**Week5-HANDSON**

**Module6-ASP.NET Core 8.0 Web API**

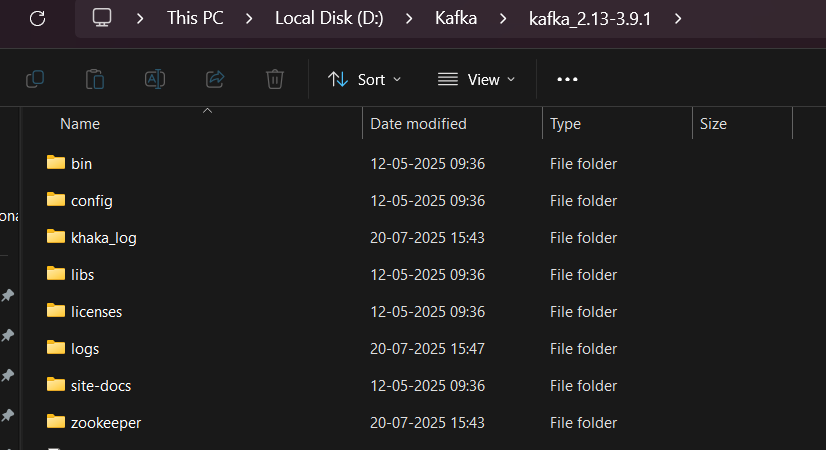
**Ex6:** **WebApi\_Handson**

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

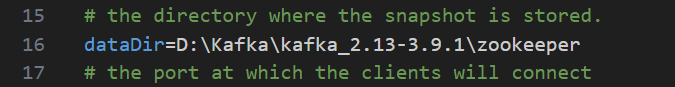
**Step1:** Install Khafka inside your system and extract

(D:\Kafka\kafka\_2.13-3.9.1)

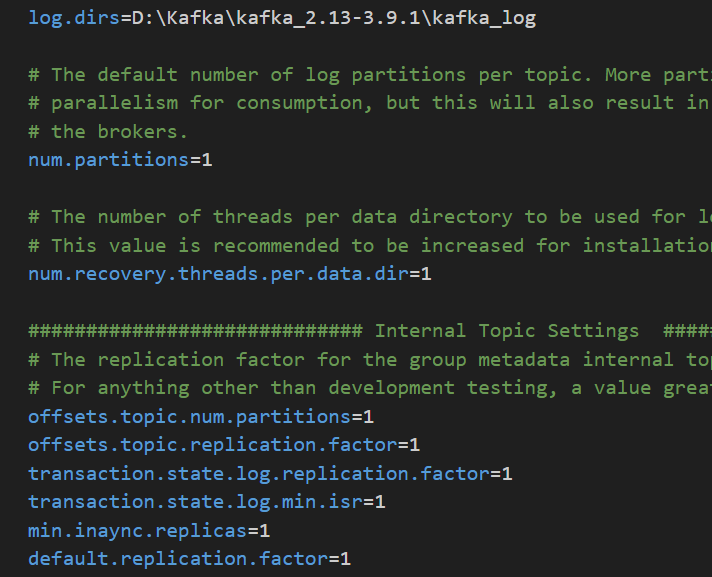
**Step2:**create “zookeeper”,”kafka\_log” folder inside it



**Step3**: Edit Khafka/config/zookeeper.properties file by below



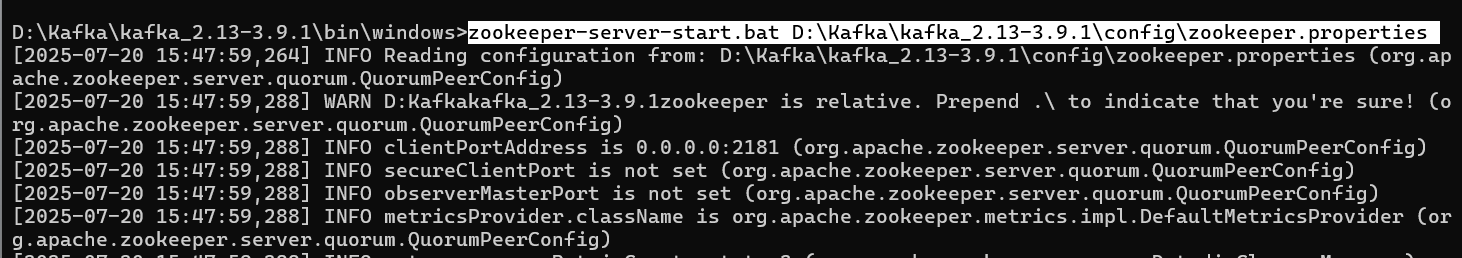
**Step4**: Edit Khafka/config/server.properties file by below



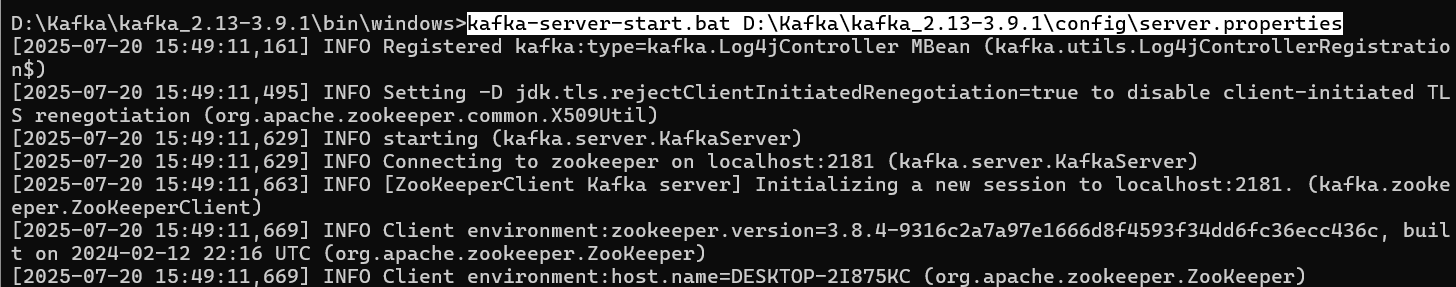
**Step5**: now add “bin” path in environment variable

