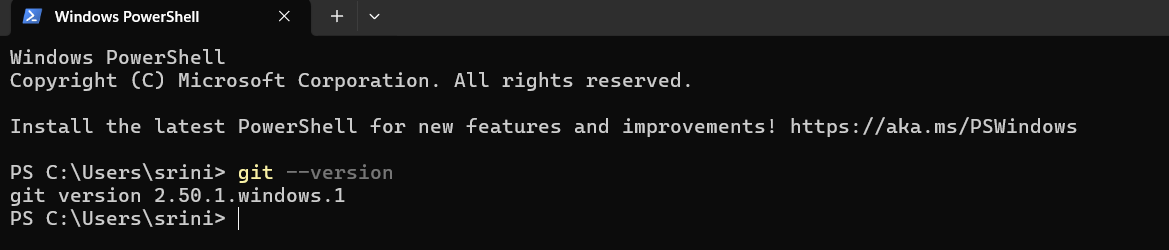
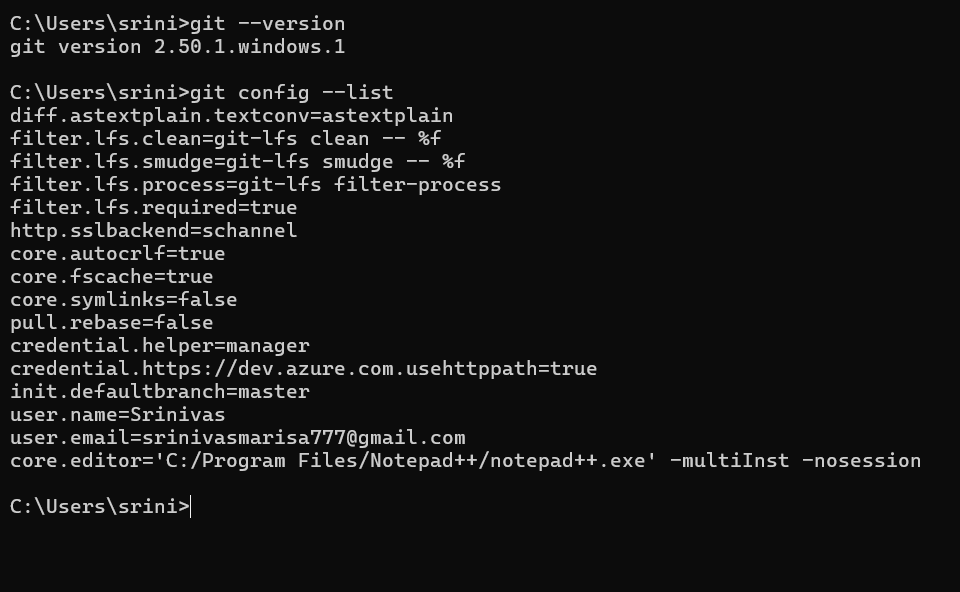
**Solution 1:**

