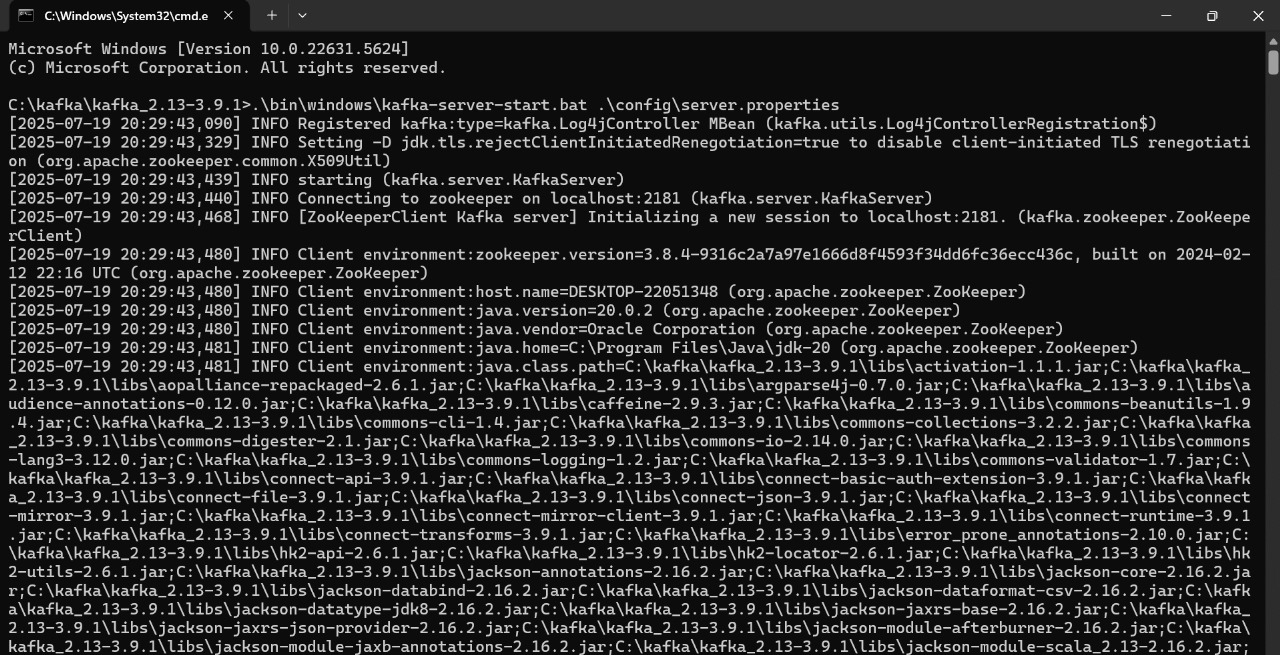
**NAME-MASHEERA AFRIN**

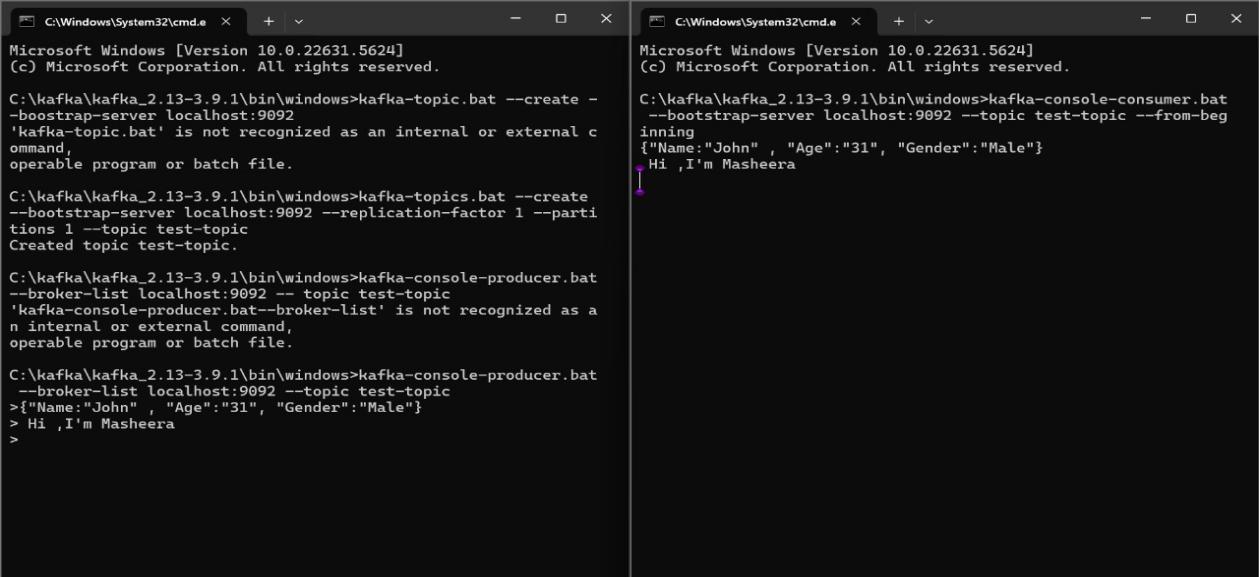
**SUPERSET ID: 6362071**

**WEBAPI HANDSON-6**

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

****

****

****

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

**Consumer.cs**

using Confluent.Kafka;

using System;

using System.Threading;

using System.Threading.Tasks;

public class Consumer

{

private readonly string \_bootstrapServers;

private readonly string \_topic;

public Consumer(string bootstrapServers, string topic)

