Week - 5 : (HandsOn Exercises)

Name : Aditya Raj

SuperSet ID : 6363544

**Lab6- WebApi\_Handson**

**Kafka Integration with C#:**

Apache Kafka is a distributed event streaming platform used for high-

performance data pipelines, streaming analytic, data integration, and

mission-critical applications. It allows:

 Producers to send data (messages),

 Topics to categorize the data,

 Consumers to read it.

**Kafka Architecture Overview**

 Producer: Sends messages to a topic.

 Topic: Logical channel for messages.

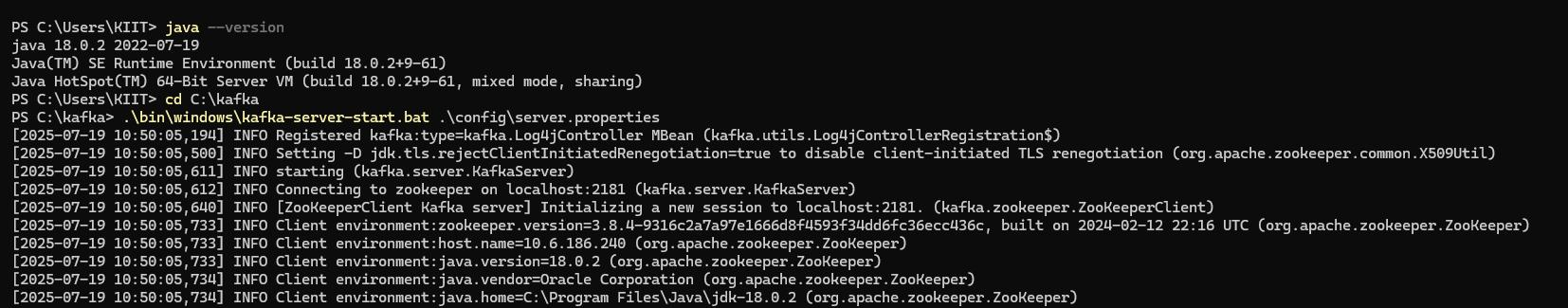
 Partition: Sub-division of a topic for load balancing.

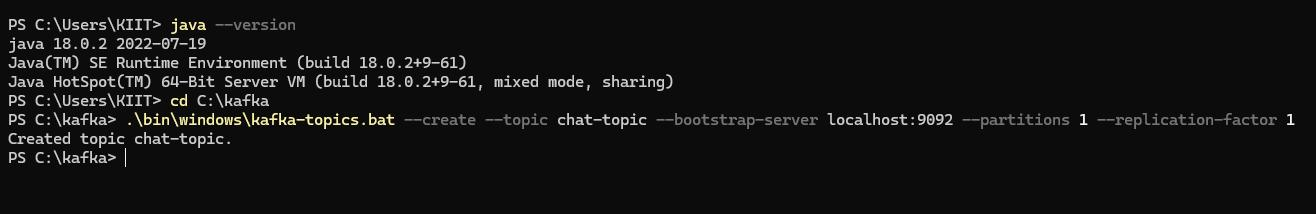
 Broker: Kafka server storing data.

 Consumer: Reads messages from topics.

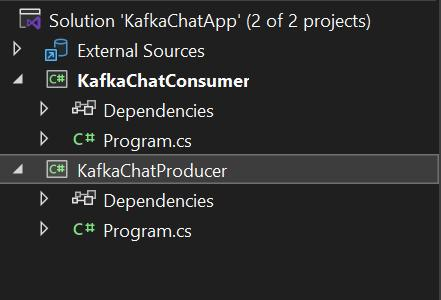
 Zookeeper: Manages Kafka brokers and cluster coordination.