**Git is installed**



I already configured git 3 years ago

**Verify the configuration**

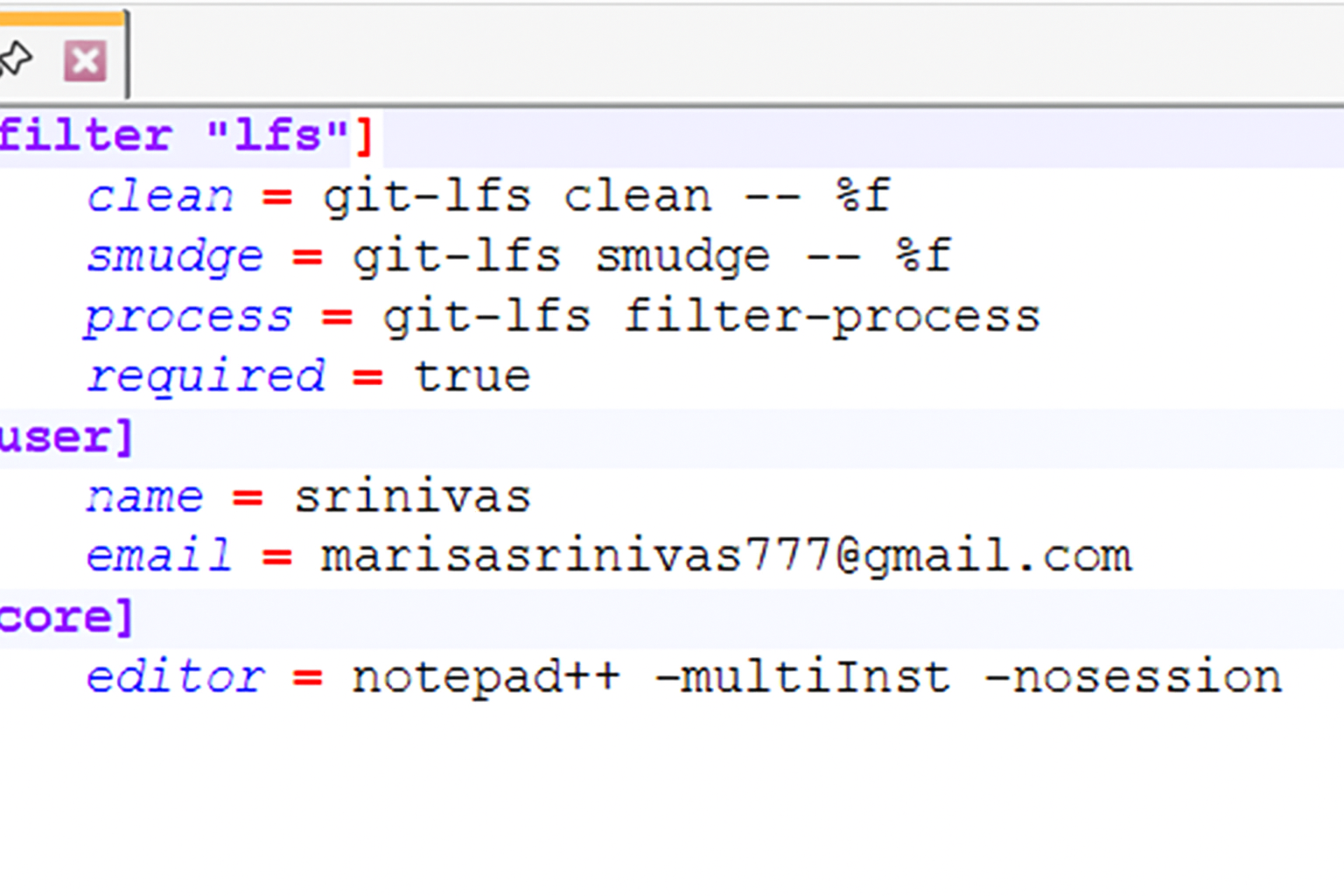


**Configure Notepad++ as Default Git Editor**

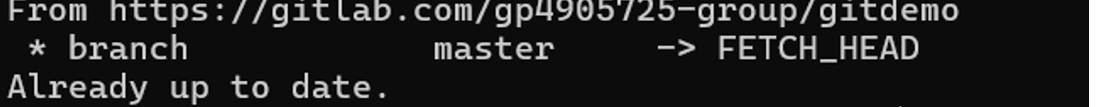
git config --global core.editor "notepad++ -multiInst -nosession"

**Verify the editor setting**

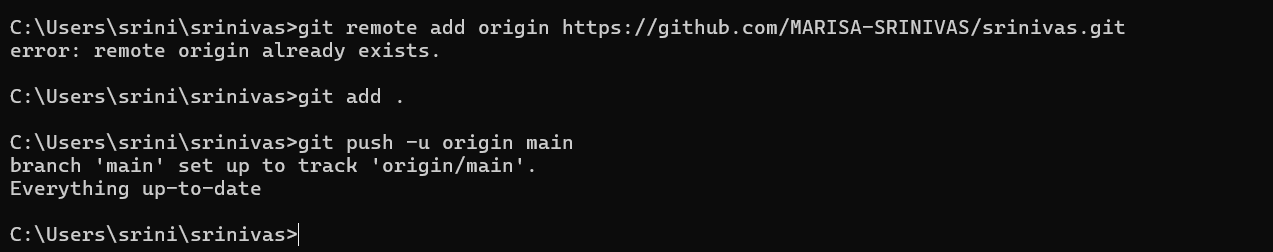
git config --global -e



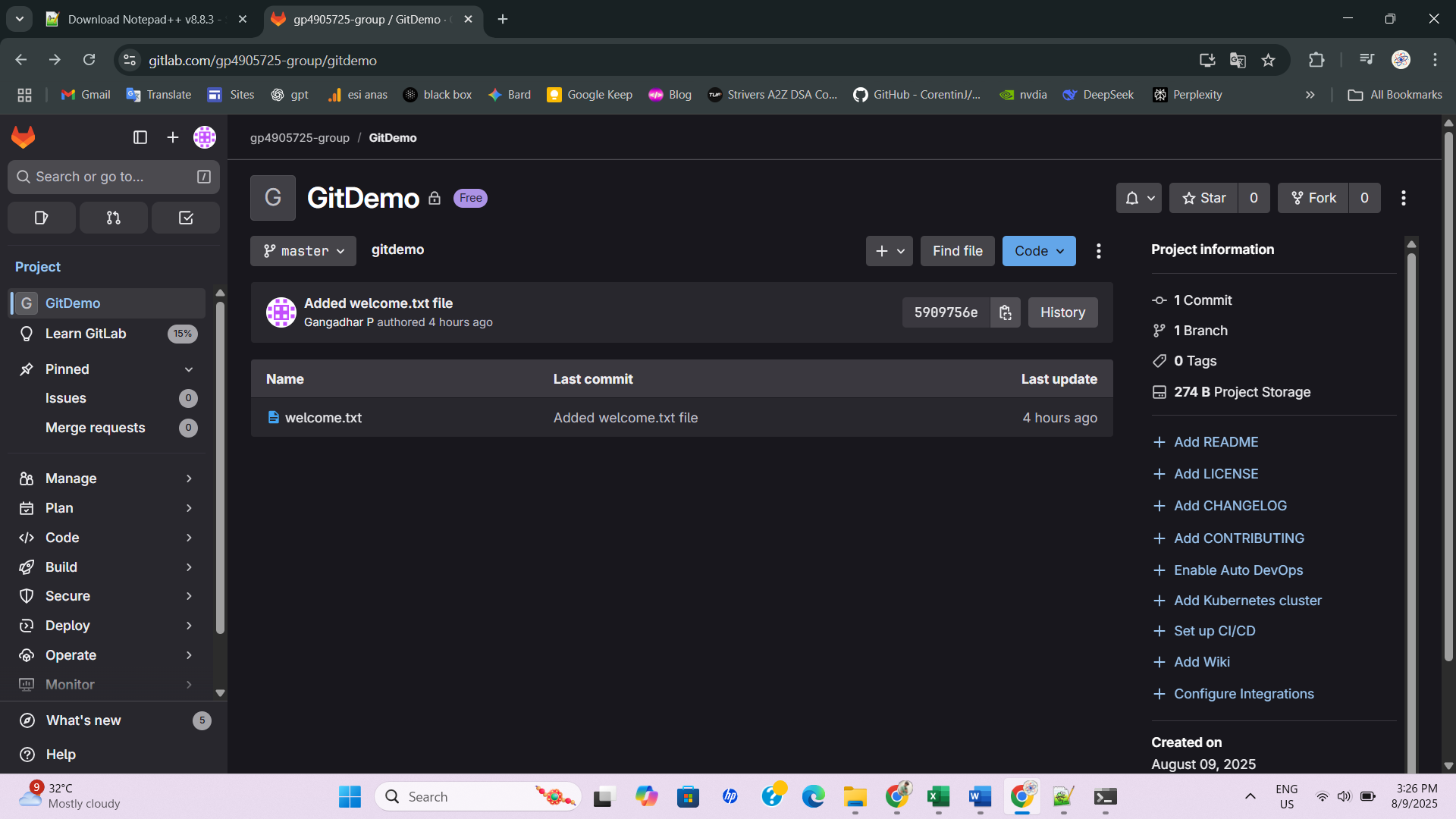
I pull the repo from gitlab to local system using the command “**git pull origin master”**



