1. **WebApi.Handson**

**Producer(Chat Sender)**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class ChatProducer

{

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 (type 'exit' to quit):");

while (true)

{

var message = Console.ReadLine();

if (message == "exit") break;

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

Console.WriteLine($"Sent: {message}");

}

}

}

**Consumer(Chat Reciver)**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class ChatProducer

{

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 (type 'exit' to quit):");

while (true)

{

var message = Console.ReadLine();

if (message == "exit") break;

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

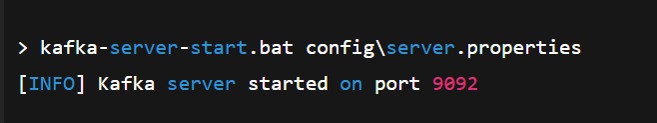
Console.WriteLine($"Sent: {message}");

}

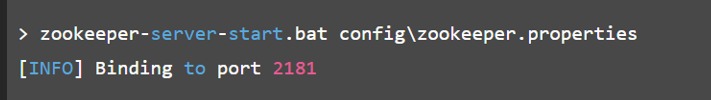
}

}

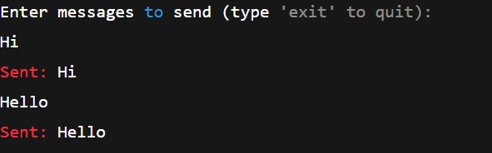
**Kafka Server Running:**

****

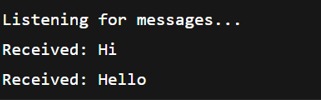
**Zookeeper Running:**

****

**Chat Producer:**

****

**Chat Consumer:**

****

**Microservices**

**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.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

var jwtSettings = builder.Configuration.GetSection("Jwt");

// Fix for possible null reference

var jwtKey = jwtSettings["Key"] ?? throw new ArgumentNullException("Jwt:Key");

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtSettings["Issuer"],

ValidAudience = jwtSettings["Audience"],

IssuerSigningKey = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes(jwtKey))

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

// Configure the HTTP request pipeline.

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**JwtAuthApi.http:**

@JwtAuthApi\_HostAddress = http://localhost:5042

GET {{JwtAuthApi\_HostAddress}}/weatherforecast/

Accept: application/json

**Appsettings.json:**

"Jwt": {

"Key": "mysupersecretkey1234567890123456",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Properties Folder:**

**Launchsettings.json:**

"profiles": {

"JwtAuthApi": {

"commandName": "Project",

"launchBrowser": true,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

},

"applicationUrl": "https://localhost:49946;http://localhost:49947"

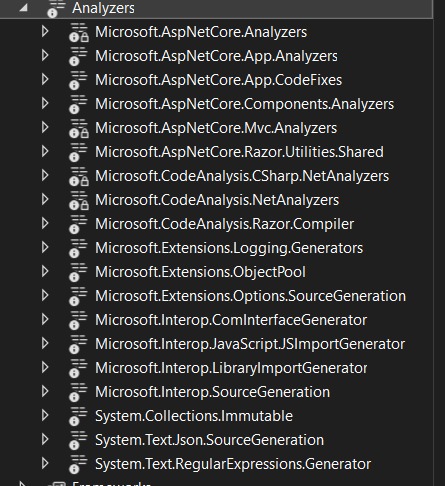
}

}

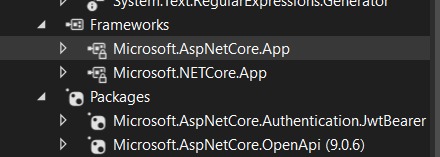
}

**Dependencies Folder:**

1. Analyzers:



1. Framework/Packages:



1. Folder Structure:

