**WEEK 5 HANDSON**

**Kafka Integration with C#:**

**CODE:**

**Program.cs:**

using System;

using System.Threading.Tasks;

class Program

{

    static async Task Main(string[] args)

    {

        Console.WriteLine("1: Producer | 2: Consumer");

        var option = Console.ReadLine();

        if (option == "1")

        {

            await Producer();

        }

        else if (option == "2")

        {

            Consumer();

        }

    }

    static async Task Producer()

    {

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

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

        Console.WriteLine("Type your chat messages (type 'exit' to quit):");

        while (true)

        {

            var input = Console.ReadLine();

            if (input == "exit") break;

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

            Console.WriteLine($"Sent: {input} (Partition: {result.Partition}, Offset: {result.Offset})");

        }

    }

    static void Consumer()

    {

        var config = new Confluent.Kafka.ConsumerConfig

        {

            BootstrapServers = "localhost:9092",

            GroupId = "chat-consumer-group",

            AutoOffsetReset = Confluent.Kafka.AutoOffsetReset.Earliest

        };

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

        consumer.Subscribe("chat-topic");

        Console.WriteLine("Listening for chat messages (Ctrl+C to quit)...");

        try

        {

            while (true)

            {

                var msg = consumer.Consume();

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

            }

        }

        catch (OperationCanceledException)

        {

            consumer.Close();