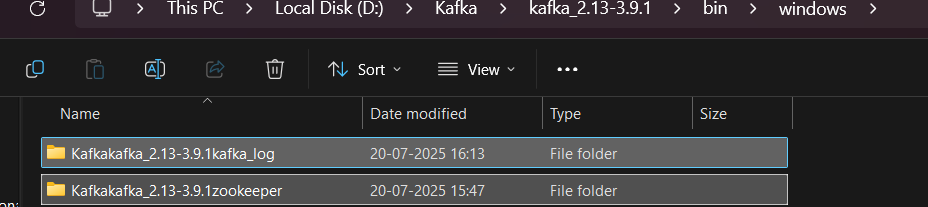
(D:\Kafka\kafka\_2.13-3.9.1\bin)

**Step6**: To start zookeeper cmd of D:\Kafka\kafka\_2.13-3.9.1\bin\windows and Run

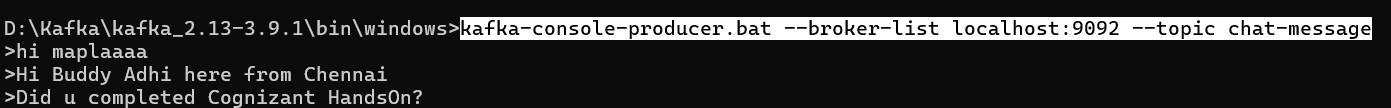


**Step7**: To start server open new cmd D:\Kafka\kafka\_2.13-3.9.1\bin\windows and run

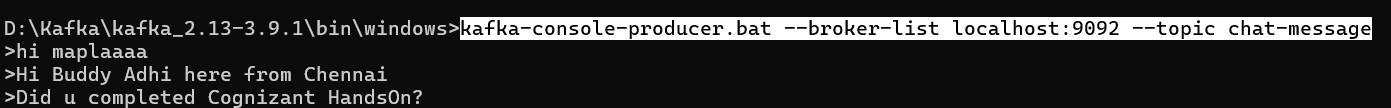
Note: you can see the 2 new folders created which indicates the zookeeper and kafka server comment worked



**Step8**: To create Producer(sender) open new cmd D:\Kafka\kafka\_2.13-3.9.1\bin\windows and run and “send” some message

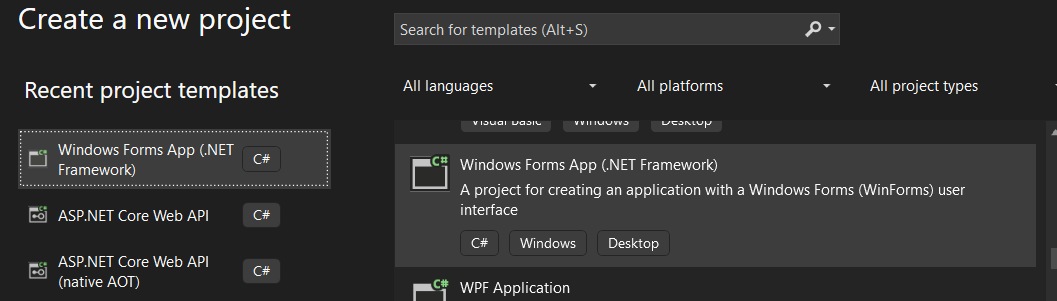


**Step9**:To create Consumer(reciver) open new cmd D:\Kafka\kafka\_2.133.9.1\bin\windows and run and you can see “received messages”

