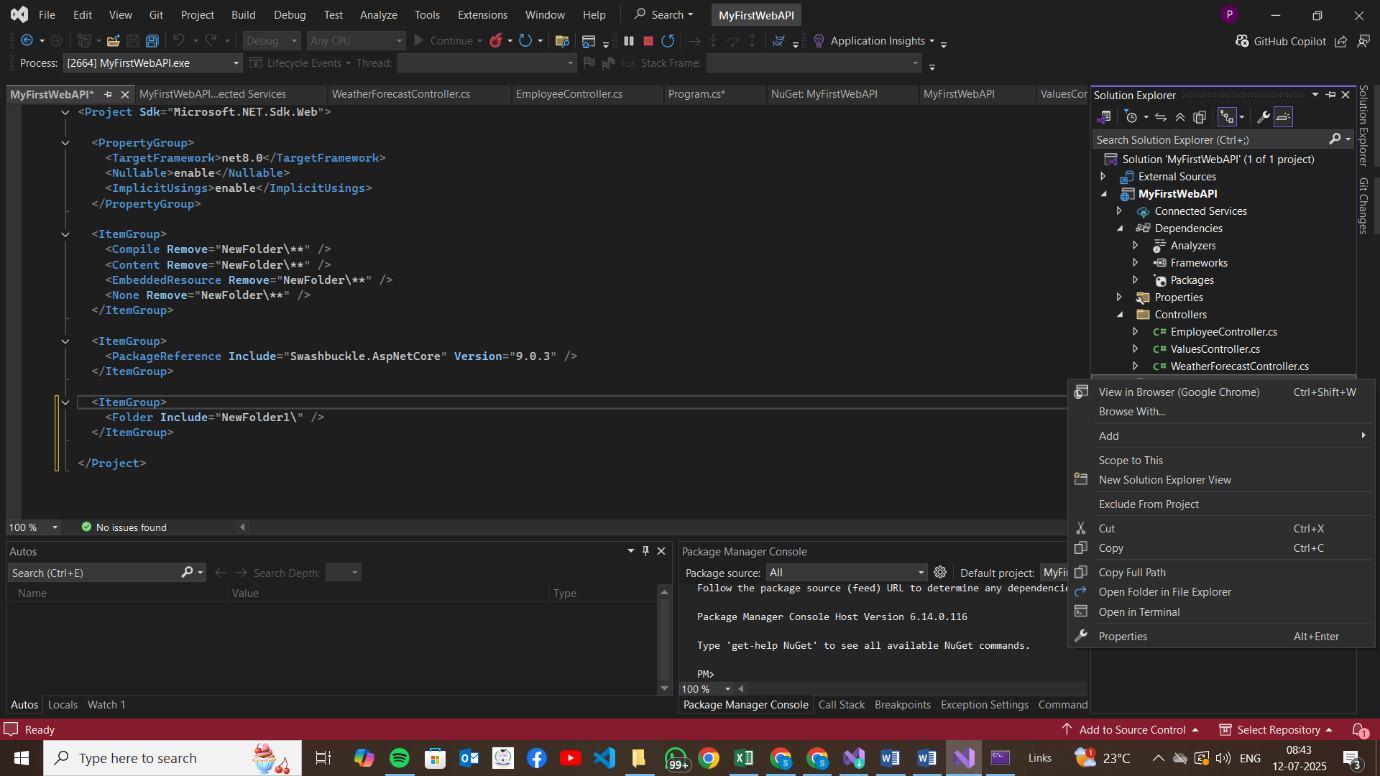
*Initial Kafka Integration:-*

****

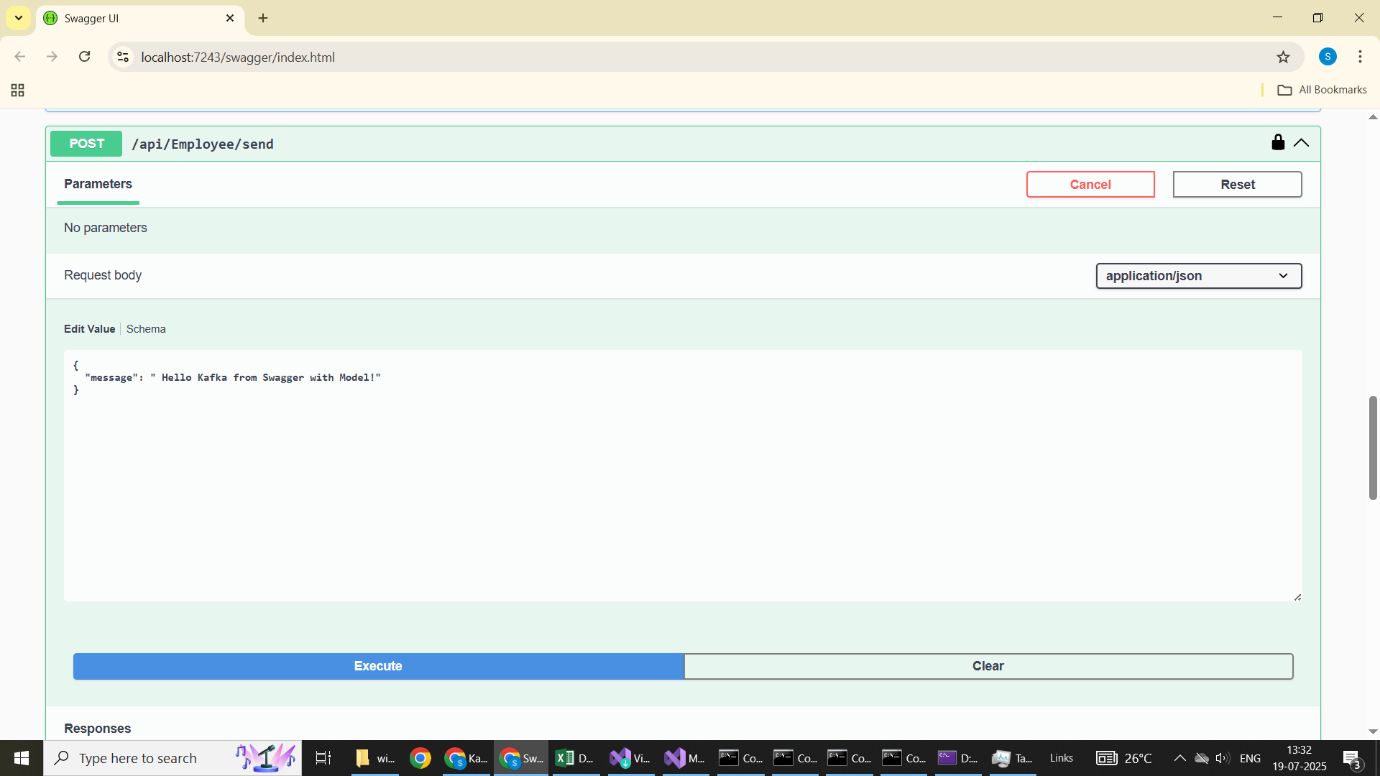
*MyWebAPI Output :-*