        }

    }

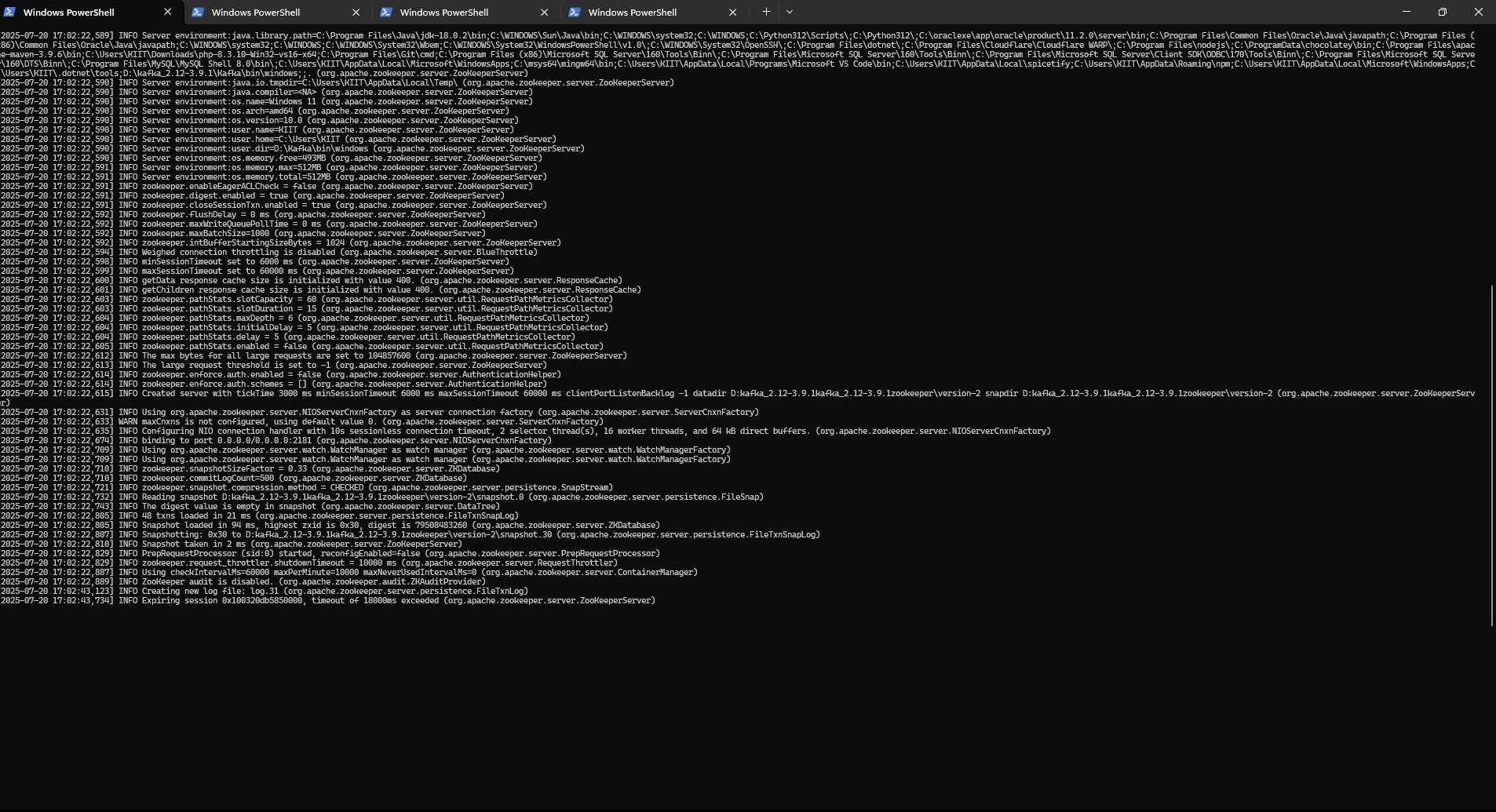
}

**OUTPUTS:**

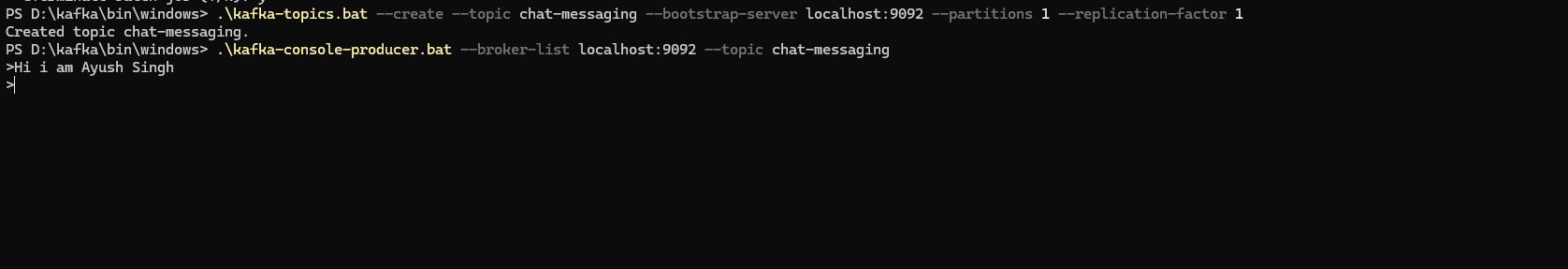
**Kafka Running:**

****

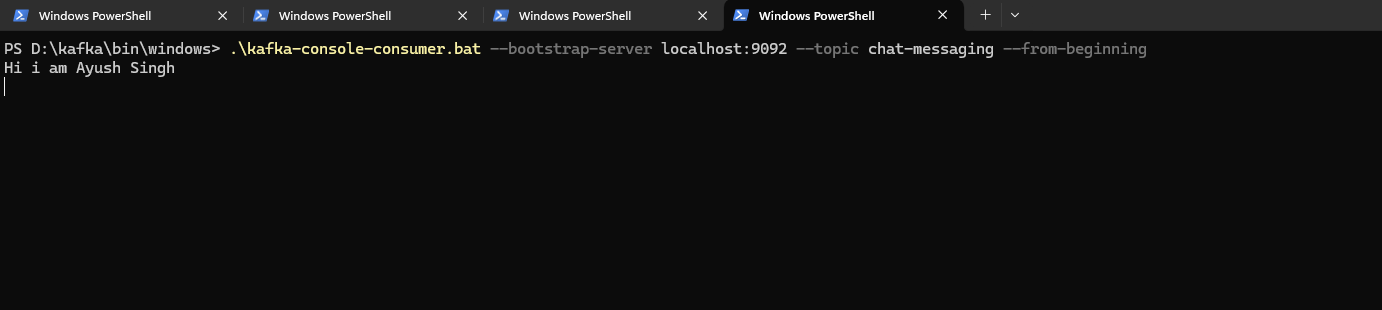
**Zookeeper Running:**



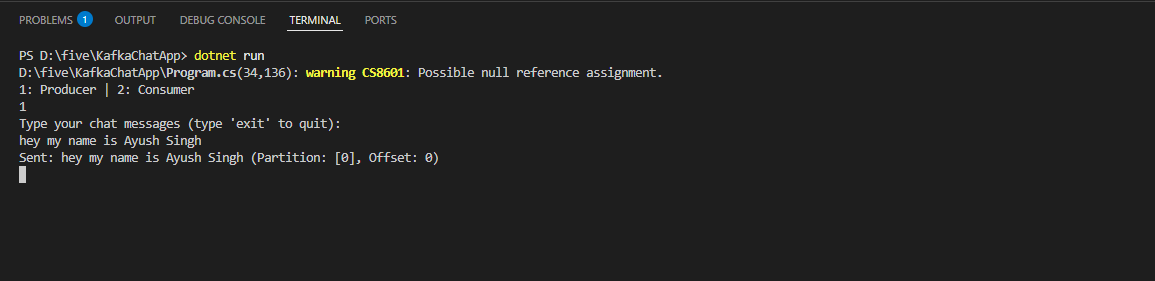
**Topic Creation and Creating Publisher in Command Prompt:**



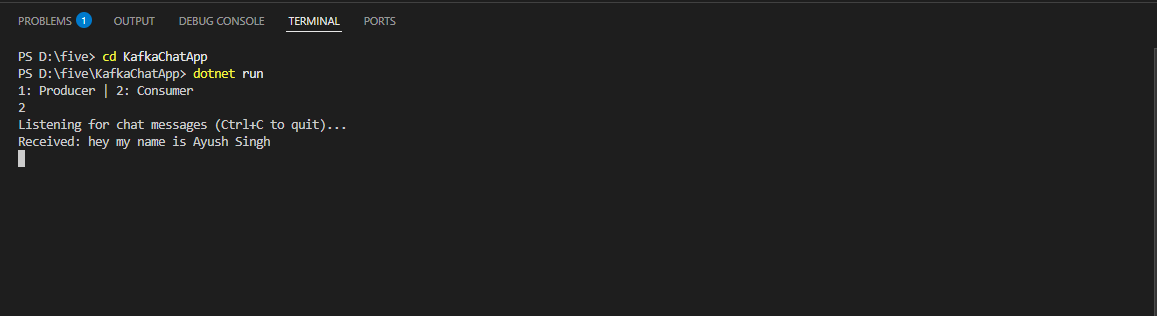
**Client Application (Consumer):**



**Topic Creation and Creating Publisher in Command Prompt(VS CODE):**



**Client Application (Consumer VS CODE):**

****

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

**CODE:**

**Program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

    .AddJwtBearer(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();

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**MODELS:**

**LoginModel.cs:**

namespace JwtAuthDemo.Models

{

    public class LoginModel

    {

        public string Username { get; set; }

        public string Password { get; set; }

    }

}

**CONTROLLERS:**

**AuthController.cs:**

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 (IsValidUser(model))

            {

                var token = GenerateJwtToken(model.Username);

                return Ok(new { Token = token });

            }

            return Unauthorized();

        }

        private bool IsValidUser(LoginModel model)

        {

            // ✅ Dummy user check

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

        }

        private string GenerateJwtToken(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(60),

                signingCredentials: creds);

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

**SecureController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class SecureController : ControllerBase

    {

        [Authorize]

        [HttpGet]

        public IActionResult Get()

        {

            return Ok("This is a secure endpoint. You are authenticated!");