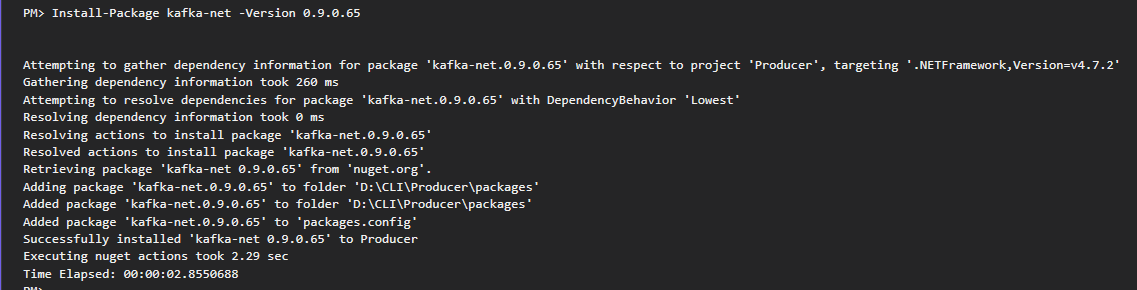


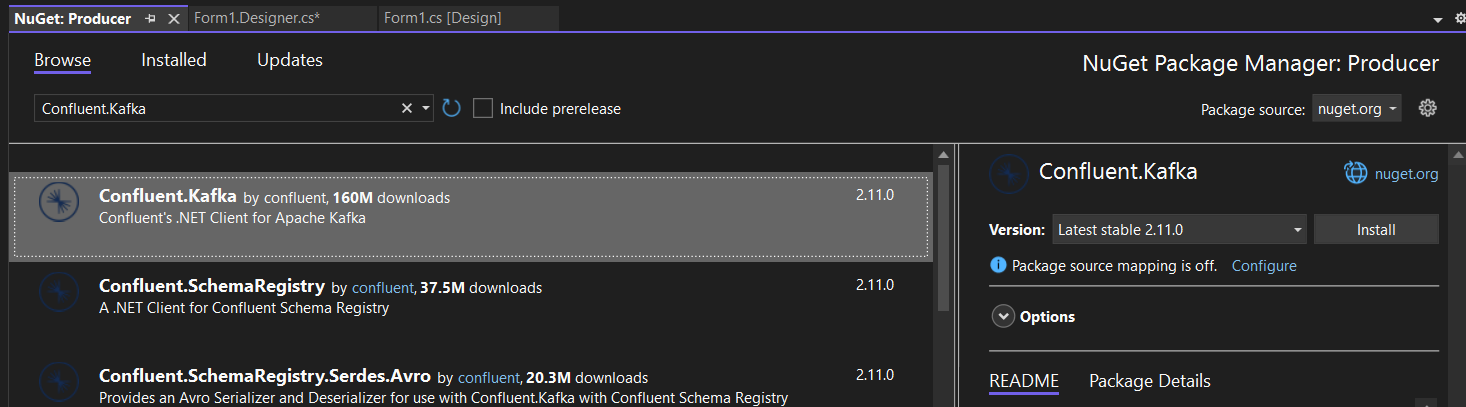
**2. Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.**

Step1:create Project “Producer” and “Consumer” in vs community with theme “Windows Forms App”



Step2:Install Kafka-net using package manager by below refered



Step3:Install Confluent.Kafka through NuGet Manager Browser****

Step4: Update “Form1.cs” code of “Producer” by below

using System;

using System.Windows.Forms;

using Confluent.Kafka;

namespace Producer

{

public partial class Form1 : Form

{

private ProducerConfig \_producerConfig;

private IProducer<Null, string> \_producer;

private const string KafkaBootstrapServers = "localhost:9092";

private const string KafkaTopic = "your-topic-name";

public Form1()

{

InitializeComponent();

InitializeKafkaProducer();

}

private void InitializeKafkaProducer()

{

\_producerConfig = new ProducerConfig

{

BootstrapServers = KafkaBootstrapServers

};

try

{

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

btnSend.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show($"Failed to initialize Kafka producer: {ex.Message}", "Producer Initialization Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

btnSend.Enabled = false;

}

}

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

{

string messageToSend = txtMessage.Text;

try

{

var deliveryReport = await \_producer.ProduceAsync(KafkaTopic, new Message<Null, string> { Value = messageToSend });

MessageBox.Show($"Message sent successfully to topic '{KafkaTopic}'!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

txtMessage.Clear();

}

catch (ProduceException<Null, string> ex)

{

MessageBox.Show($"Failed to send message: {ex.Error.Reason}", "Kafka Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

protected override void OnFormClosed(FormClosedEventArgs e)

{

\_producer?.Flush(TimeSpan.FromSeconds(10));

\_producer?.Dispose();

base.OnFormClosed(e);

}

}

}

Step5: Update “Form1.Designer.cs” code of “Producer” by below

namespace Producer

{

partial class Form1

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

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

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

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

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Text = "Kafka Sender";

this.Name = "Form1";

this.txtMessage.Location = new System.Drawing.Point(30, 30);

this.txtMessage.Name = "txtMessage";

this.txtMessage.Size = new System.Drawing.Size(400, 26);

this.txtMessage.TabIndex = 0;

this.btnSend.Location = new System.Drawing.Point(450, 30);

this.btnSend.Name = "btnSend";

this.btnSend.Size = new System.Drawing.Size(100, 30);

this.btnSend.TabIndex = 1;

this.btnSend.Text = "Send";

this.btnSend.UseVisualStyleBackColor = true;

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

this.Controls.Add(this.txtMessage);

this.Controls.Add(this.btnSend);

}

private System.Windows.Forms.TextBox txtMessage;

private System.Windows.Forms.Button btnSend;

}

}

Step6: Update “Form1.cs” code of “Consumer” by below

using System;

using System.Windows.Forms;

using Confluent.Kafka;

using System.Threading.Tasks;

namespace Consumer

{

public partial class Form1 : Form

{

private ConsumerConfig \_consumerConfig;

private IConsumer<Ignore, string> \_consumer;

private const string KafkaBootstrapServers = "localhost:9092";

private const string KafkaConsumerGroupId = "my\_consumer\_group";

public Form1()

{

InitializeComponent();

InitializeKafkaConsumer();

}

private void InitializeKafkaConsumer()

{

\_consumerConfig = new ConsumerConfig

{

BootstrapServers = KafkaBootstrapServers,

GroupId = KafkaConsumerGroupId,

AutoOffsetReset = AutoOffsetReset.Earliest

};

try

{

\_consumer = new ConsumerBuilder<Ignore, string>(\_consumerConfig)

.SetErrorHandler((consumer, err) =>

{

if (err.IsFatal)

{

MessageBox.Show($"Fatal Kafka error: {err.Reason}. Consumer will stop.", "Kafka Consumer Fatal Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

Console.WriteLine($"Kafka Consumer Error: {err.Reason}");

}

})

.SetLogHandler((consumer, log) =>

{

})

.Build();

Task.Run(() => ConsumeMessages());

lblStatus.Text = $"Listening to topic:";

}

catch (Exception ex)

{

MessageBox.Show($"Failed to initialize Kafka consumer: {ex.Message}", "Consumer Initialization Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

lblStatus.Text = "Consumer initialization failed.";

}

}

private void ConsumeMessages()

{

try

{

while (\_consumer != null)

{

var consumeResult = \_consumer.Consume(TimeSpan.FromSeconds(1));

if (consumeResult.IsPartitionEOF)

{

Console.WriteLine($"Reached end of topic {consumeResult.TopicPartitionOffset}.");

continue;

}

if (consumeResult.Message != null)

{

string receivedMessage = consumeResult.Message.Value;

if (txtReceivedMessages.InvokeRequired)

{

txtReceivedMessages.Invoke(new Action(() => AddMessageToDisplay(receivedMessage)));

}

else

{

AddMessageToDisplay(receivedMessage);

}

}

}

}

catch (ConsumeException ex)

