**Microservice HandsOn:**

**program.cs:**

**using Microsoft.AspNetCore.Authentication.JwtBearer;**

**using Microsoft.IdentityModel.Tokens;**

**using System.Text;**

**var builder = WebApplication.CreateBuilder(args);**

**// Add services to the container.**

**builder.Services.AddControllers();**

**builder.Services.AddEndpointsApiExplorer();**

**builder.Services.AddSwaggerGen();**

**builder.Services.AddAuthentication("Bearer")**

**.AddJwtBearer("Bearer", 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();**

**app.UseAuthentication();**

**app.UseAuthorization();**

**app.MapControllers();**

**app.Run();**

**login.cs:**

**namespace MicroService.Model**

**{**

**public class Login**

**{**

**public string Username { get; set; }**

**public string Password { get; set; }**

**}**

**}**

**user.cs:**

**namespace MicroService.Model**

**{**

**public class User**

**{**

**public string Username { get; set; }**

**public string Password { get; set; }**

**}**

**}**

**Authcontroller.cs:**

**using System.Security.Claims;**

**using System.Text;**

**using Microsoft.AspNetCore.Http;**

**using Microsoft.AspNetCore.Mvc;**

**using MicroService.Model;**

**using System.IdentityModel.Tokens.Jwt;**

**using Microsoft.IdentityModel.Tokens;**

**namespace MicroService.Controllers**

**{**

**[ApiController]**

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

**public class AuthController : ControllerBase**

**{**

**[HttpPost("login")]**

**public IActionResult Login([FromBody] Login model)**

**{**

**if (IsValidUser(model))**

**{**

**var token = GenerateJwtToken(model.Username);**

**return Ok(new { Token = token });**

**}**

**return Unauthorized();**

**}**

**private bool IsValidUser(Login model)**

**{**

**return model.Username == "User" && model.Password == "12345";**

**}**

**private string GenerateJwtToken(string username)**

**{**

**var claims = new[]**

**{**

**new Claim(ClaimTypes.Name, username)**

**};**

**var key = new**

**SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisIsASecretKeyForJwtTokenForPractice"));**

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

**var token = new JwtSecurityToken(**

**issuer: "MyAuthServer",**

**audience: "MyApiUsers",**

**claims: claims,**

**expires: DateTime.Now.AddMinutes(60),**

**signingCredentials: creds);**

**return new JwtSecurityTokenHandler().WriteToken(token);**

**}**

**}**

**}**

**securecontroller.cs:**

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Mvc;**

**[ApiController]**

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

**public class SecureController : ControllerBase**

**{**

**[HttpGet("data")]**

**[Authorize]**

**public IActionResult GetSecureData()**

**{**

**return Ok("This is protected data.");**

**}**

**}**

**WebAPI-6 HandsOn:**

**FORM1.CS:**

using KafkaNet;

using KafkaNet.Model;

using KafkaNet.Protocol;

using System;

using System.Collections.Generic;

using System.Threading;

using System.Windows.Forms;

using KafkaMessage = KafkaNet.Protocol.Message;

namespace KafkaHandson

{

public partial class Form1 : Form

{

private const string TopicName = "chat-topic";

private static readonly Uri KafkaUri = new Uri("http://localhost:9092");

public Form1()

{

InitializeComponent();

}

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

{

if (string.IsNullOrWhiteSpace(txtMessage.Text))

{

MessageBox.Show("Please Enter Message", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

string payload = txtMessage.Text.Trim();

Thread sendMessage = new Thread(() =>

{

var message = new KafkaMessage(payload);

var options = new KafkaOptions(KafkaUri);

var router = new BrokerRouter(options);

var client = new Producer(router);

client.SendMessageAsync(TopicName, new List<KafkaMessage> { message }).Wait();

});

sendMessage.Start();

txtMessage.Clear();

}

}

}