I push the files from the local system to Gitlab repos “ **git push origin master”**

****

**Result:**



**Solution 2:**

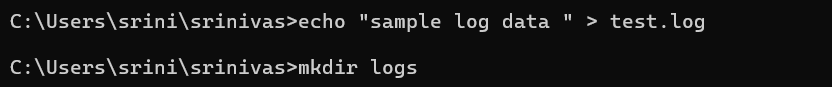
**In this hands-on lab, you will learn how to:**

**Implement git ignore command to ignore unwanted files and folders**

I added the 1 file and 1 directory, these are test.log and log/ in the above first solution gitdemo repository. Im using this commands in CLI

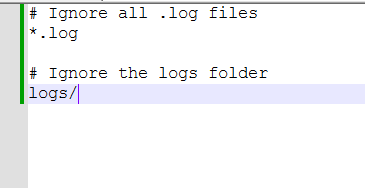
**echo "Sample log data" > test.log**

**mkdir logs**

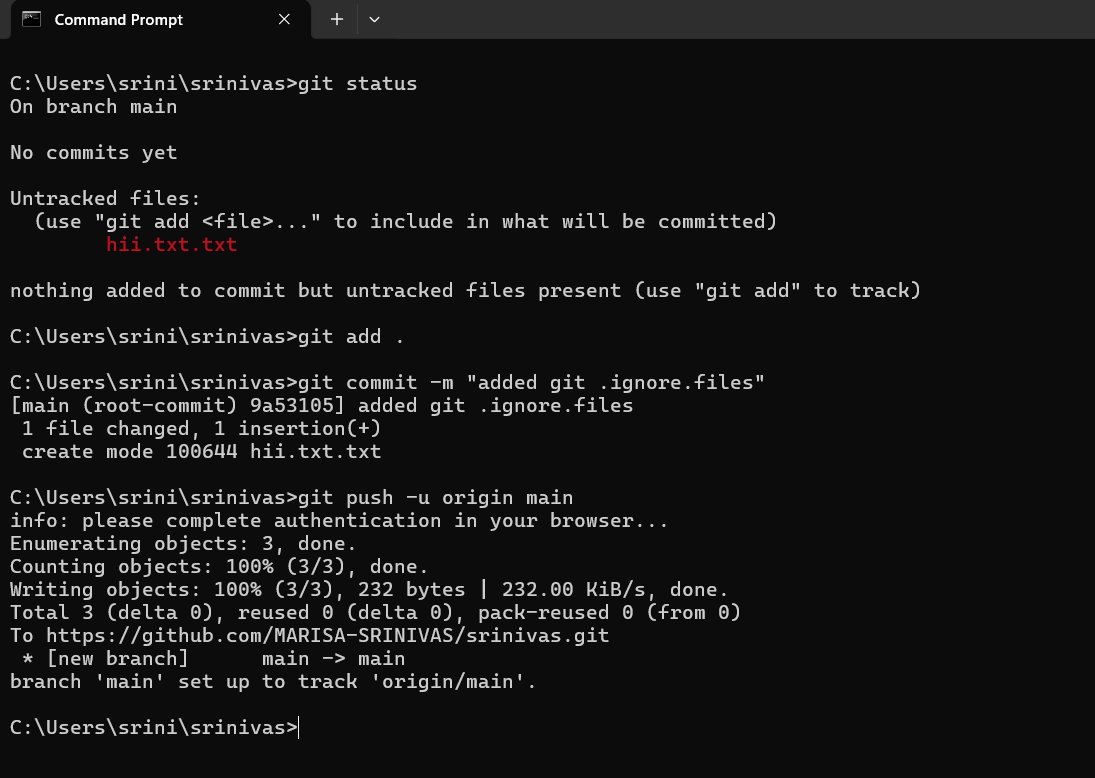


After than I create the .gitignore file inside the gitdemo repository using th notepad++

