**Web\_API\_HandsOn\_6 (Kafka Integration with C#)**

Q1.) Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prompt.

**Chat Message Screenshot :**

A screenshot of a computer

AI-generated content may be incorrect.

Q2. Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.

**Producer Code (Program.cs) :**

using System;

using System.Threading.Tasks;

using Confluent.Kafka;

class Program

{

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 ('exit' to quit):");

while (true)

{

var messageValue = Console.ReadLine();

if (messageValue == "exit") break;

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

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

}

}

}

**Producer Code Terminal Screenshot :**

A screenshot of a computer

AI-generated content may be incorrect.

**Consumer Code (Program.cs) :**

using System;

using Confluent.Kafka;

class Program

{

static void Main(string[] args)

{

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

GroupId = "chat-consumer-group",

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("test-topic");

Console.WriteLine("Listening to chat messages...");

try

{

while (true)

{

var consumeResult = consumer.Consume();

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

}

}

catch (OperationCanceledException)

{

consumer.Close();

}

}

}

**Consumer Code Terminal Screenshot :**

A screenshot of a computer

AI-generated content may be incorrect.

**Microservices**

**Question 1: Implement JWT Authentication in ASP.NET Core Web API**

**Codes:**

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

namespace JwtAuthDemo

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new() { Title = "JwtAuthDemo", Version = "v1" });

c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Description = "JWT Authorization header using the Bearer scheme (Example: 'Bearer {token}')",

Name = "Authorization",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

// JWT Authentication Configuration

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();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Add authentication & authorization middleware

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}

**AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_configuration;

public AuthController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized("Invalid username or password");

}

private bool IsValidUser(LoginModel model)

{

// Hardcoded user validation (you can replace with database validation)

return model.Username == "Soumili" && model.Password == "6362268";

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Jwt:Key"]));

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

var token = new JwtSecurityToken(

issuer: \_configuration["Jwt:Issuer"],

audience: \_configuration["Jwt:Audience"],

claims: claims,

expires: DateTime.Now.AddMinutes(60),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

}

**Screenshot: (Without Authorize)**

A screenshot of a computer

AI-generated content may be incorrect.

**UserController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class UserController : ControllerBase

{

[HttpGet("getdata")]

[Authorize]

public IActionResult GetData()

{

var username = User.Identity?.Name;

return Ok($"Hello {username}, you have accessed a protected endpoint!");

}

}

}

**Protected Endpoint :**

A screenshot of a computer

AI-generated content may be incorrect.