{

MessageBox.Show($"Consume error: {ex.Error.Reason}", "Kafka Consume Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"An unexpected error occurred during consumption: {ex.Message}", "Unexpected Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void AddMessageToDisplay(string message)

{

txtReceivedMessages.AppendText($"Received: {message}{Environment.NewLine}");

txtReceivedMessages.ScrollToCaret();

}

protected override void OnFormClosed(FormClosedEventArgs e)

{

try

{

\_consumer?.Close();

}

catch (Exception ex)

{

Console.WriteLine($"Error closing Kafka consumer: {ex.Message}");

}

finally

{

\_consumer?.Dispose();

\_consumer = null;

}

base.OnFormClosed(e);

}

}

}

Step7: Update “Form1.Designer.cs” code of “Consumer” by below

namespace Consumer

{

partial class Form1

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

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

this.lblStatus = new System.Windows.Forms.Label();

this.SuspendLayout();

this.txtReceivedMessages.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

| System.Windows.Forms.AnchorStyles.Left)

| System.Windows.Forms.AnchorStyles.Right)));

this.txtReceivedMessages.Location = new System.Drawing.Point(12, 41);

this.txtReceivedMessages.Multiline = true;

this.txtReceivedMessages.Name = "txtReceivedMessages";

this.txtReceivedMessages.ReadOnly = true;

this.txtReceivedMessages.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;

this.txtReceivedMessages.Size = new System.Drawing.Size(776, 397);

this.txtReceivedMessages.TabIndex = 1;

this.lblStatus.AutoSize = true;

this.lblStatus.Location = new System.Drawing.Point(12, 17);

this.lblStatus.Name = "lblStatus";

this.lblStatus.Size = new System.Drawing.Size(100, 13);

this.lblStatus.TabIndex = 0;

this.lblStatus.Text = "Initializing Consumer...";

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.lblStatus);

this.Controls.Add(this.txtReceivedMessages);

this.Name = "Form1";

this.Text = "Kafka Consumer";

this.ResumeLayout(false);

this.PerformLayout();

}

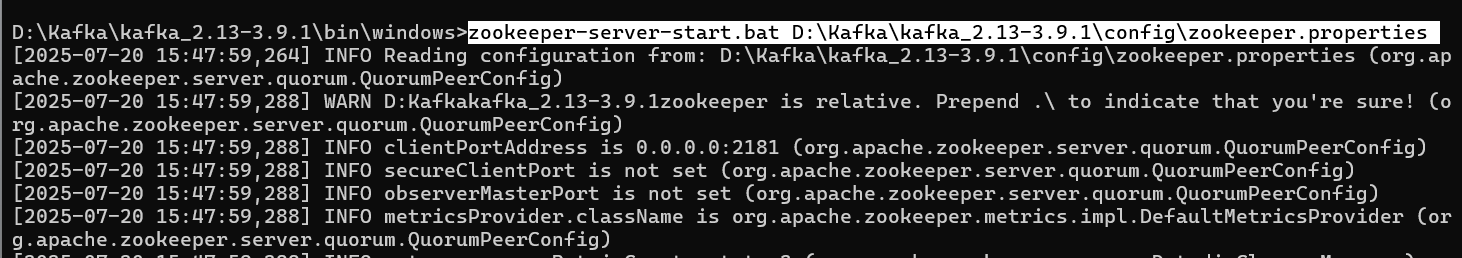
private System.Windows.Forms.TextBox txtReceivedMessages;

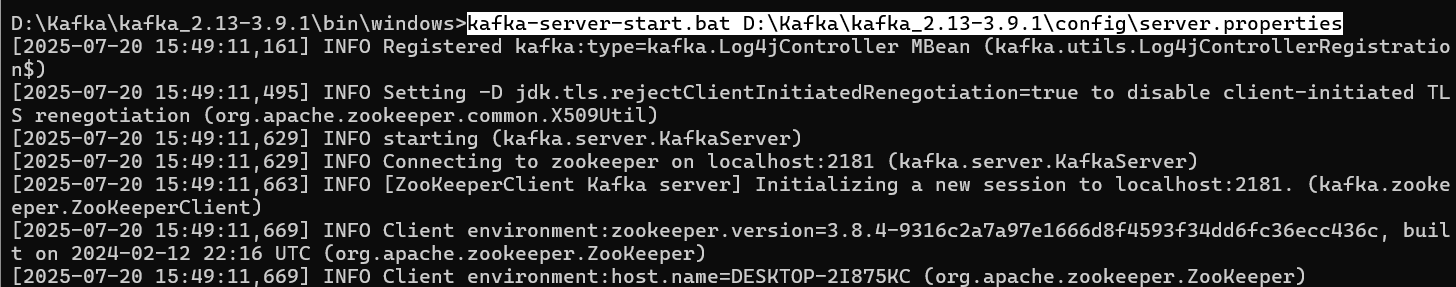
private System.Windows.Forms.Label lblStatus;