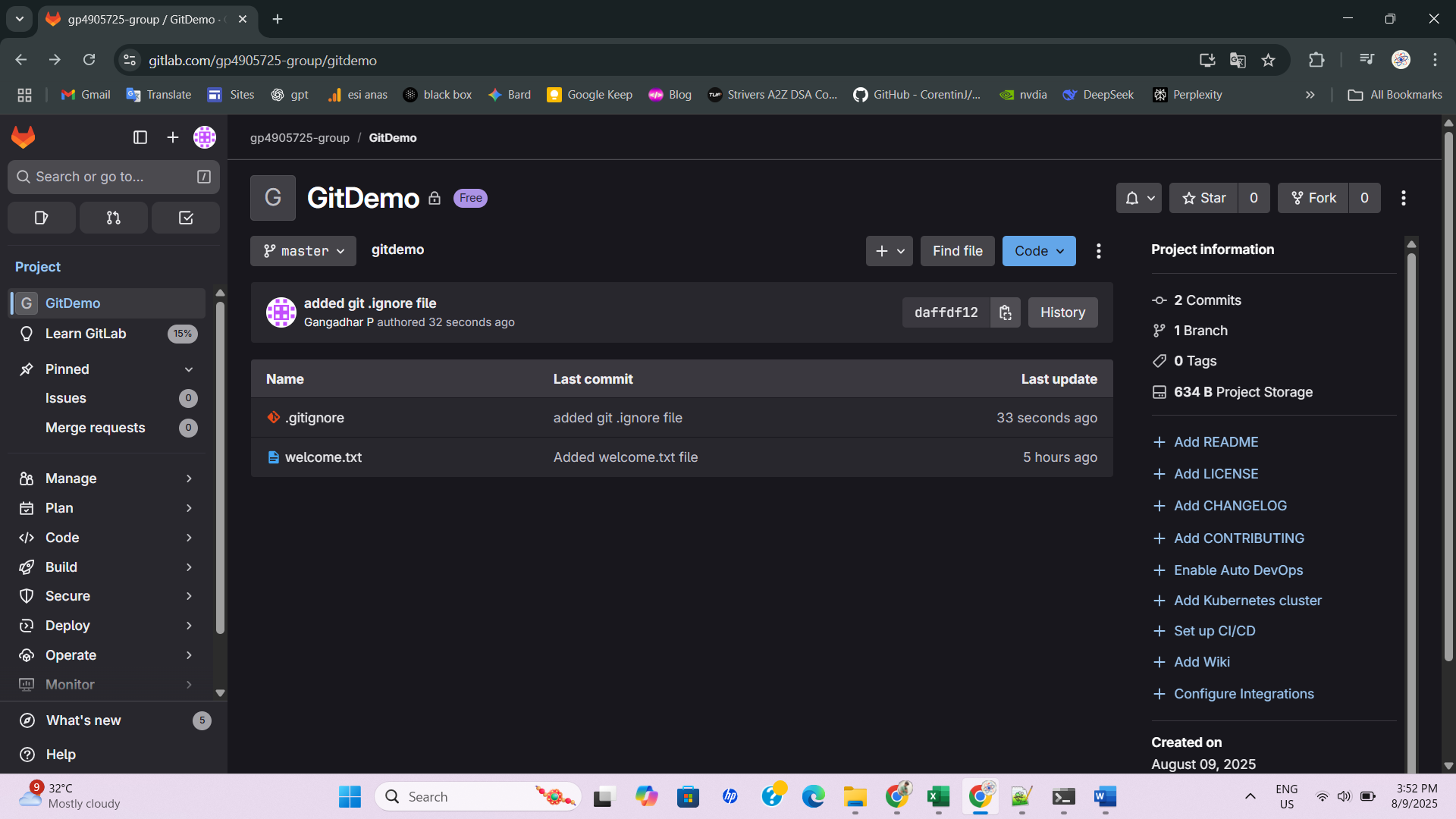
The command i used for this “**notepad++ .gitignore”**

****

It ignores the all the .log extension files and whole log dir while im push the code into gitlab or github



**Result**:



**Solution 3:**

I created the new branch using this command “**git branch GitNewBranch**”

And I list all branches using the command **“git branch -a”**

I switch the new branch from the current master branch using the command **“Switch to the new branch”**

And i added the files in the newbranch

echo "This is a test file for GitNewBranch" > branchfile.txt

git add branchfile.txt

git commit -m "Added branchfile.txt in GitNewBranch"

After that i checked the status of the git using  **“git status”**

**Merging the branches:**

Now Im switch the new branch to master branch using the command “**git checkout master**”