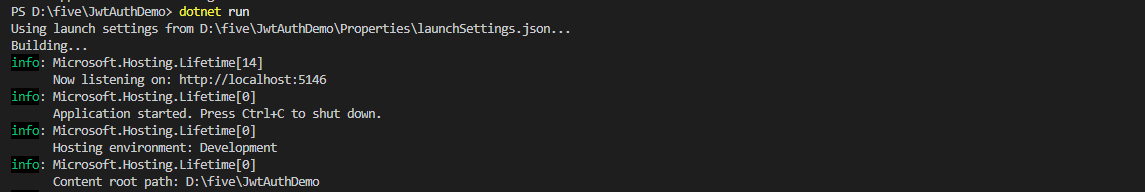
        }

    }

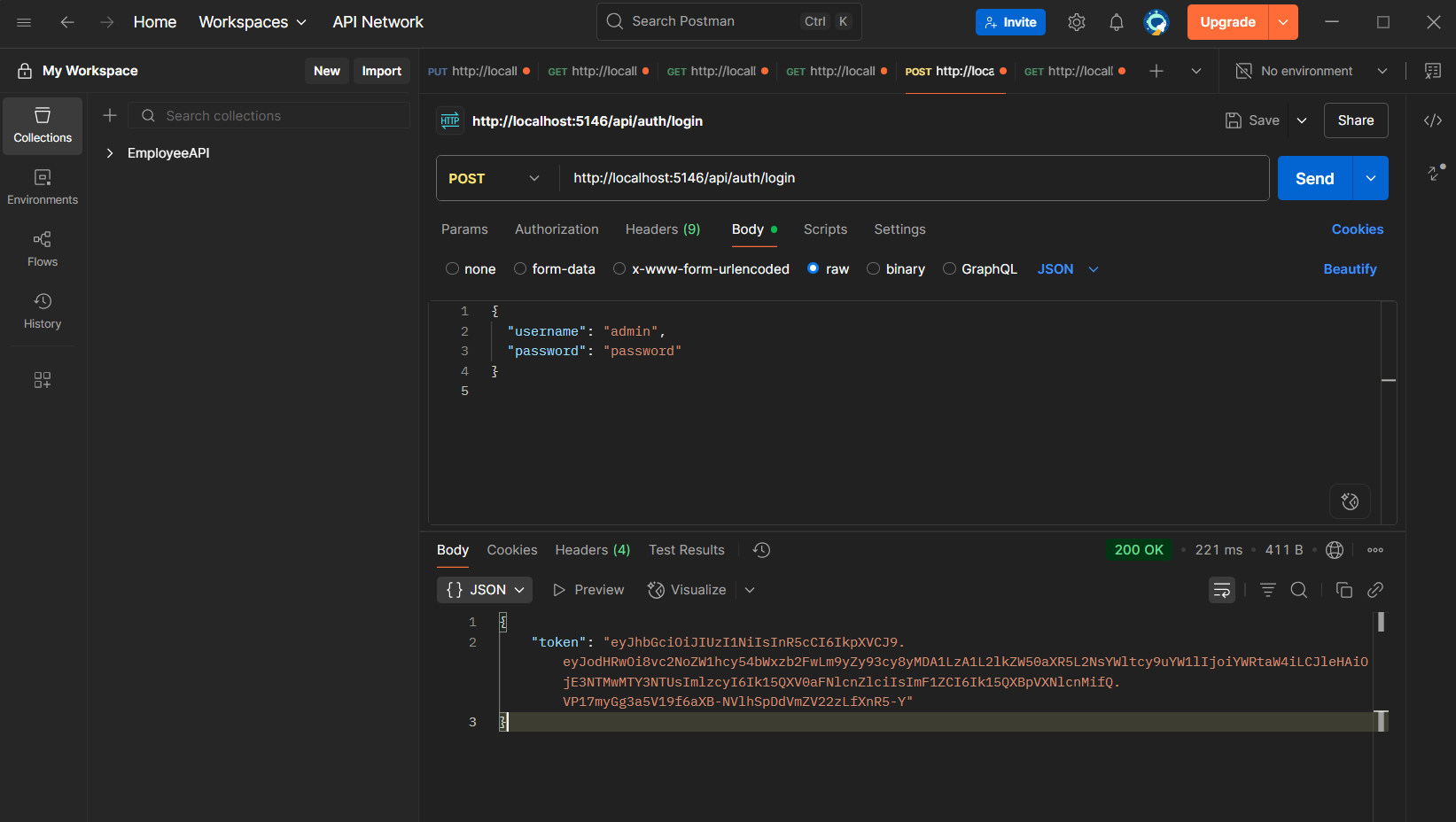
}