}

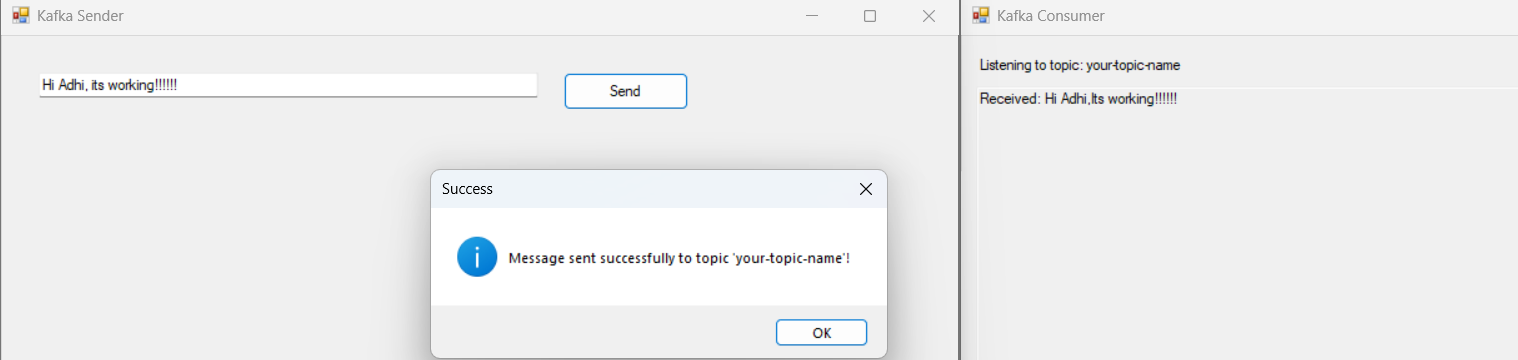
}

Step8:Turn on zookeeper,Kafka server





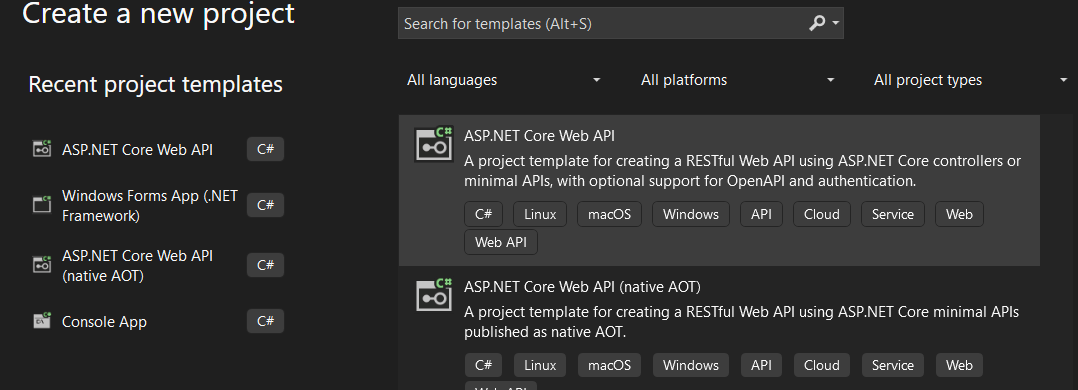
Step9:Now run both “Producer” and “Consumer” and send message from sender and check reciving in consumer

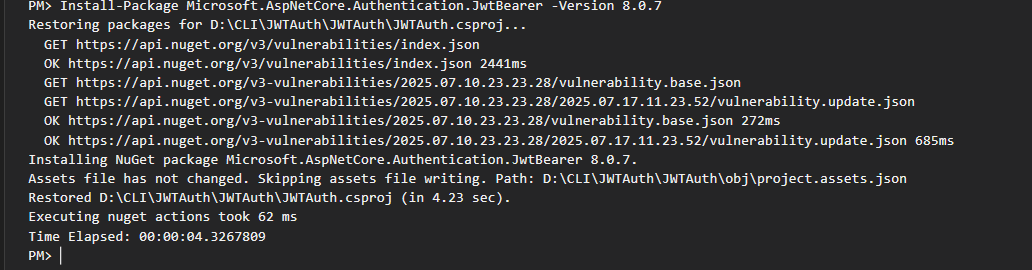


**Module7-Microservices Architecture using ASP.NET Core Web API**

**Ex1:Microservices – JWT**

**Step1: create project with “ASP.NET Core Web API” template**

****

**Step2: install JWTBearer package through package Manager**

**Step3: Create folder-“models” and add “LoginModel.cs” and add below code**

namespace JWTAuth.models

{

public class LoginModel

{

public string Username { get; set; } = "";

public string Password { get; set; } = "";

}

}

**Step4: add “AuthController.cs” inside “Controller” folder and add below code**

using JWTAuth.models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JWTAuth.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_configurationSettings;

public AuthController(IConfiguration config)

{

\_configurationSettings = config;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel userInput)

{

if (IsValidUser(userInput))

{

var generatedToken = GenerateJwtToken(userInput.Username);

return Ok(new { token = generatedToken });

}

return Unauthorized();

}

[Authorize]

[HttpGet("secret")]

public IActionResult SecretEndpoint()

{

return Ok("Authorized and can access protected endpoint.");

}

[Authorize]

[HttpGet("data")]

public IActionResult GetUserData()

{

var userName = User.Identity?.Name;

return Ok($"This is protected data.{userName}!");

}

private bool IsValidUser(LoginModel credentials)

{

return credentials.Username == "adhi" && credentials.Password == "adhi@123";

}

private string GenerateJwtToken(string userName)

{

var userClaims = new[]

{

new Claim(ClaimTypes.Name, userName),

new Claim(ClaimTypes.Role, "User")

};

var jwtKey = \_configurationSettings["Jwt:Key"]!;

var jwtIssuer = \_configurationSettings["Jwt:Issuer"]!;

var jwtAudience = \_configurationSettings["Jwt:Audience"]!;

var jwtDurationMinutes = double.Parse(\_configurationSettings["Jwt:DurationInMinutes"]!);

var securityKeyBytes = Encoding.UTF8.GetBytes(jwtKey);

var signingKey = new SymmetricSecurityKey(securityKeyBytes);

var credentials = new SigningCredentials(signingKey, SecurityAlgorithms.HmacSha256);

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(userClaims),

Expires = DateTime.UtcNow.AddMinutes(jwtDurationMinutes),

SigningCredentials = credentials,

Issuer = jwtIssuer,

Audience = jwtAudience

};

var tokenHandler = new JwtSecurityTokenHandler();

var token = tokenHandler.CreateToken(tokenDescriptor);

return tokenHandler.WriteToken(token);

}

}

}

**Step5: update “appsettings.json” by below**

{

"Jwt": {

"Key": "ThisIsAStrongSecretKeyForJwtToken@123123123",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 10

}

}

**Step6: update “Program.cs” by below**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = builder.Configuration["Jwt:Issuer"],

ValidAudience = builder.Configuration["Jwt:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]!)