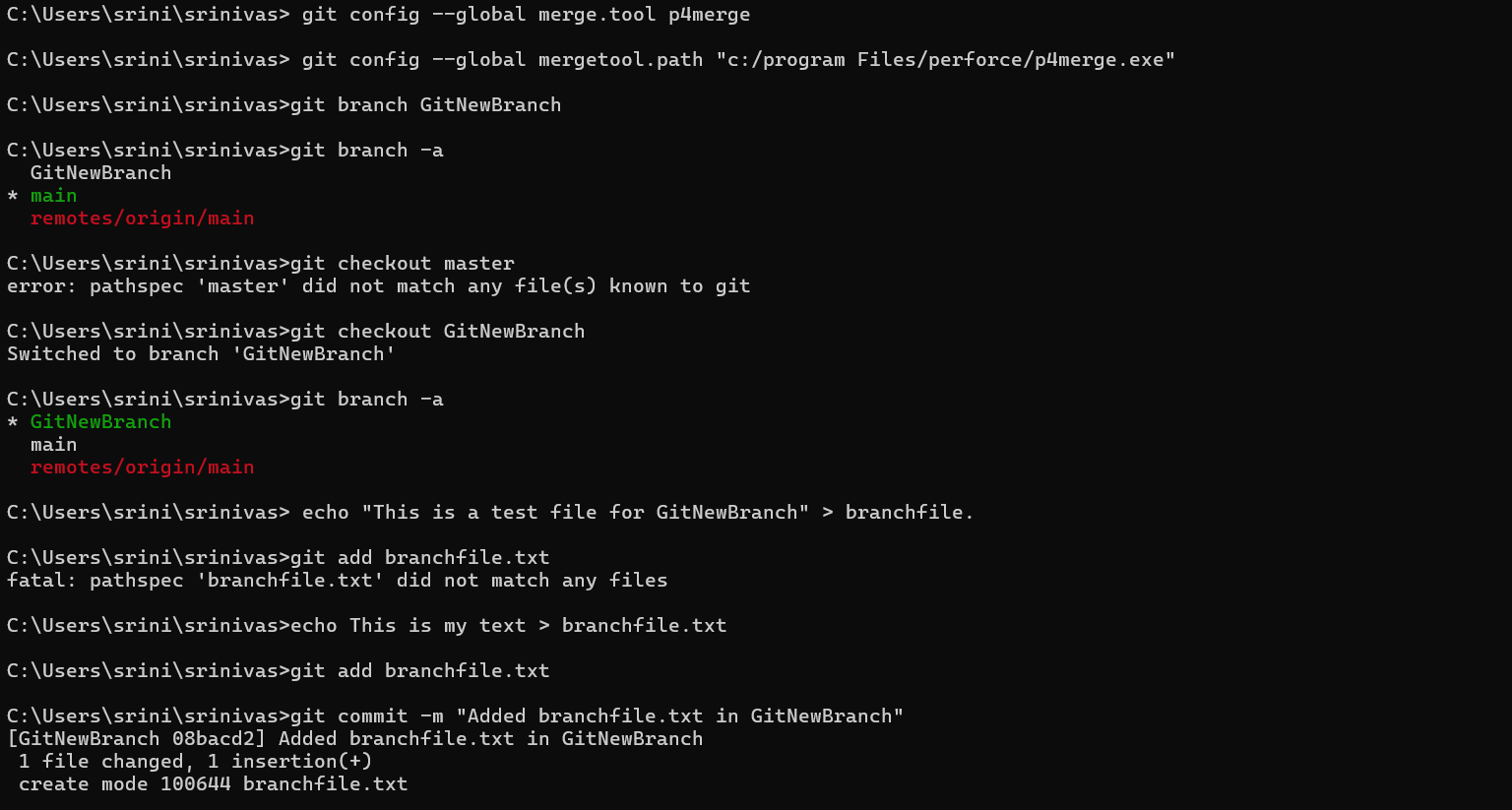
I show command-line differences between master and GitNewBranch using command “**git diff master GitNewBranch”**

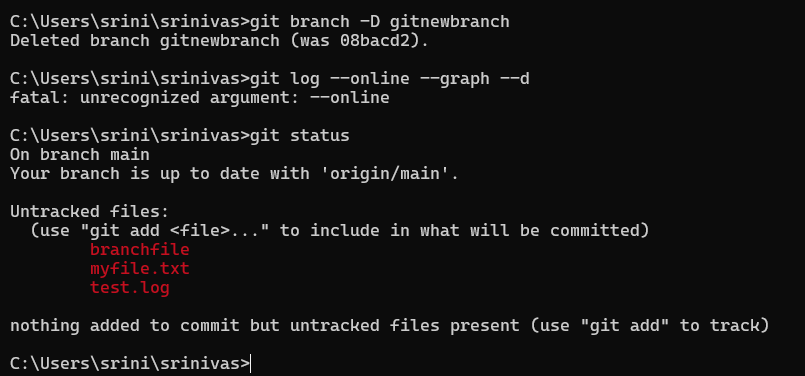
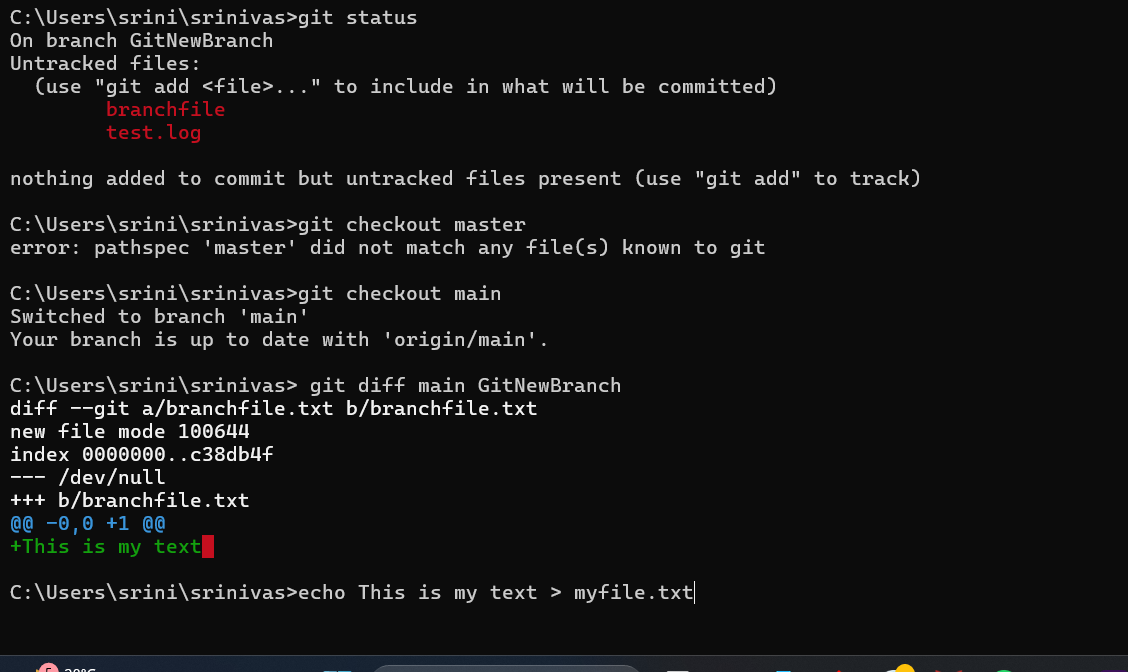
After i merge branch into master using the command “**git merge GitNewBranch”**