**Solution directory for hands on 1:**

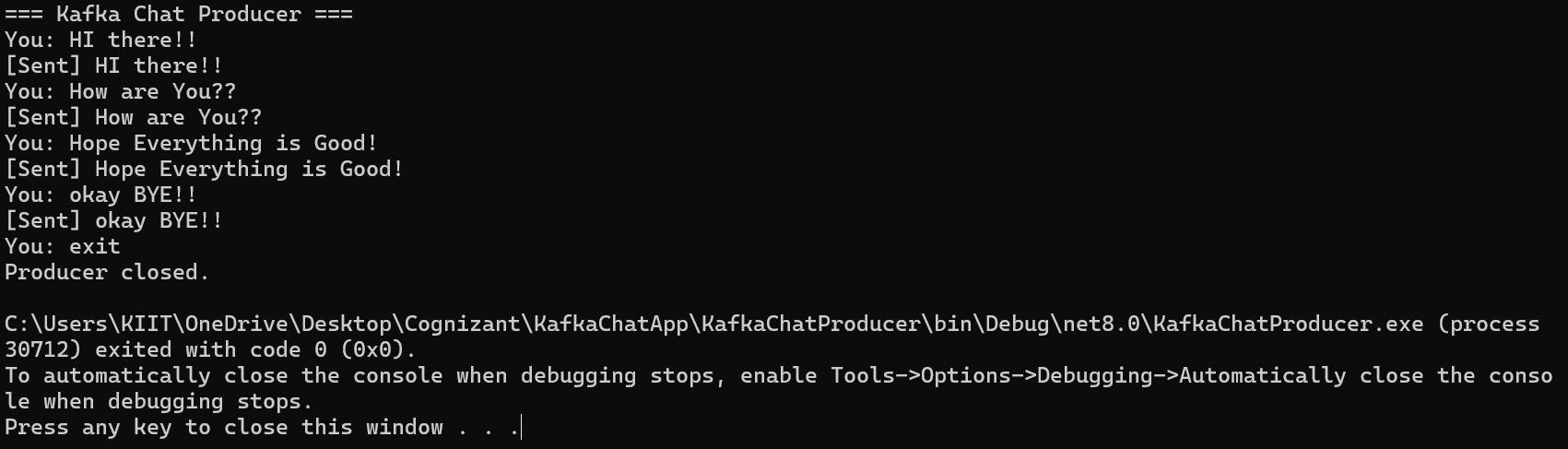


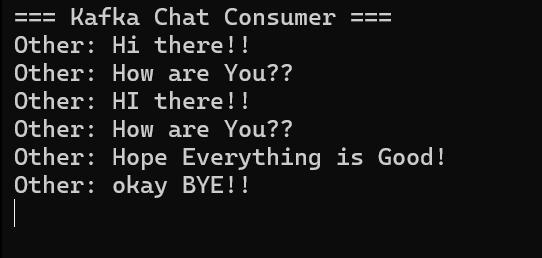
**1. Creating a Chat Application which uses Kafka as a streaming**

**platform and consuming the chat messages in the command prompt’s**

**output.**

**Chat in Producer**





1. Hands-On Exercises: Authentication and Authorization in ASP.NET Core

Web API Microservices

This document contains 4 hands-on exercises focusing on Authentication and Authorization

in ASP.NET Core Web API microservices, with an emphasis on implementing JWT (JSON

Web Tokens) authentication. Each exercise includes a scenario, step-by-step instructions,

and complete solution code.

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

Scenario:

You are building a microservice that requires secure login. You need to implement JWT based authentication.

Steps:

1. Create a new ASP.NET Core Web API project.

2. Add a `User` model and a login endpoint.

3. Generate a JWT token upon successful login.

4. Secure an endpoint using `[Authorize]`.

Solution Code:

Install NuGet Packages:

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

appsettings.json:

{

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

}

}

Program.cs:

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();

AuthController.cs:

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new

SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisIsASecretKeyForJwtToken"));

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);

}

}

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var config = builder.Configuration;

Console.WriteLine("JWT KEY: " + config["Jwt:Key"]);

var key = Encoding.UTF8.GetBytes(config["Jwt:Key"]);

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = config["Jwt:Issuer"],

ValidAudience = config["Jwt:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(key)

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

{

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

namespace JwtAuthDemo.Models

{

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

}

using JwtAuthDemo.Models;

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 \_config;

public AuthController(IConfiguration config)

{

\_config = config;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

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

{

var token = GenerateToken(model.Username);

return Ok(new { token });

}

return Unauthorized();

}

private string GenerateToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

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

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

var token = new JwtSecurityToken(

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

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

claims: claims,

expires: DateTime.Now.AddMinutes(Convert.ToDouble(\_config["Jwt:DurationInMinutes"])),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