)

};

});

builder.Services.AddAuthorization();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(swaggerConfig =>

{

swaggerConfig.SwaggerDoc("v1", new OpenApiInfo { Title = "Secure API", Version = "v1" });

swaggerConfig.AddSecurityDefinition("BearerAuth", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your valid JWT token.\nExample: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

});

swaggerConfig.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "BearerAuth"

}

},

Array.Empty<string>()

}

});

});

var appInstance = builder.Build();

if (appInstance.Environment.IsDevelopment())

{

appInstance.UseSwagger();

appInstance.UseSwaggerUI();

}

appInstance.UseAuthentication();

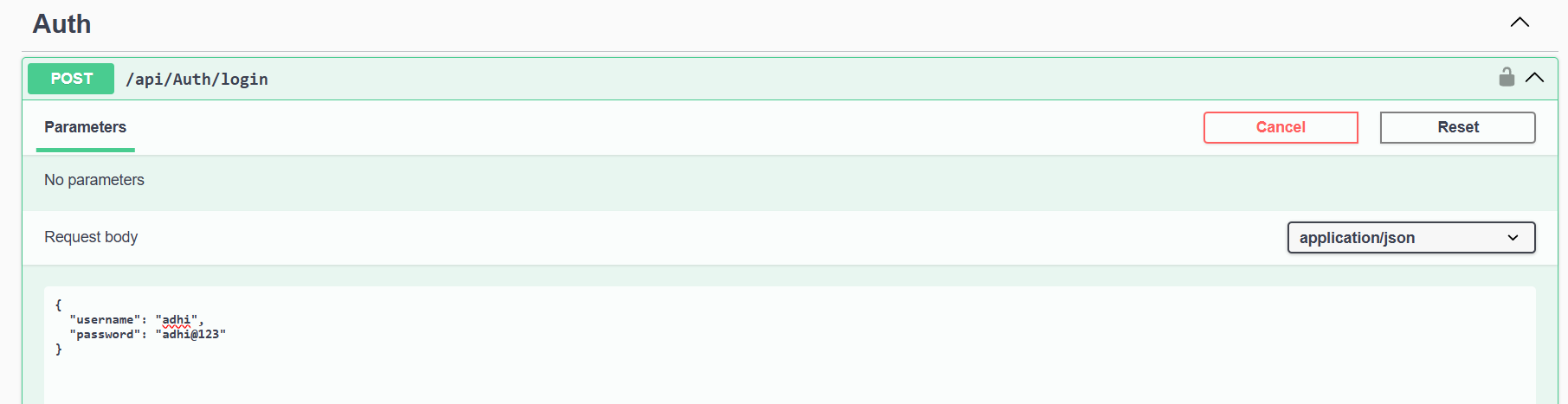
appInstance.UseAuthorization();

appInstance.MapControllers();

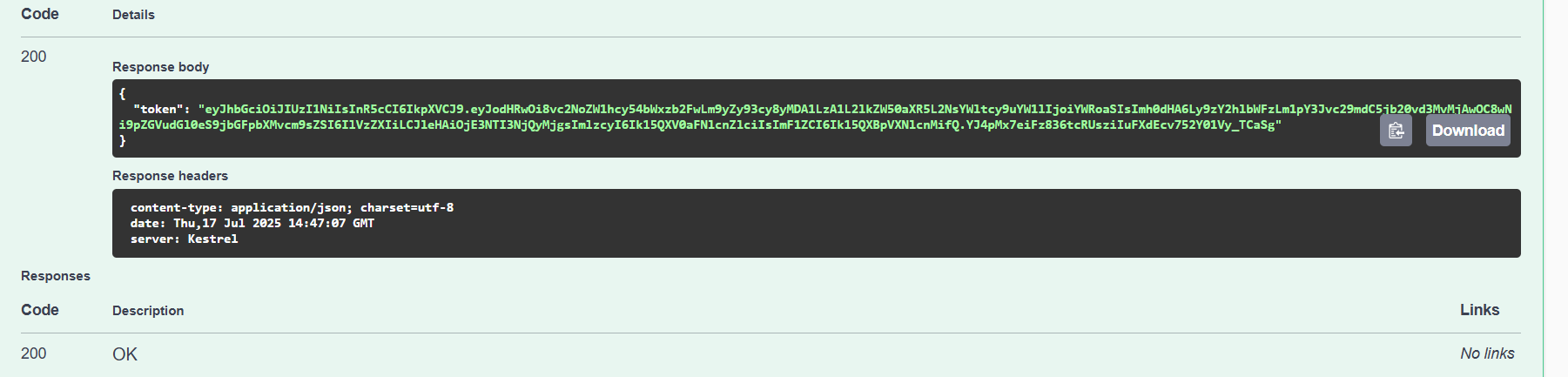
appInstance.Run();