**OUTPUTS:**

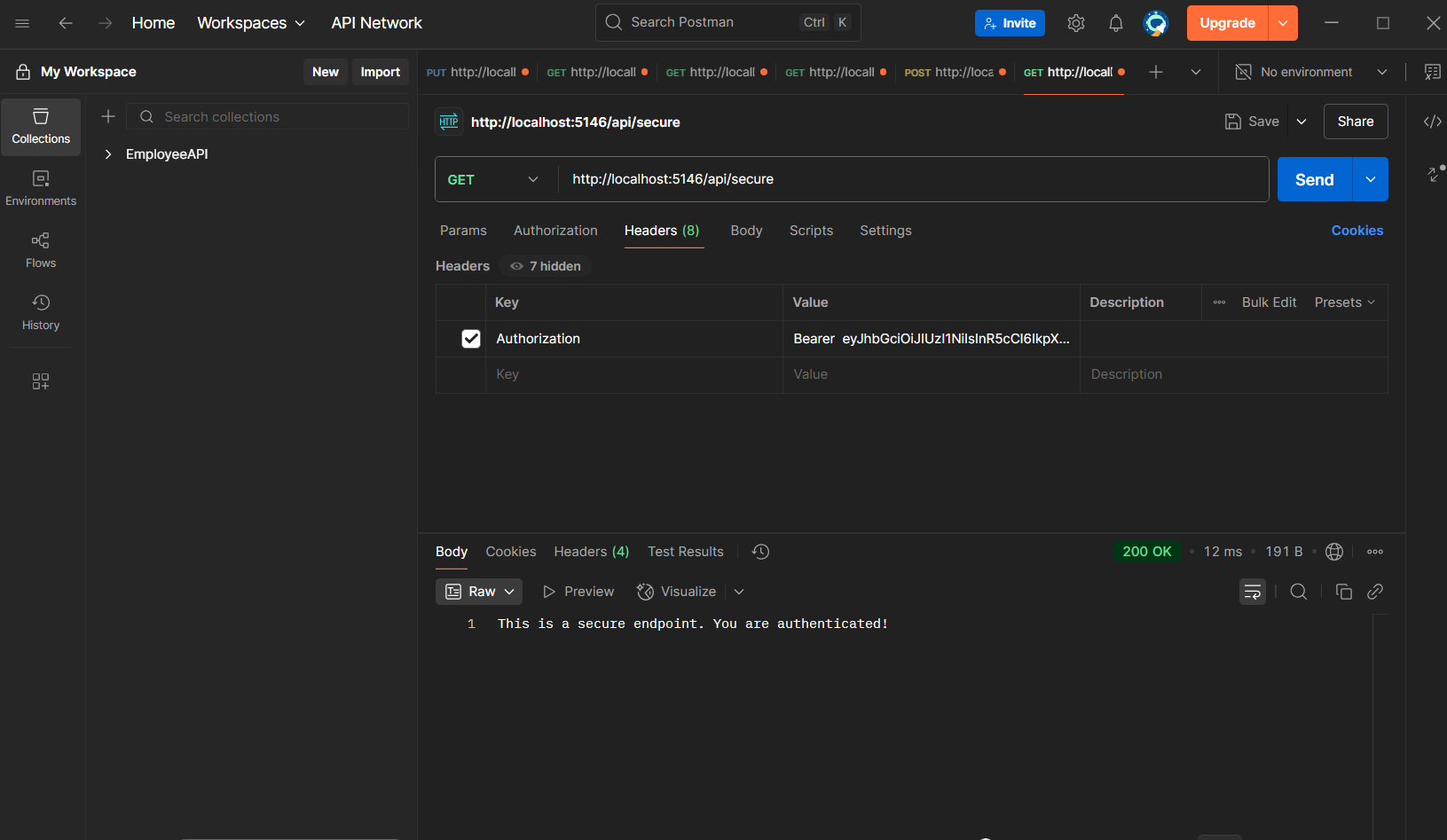
**BUILDING:**



**POST(Bearer Token Generation):**

****

**GET:**

****