When i want to see the merge history using the command **“git log --oneline --graph --decorate”**

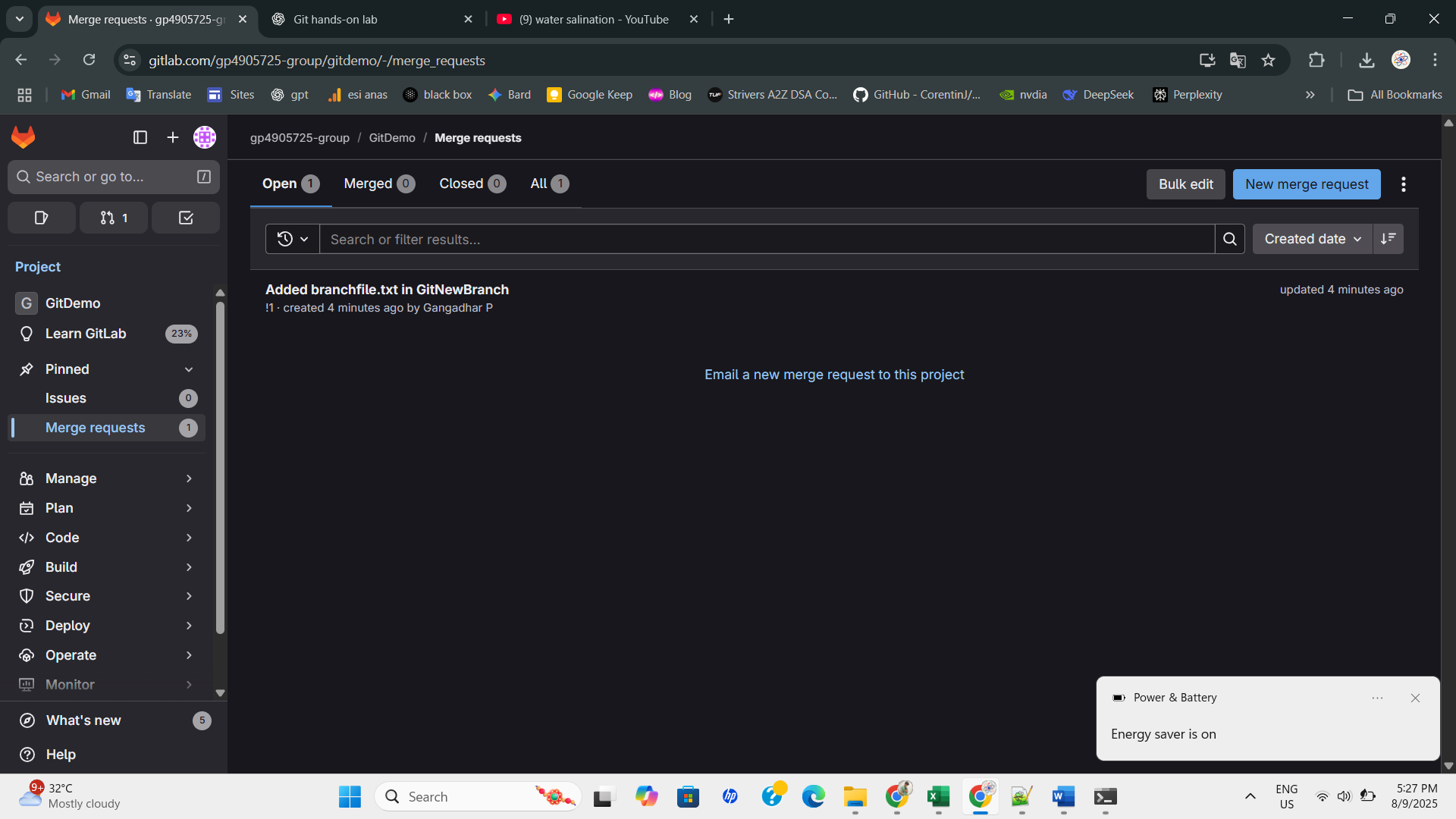
Delete the branch after merge using the command “**git branch -d GitNewBranch**”