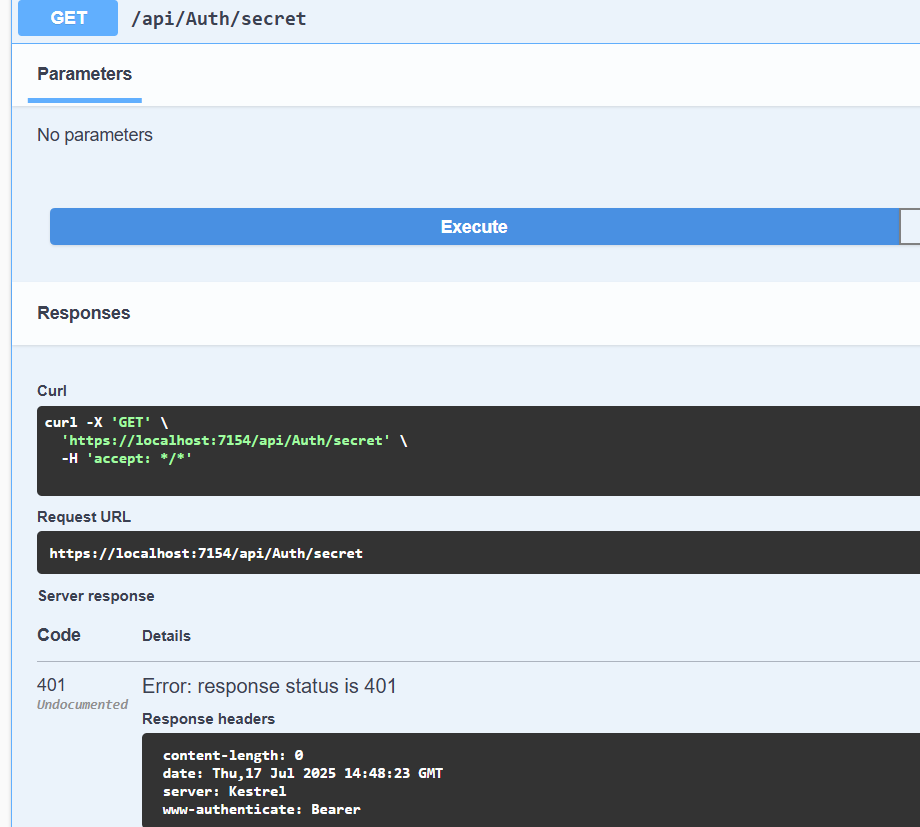
**Output:**

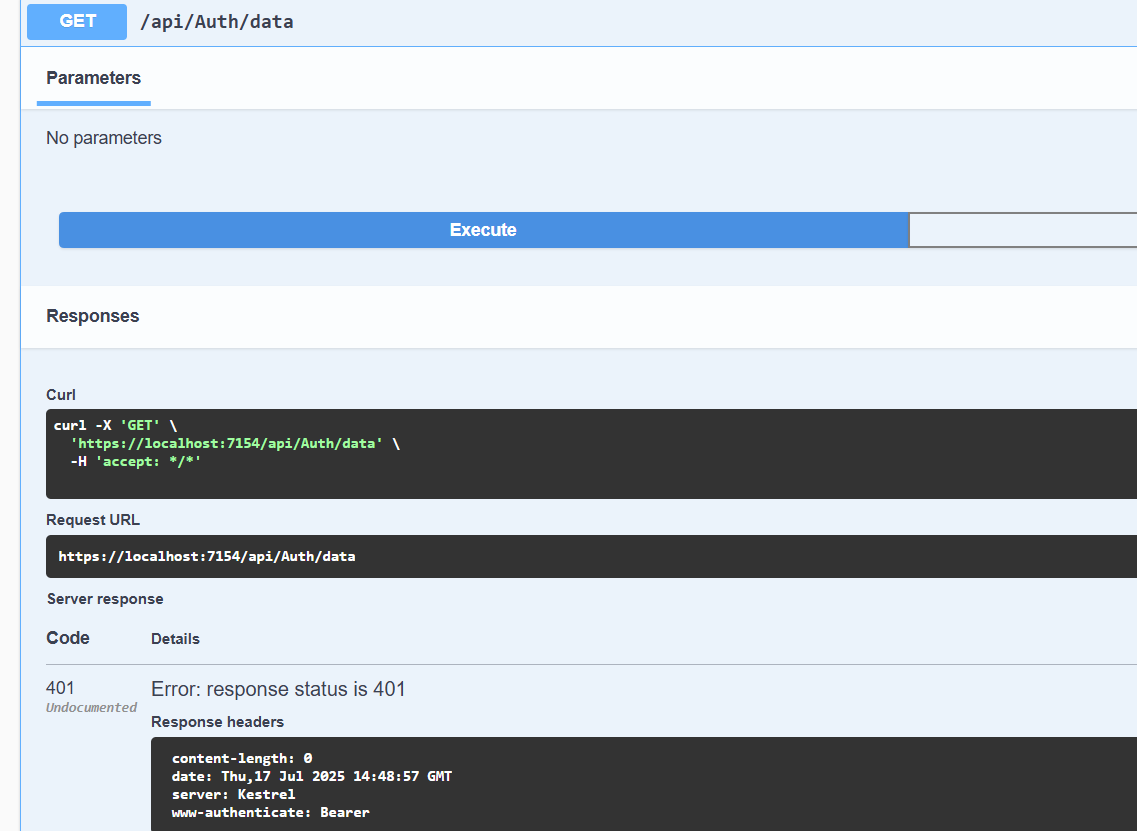
**Give user name and password as per given in program.cs**

