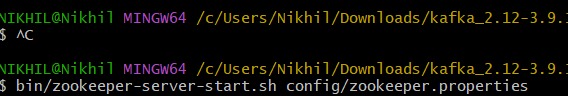
**HANDS ON WEEK 5**

**ASP.NETCore 8.0 Web API**

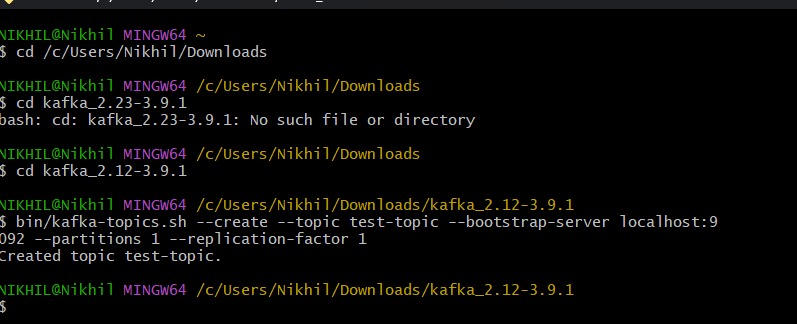
**6.WEB\_API Handson**

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



****

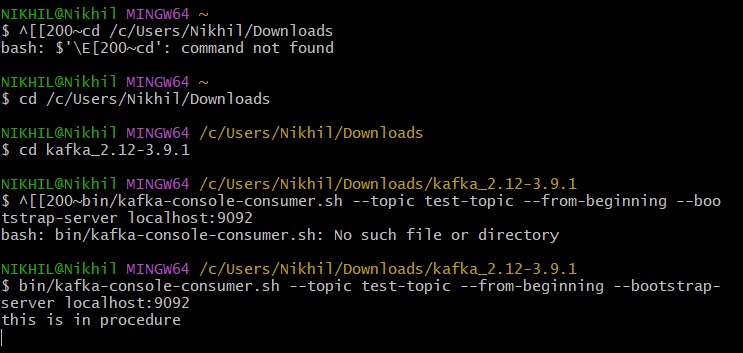
**CREATING A TOPIC :**



**PRODUCER :**



**CONSUMER** :



**KAFKA PRODUCER APP :**

producer.cs :

using System;

using Confluent.Kafka;

class Program

{

static async Task Main(string[] args)

{

var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

Console.WriteLine("Kafka Chat Producer Started. Type your message:");

using var producer = new ProducerBuilder<Null, string>(config).Build();

while (true)

{

var message = Console.ReadLine();

if (message == "exit") break;

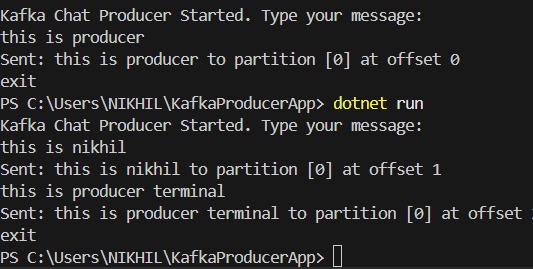
var result = await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = message });

Console.WriteLine($"Sent: {message} to partition {result.Partition} at offset {result.Offset}");

       }

    }

}



**KAFKA CONSUMER APP :**

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

};

Console.WriteLine("Kafka Chat Consumer Started...");

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

consumer.Subscribe("chat-topic");

try

{

while (true)

{

var cr = consumer.Consume();

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

}

}

catch (OperationCanceledException)

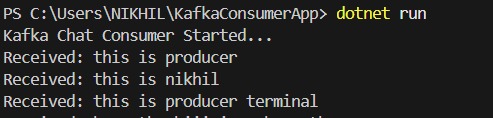
{

consumer.Close();

       }

    }

}



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

**Form1.cs :**

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using Confluent.Kafka;

namespace KafkaChatGUI

{

public partial class Form1 : Form

{

private IProducer<Null, string> \_producer;

private CancellationTokenSource \_cancellationTokenSource;

private string \_username;

public Form1()

{

\_username = PromptForUsername();

InitializeComponent();

\_producer = new ProducerBuilder<Null, string>(

new ProducerConfig { BootstrapServers = "localhost:9092" }).Build();

\_cancellationTokenSource = new CancellationTokenSource();

StartConsumer();

}

private string PromptForUsername()

{

string input = "";

while (string.IsNullOrWhiteSpace(input))

{

input = Microsoft.VisualBasic.Interaction.InputBox(

"Enter your username:", "Username", "User");

}

return input;

}

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

{

if (!string.IsNullOrWhiteSpace(txtMessage.Text))

{

string message = $"{\_username}: {txtMessage.Text}";

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

txtMessage.Clear();

}

}

private void StartConsumer()

{

Task.Run(() =>

{

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

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

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("chat-topic");

try

{

while (!\_cancellationTokenSource.Token.IsCancellationRequested)