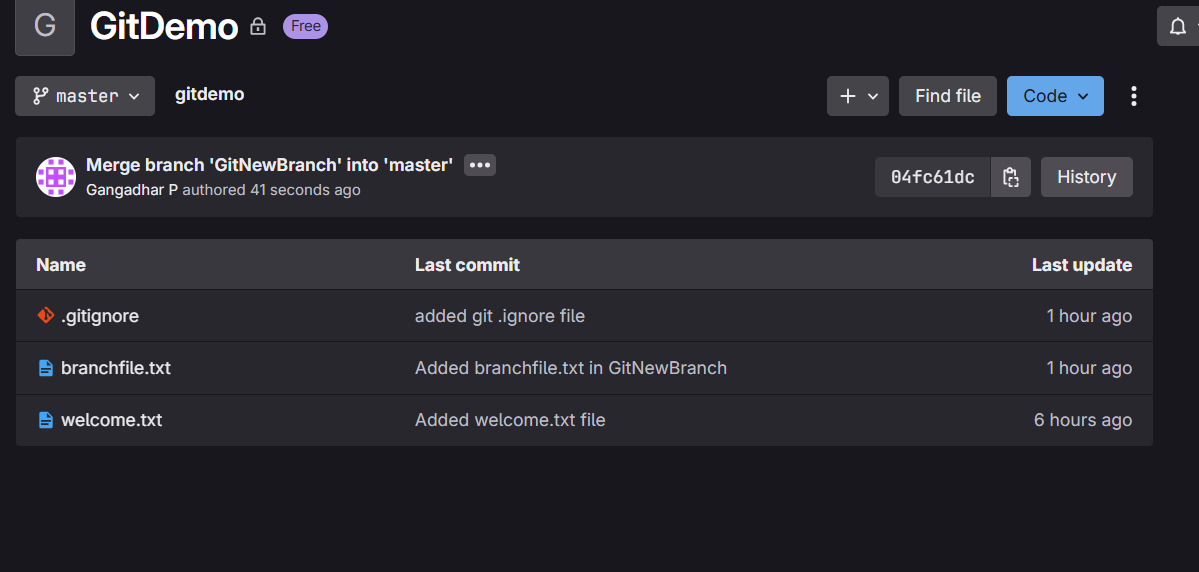
Now verify the status of the git using the command “**git status”**





I created the merge request and merged successfully





**Solution 4:**

**Validate JWT Token Expiry and Handle Unauthorized Access Scenario:**

**You want to handle expired or invalid tokens gracefully.**

**Steps: 1. Configure JWT bearer events.**

**2. Return custom messages for unauthorized access.**

**//Program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.IdentityModel.Tokens.Jwt;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddAuthentication(options =>

{

    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

    options.DefaultChallengeScheme = 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"]!))

    };

    options.Events = new JwtBearerEvents

    {

        OnAuthenticationFailed = async context =>

        {

            context.Response.StatusCode = 401;

            context.Response.ContentType = "application/json";

            var message = context.Exception is SecurityTokenExpiredException

                ? "{\"error\": \"Token has expired.\"}"

                : "{\"error\": \"Invalid token.\"}";

            await context.Response.WriteAsync(message);

        }

    };

});

builder.Services.AddAuthorization();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

    options.SwaggerDoc("v1", new OpenApiInfo { Title = "JwtAuthDemo", Version = "v1" });

    options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        Name = "Authorization",

        Type = SecuritySchemeType.ApiKey,

        Scheme = "Bearer",

        BearerFormat = "JWT",

        In = ParameterLocation.Header,

        Description = "Enter 'Bearer' followed by your token. Example: `Bearer abc123...`"

    });

    options.AddSecurityRequirement(new OpenApiSecurityRequirement

    {

        {

            new OpenApiSecurityScheme

            {

                Reference = new OpenApiReference

                {

                    Type = ReferenceType.SecurityScheme,

                    Id = "Bearer"

                }

            },

            Array.Empty<string>()

        }

    });

});

vår app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

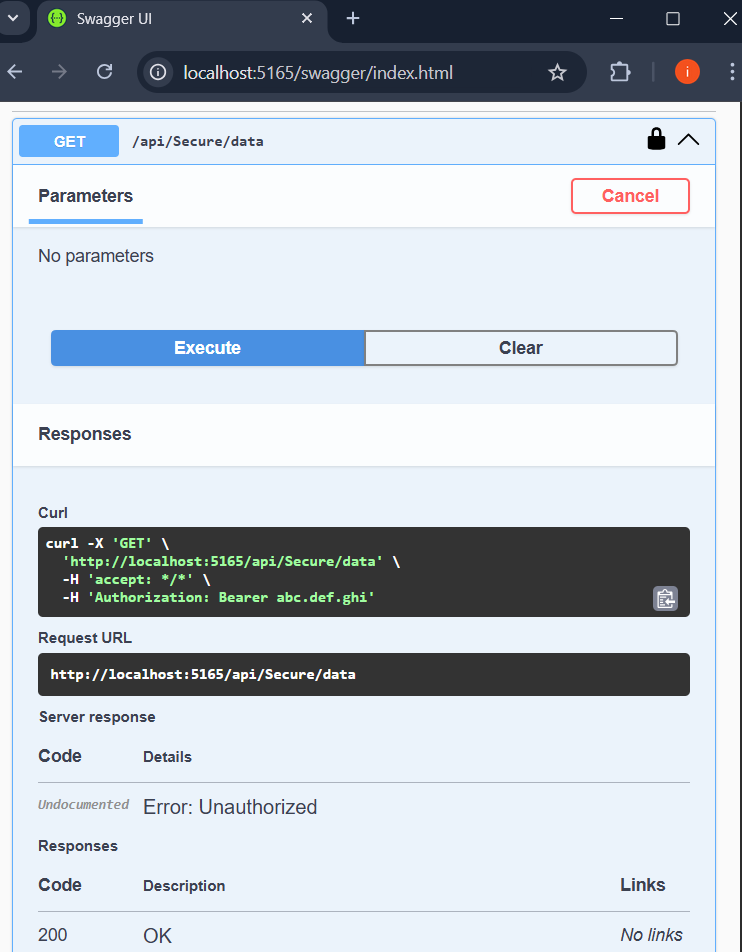
app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