{

\_bootstrapServers = bootstrapServers;

\_topic = topic;

}

public void StartConsuming(CancellationToken cancellationToken)

{

var config = new ConsumerConfig

{

BootstrapServers = \_bootstrapServers,

GroupId = Guid.NewGuid().ToString(),

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe(\_topic);

try

{

while (!cancellationToken.IsCancellationRequested)

{

var result = consumer.Consume(cancellationToken);

Console.WriteLine(result.Message.Value);

}

}

catch (OperationCanceledException) { }

finally

{

consumer.Close();

}

}

}

**Producer.cs**

using Confluent.Kafka;

using System.Threading.Tasks;

public class Producer

{

private readonly IProducer<Null, string> \_producer;

private readonly string \_topic;

public Producer(string bootstrapServers, string topic)

{

var config = new ProducerConfig { BootstrapServers = bootstrapServers };

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

\_topic = topic;

}

public async Task SendMessage(string message)

{

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

}

}

using System;

using System.Threading;

using System.Threading.Tasks;

**Program.cs**

class Program

{

static async Task Main(string[] args)

{

var producer = new Producer("localhost:9092", "chat-messages");

var consumer = new Consumer("localhost:9092", "chat-messages");

var cts = new CancellationTokenSource();

var consumerTask = Task.Run(() => consumer.StartConsuming(cts.Token));

Console.Write("Enter your name: ");

string? user = Console.ReadLine();

if (string.IsNullOrWhiteSpace(user))

user = "Anonymous";

Console.WriteLine("Start chatting! (type 'exit' to quit)");

while (true)

{

string? msg = Console.ReadLine();

if (string.IsNullOrWhiteSpace(msg)) continue;

if (msg.ToLower() == "exit")

{

cts.Cancel();

await consumerTask;

break;

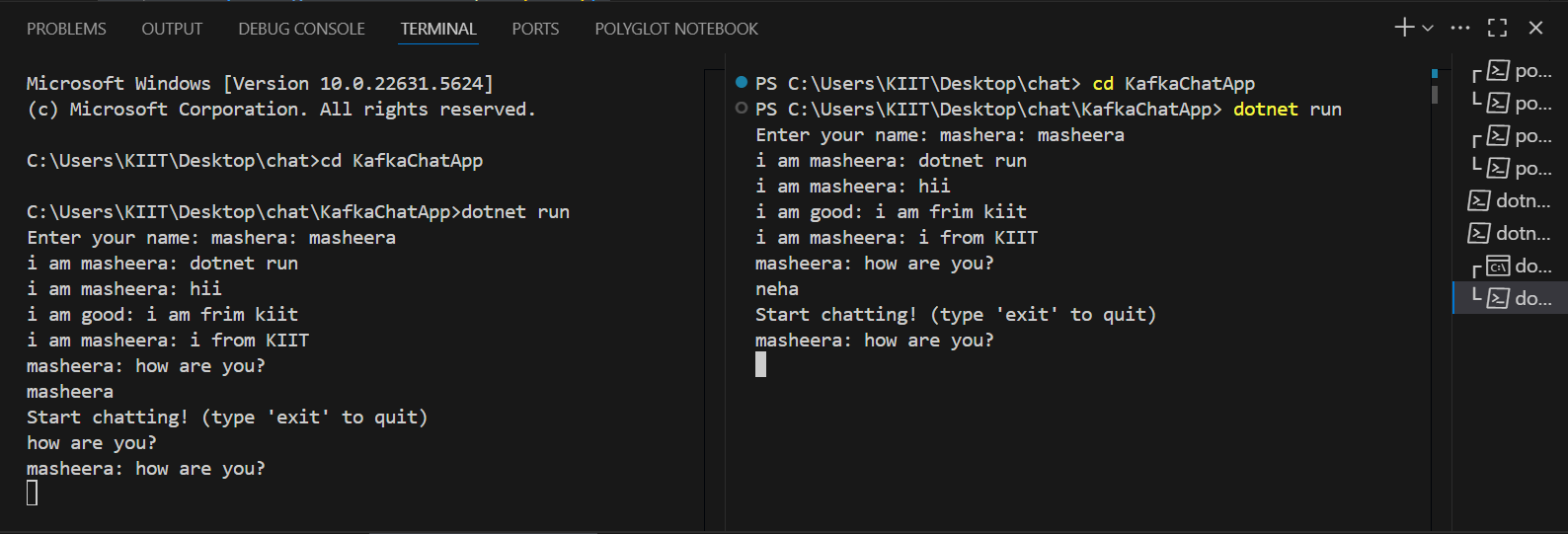
}

await producer.SendMessage($"{user}: {msg}");

}

}

}



**MICROSERVICES**

**AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthDemo.Models;

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

{

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 keyString = \_configuration["Jwt:Key"];

if (string.IsNullOrEmpty(keyString))

throw new InvalidOperationException("JWT key is missing in configuration.");

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(keyString));

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);

}

}

**ValueController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

[Authorize]

public IActionResult Get()

{

return Ok("You are authorized!");

}

[HttpGet("secret")]

[Authorize]

public IActionResult GetSecret()

{

return Ok("Top secret value!");

}

}

**PROGRAM.CS**

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();

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();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Models/LoginModel.cs**

namespace JwtAuthDemo.Models

{

public class LoginModel

{

public required string Username { get; set; }

public required string Password { get; set; }

}

}