{

var cr = consumer.Consume(\_cancellationTokenSource.Token);

AppendTextToChat(cr.Message.Value);

}

}

catch (OperationCanceledException) { }

finally

{

consumer.Close();

}

});

}

private void AppendTextToChat(string text)

{

if (txtChatLog.InvokeRequired)

{

txtChatLog.Invoke(new Action(() => {

txtChatLog.AppendText(text + Environment.NewLine);

}));

}

else

{

txtChatLog.AppendText(text + Environment.NewLine);

}

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

\_cancellationTokenSource.Cancel();

\_producer?.Dispose();

base.OnFormClosing(e);

       }

    }

}

**Form1design.cs :**

namespace KafkaChatGUI

{

partial class Form1

{

private System.ComponentModel.IContainer components = null;

private System.Windows.Forms.TextBox txtChatLog;

private System.Windows.Forms.TextBox txtMessage;

private System.Windows.Forms.Button btnSend;

protected override void Dispose(bool disposing)

{

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

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

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

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

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

this.SuspendLayout();

//

// txtChatLog

//

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

this.txtChatLog.Multiline = true;

this.txtChatLog.Name = "txtChatLog";

this.txtChatLog.ReadOnly = true;

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

this.txtChatLog.Size = new System.Drawing.Size(360, 200);

this.txtChatLog.TabIndex = 0;

//

// txtMessage

//

this.txtMessage.Location = new System.Drawing.Point(12, 220);

this.txtMessage.Name = "txtMessage";

this.txtMessage.Size = new System.Drawing.Size(260, 23);

this.txtMessage.TabIndex = 1;

//

// btnSend

//

this.btnSend.Location = new System.Drawing.Point(280, 220);

this.btnSend.Name = "btnSend";

this.btnSend.Size = new System.Drawing.Size(92, 23);

this.btnSend.TabIndex = 2;

this.btnSend.Text = "Send";

this.btnSend.UseVisualStyleBackColor = true;

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

//

// Form1

//

this.ClientSize = new System.Drawing.Size(384, 261);

this.Controls.Add(this.btnSend);

this.Controls.Add(this.txtMessage);

this.Controls.Add(this.txtChatLog);

this.Name = "Form1";

this.Text = "Kafka Chat";

this.ResumeLayout(false);

this.PerformLayout();

       }

    }

}

**Program.cs :**

using System;

using System.Windows.Forms;

namespace KafkaChatGUI

{

static class Program

{

[STAThread]

static void Main()

{

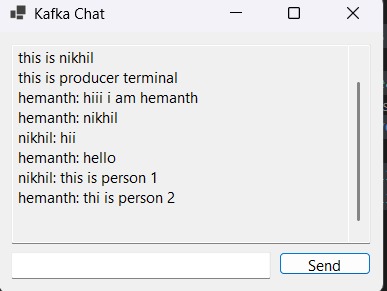
ApplicationConfiguration.Initialize(); // Or Application.EnableVisualStyles() + SetCompatibleTextRenderingDefault(false)

Application.Run(new Form1());

       }

    }

}



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

**1. Microservices - JWT**

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

**Autocontroller.cs**

using JwtAuthApi.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthApi.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();

}

private bool IsValidUser(LoginModel model)

{

// Dummy check for demo. Replace with real DB check.

return model.Username == "admin" && model.Password == "password";

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

// 🛡 Null-safe way to fetch configuration values

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

string? issuer = \_configuration["Jwt:Issuer"];

string? audience = \_configuration["Jwt:Audience"];

string? duration = \_configuration["Jwt:DurationInMinutes"];

if (string.IsNullOrEmpty(keyString) || string.IsNullOrEmpty(issuer) || string.IsNullOrEmpty(audience) || string.IsNullOrEmpty(duration))

{

throw new Exception("JWT configuration values are missing from appsettings.json.");

}

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

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

var token = new JwtSecurityToken(

issuer: issuer,

audience: audience,

claims: claims,

expires: DateTime.Now.AddMinutes(Convert.ToDouble(duration)),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

       }

    }

}

**Securecontroller.cs :**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthApi.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[Authorize]

[HttpGet("secret")]

public IActionResult GetSecret()

{

var username = User.Identity?.Name;

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

       }

    }

}

**program.cs :**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

string? jwtKey = builder.Configuration["Jwt:Key"];

string? jwtIssuer = builder.Configuration["Jwt:Issuer"];

string? jwtAudience = builder.Configuration["Jwt:Audience"];

if (string.IsNullOrEmpty(jwtKey) || string.IsNullOrEmpty(jwtIssuer) || string.IsNullOrEmpty(jwtAudience))

{

throw new Exception("JWT settings are missing in appsettings.json");

}

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtIssuer,

ValidAudience = jwtAudience,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey))

};

});

builder.Services.AddAuthorization();

builder.Services.AddControllers();

var app = builder.Build();

app.UseAuthentication();

app.UseAuthorization()

app.MapControllers();

app.Run();