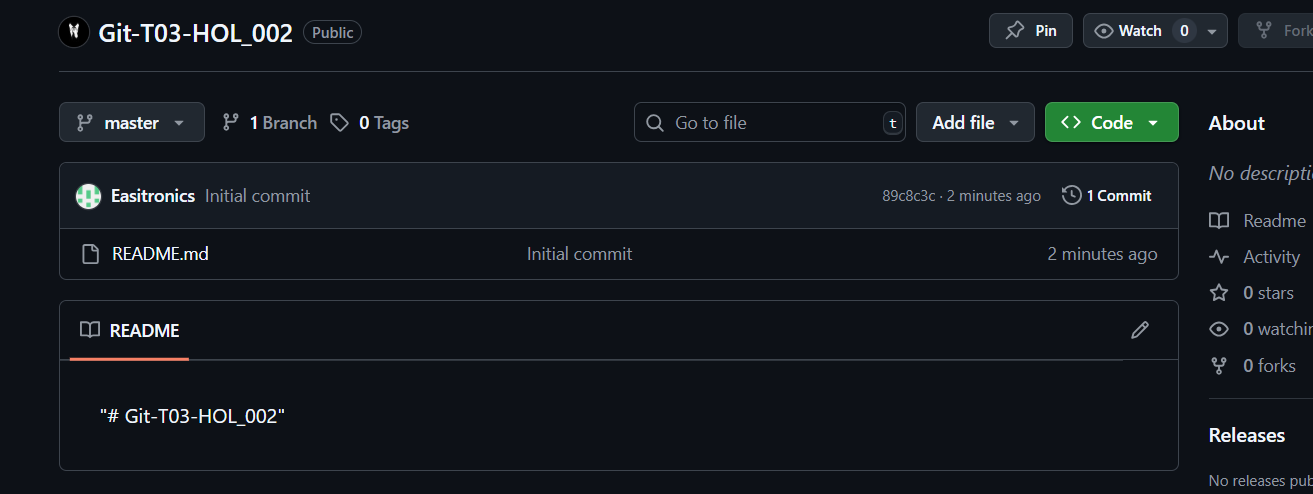
app.Run();

**Output:**

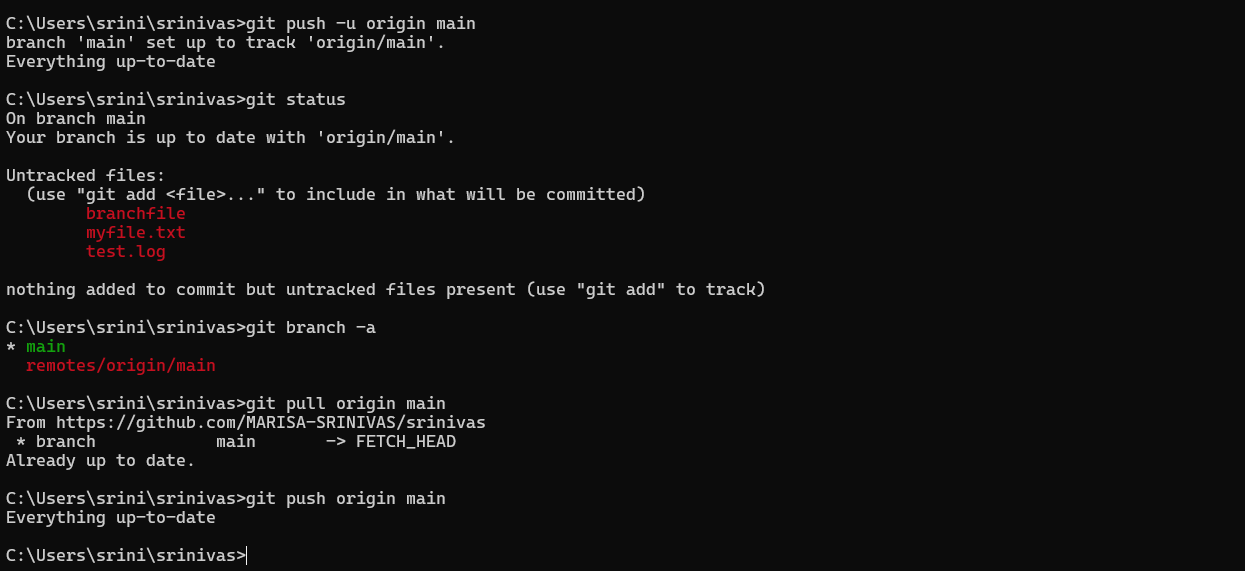


**Solution 5:**

1. Verify if master is in clean state.
2. List out all the available branches.
3. Pull the remote git repository to the master
4. Push the changes, which are pending from **“Git-T03-HOL\_002”** to the remote repository.
5. Observe if the changes are reflected in the remote repository.

**+**

**OUTPUT:**

****