**

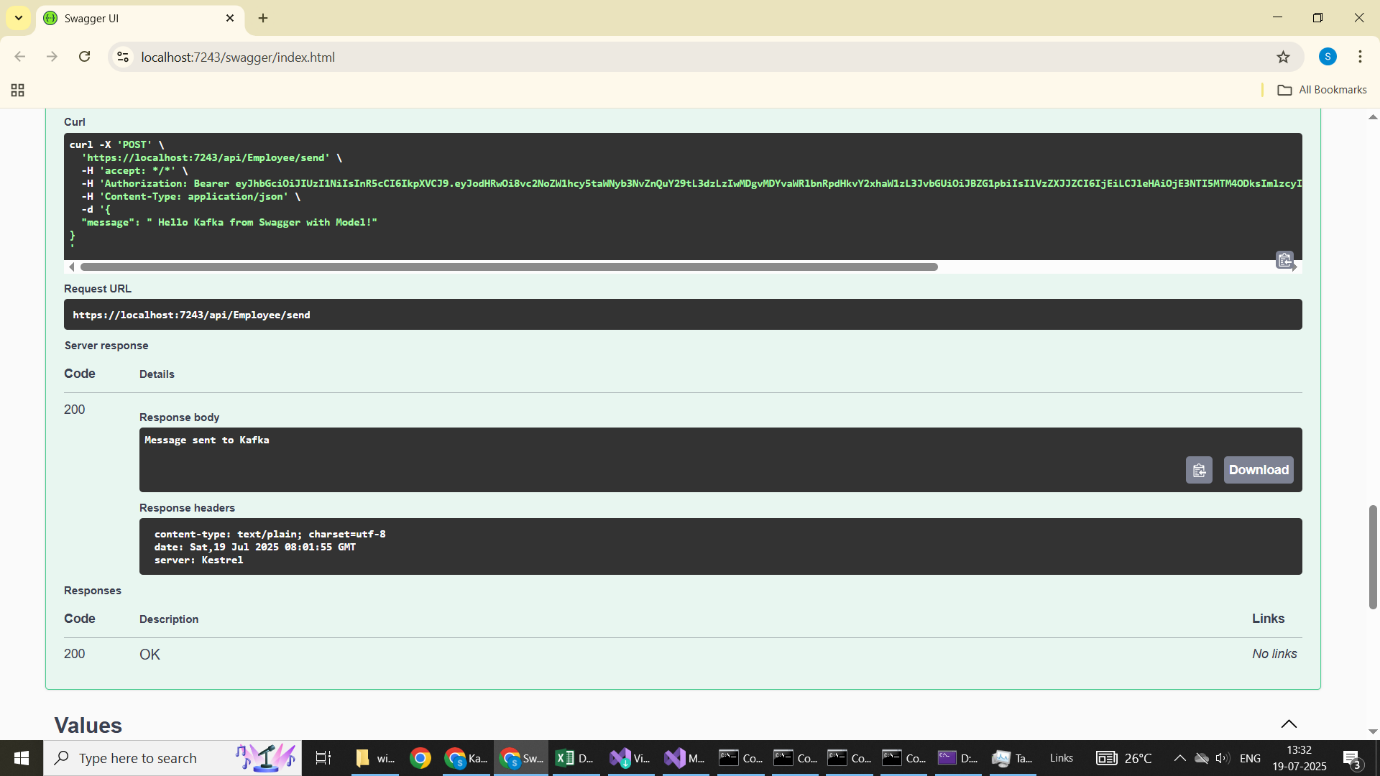
*MyWebAPI.Console:-*

**

*Swagger UI :-*

**

*Kafka Integration Output After Authorization:-*

**