****

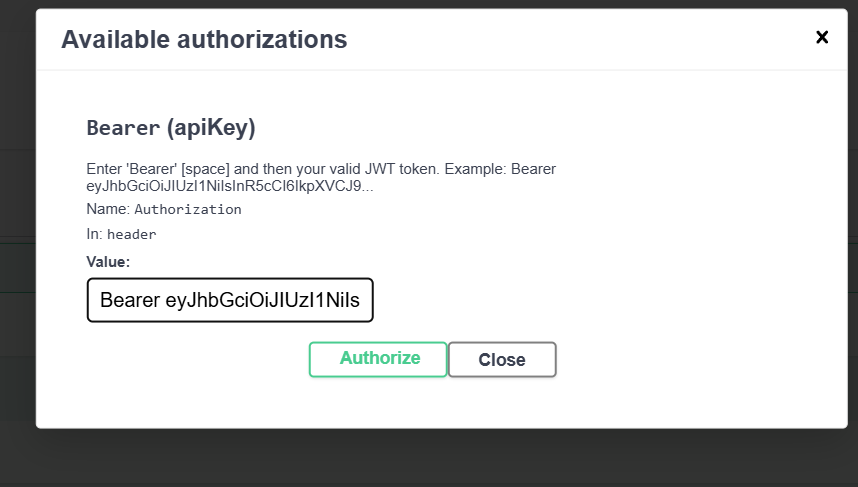
**Now execute and get token**

****

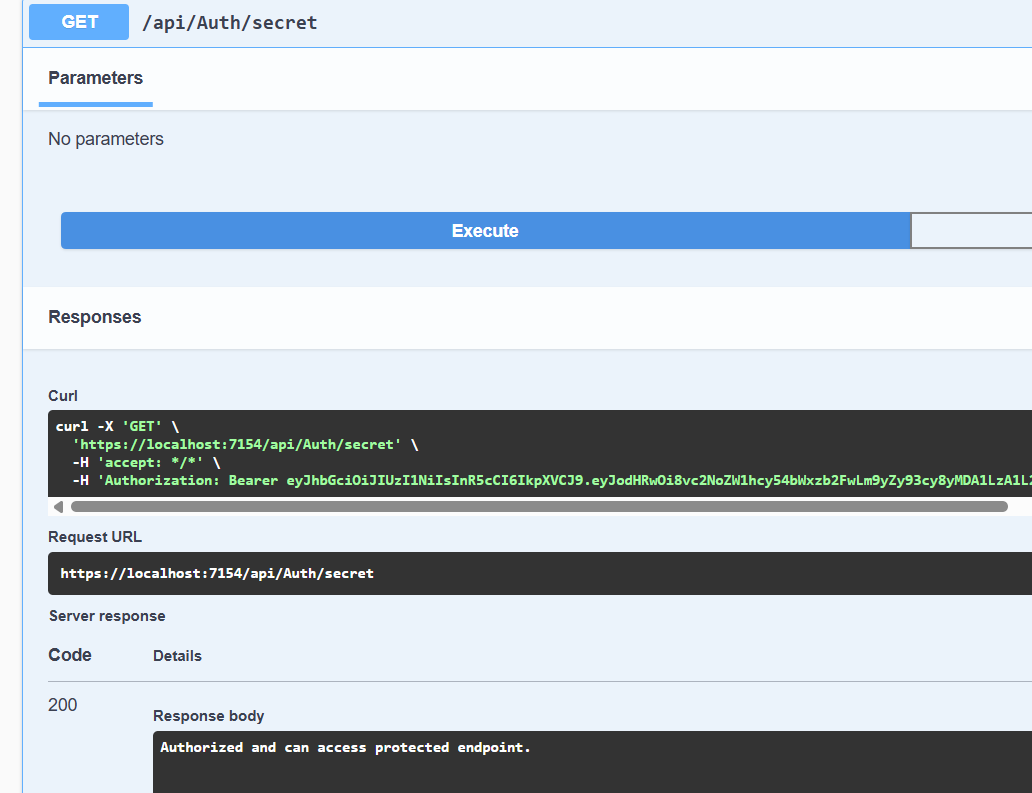
**While attempting to access “GET/api/Auth/secret” without Authorization we get 401 Unauthorized error **

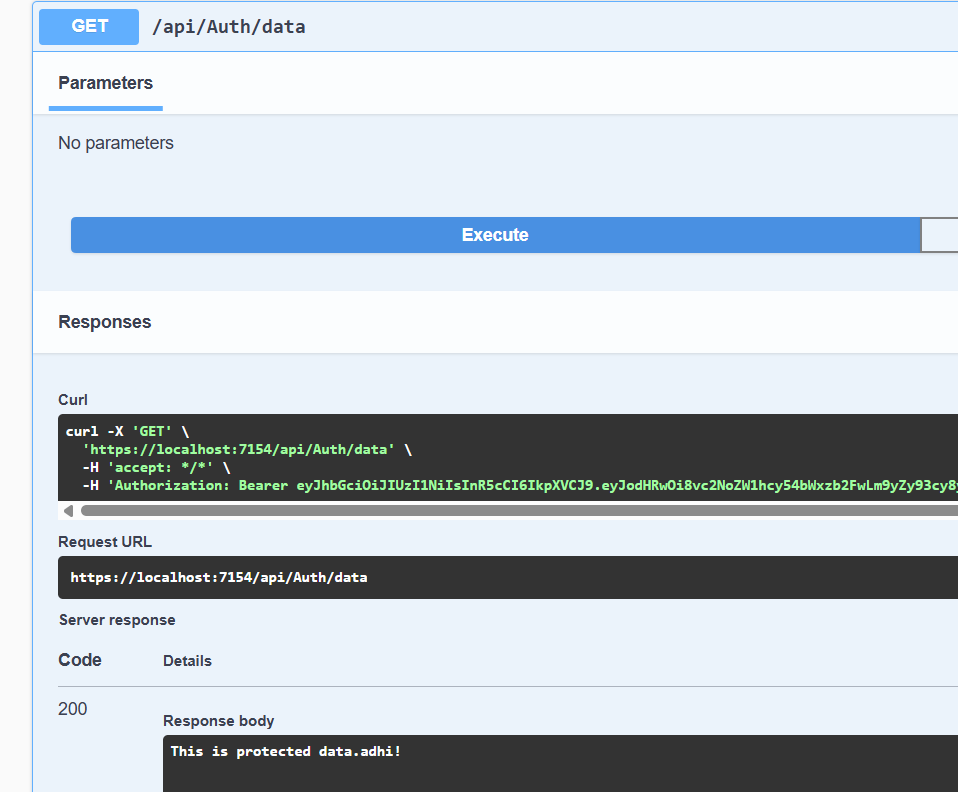
**While attempting to access “GET/api/Auth/data” without Authorization we get 401 Unauthorized error**

**Authorizing by using “Bearer <token>”**

****

**While attempting to access “GET/api/Auth/secret” with Authorization we get 200 code**

****

**While attempting to access “GET/api/Auth/data” with Authorization we get 200 code **