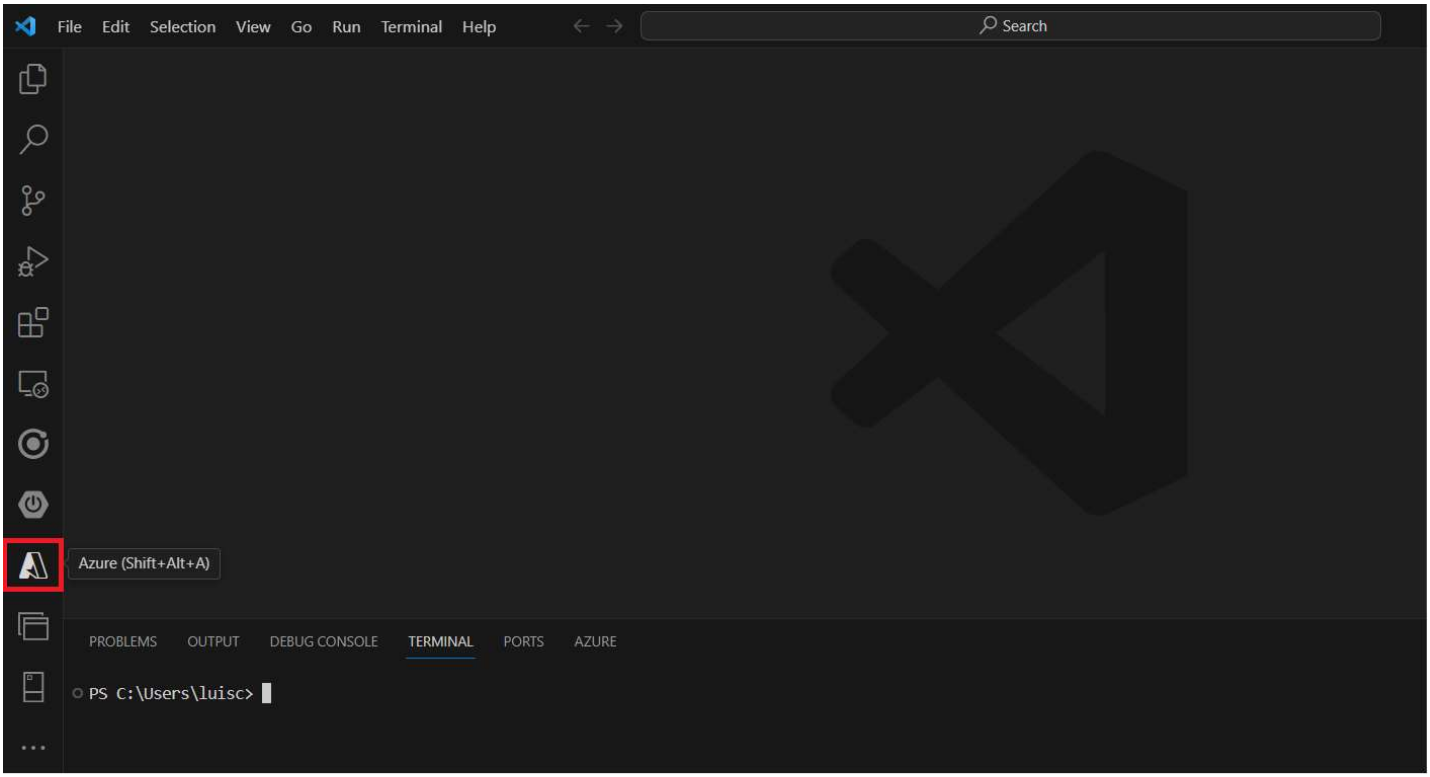
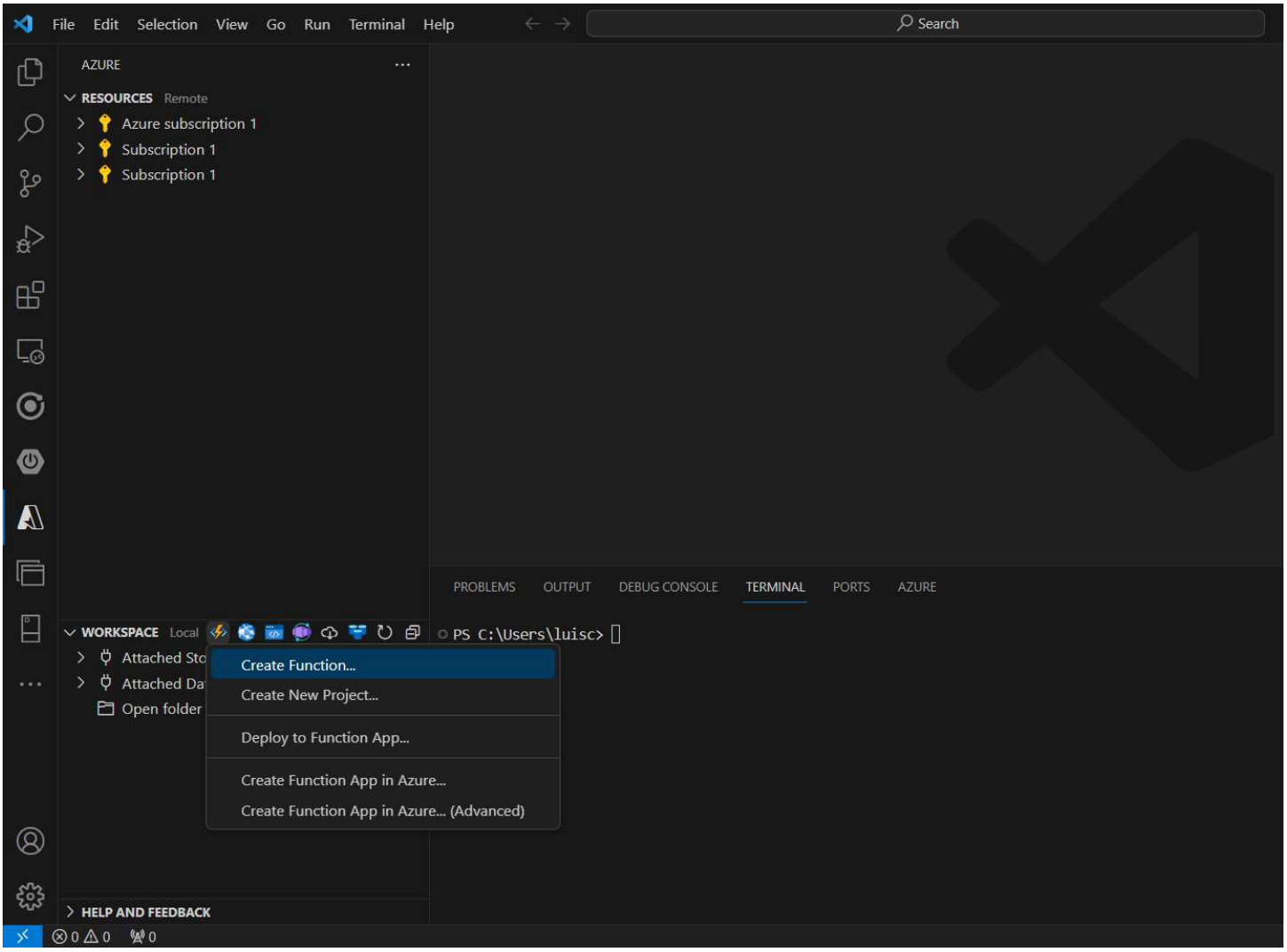
Azure SDK for .NET: How to create Azure ResourceGroup from Azure Function

1. Create the Azure Function in VSCode with Azure SDK for .NET

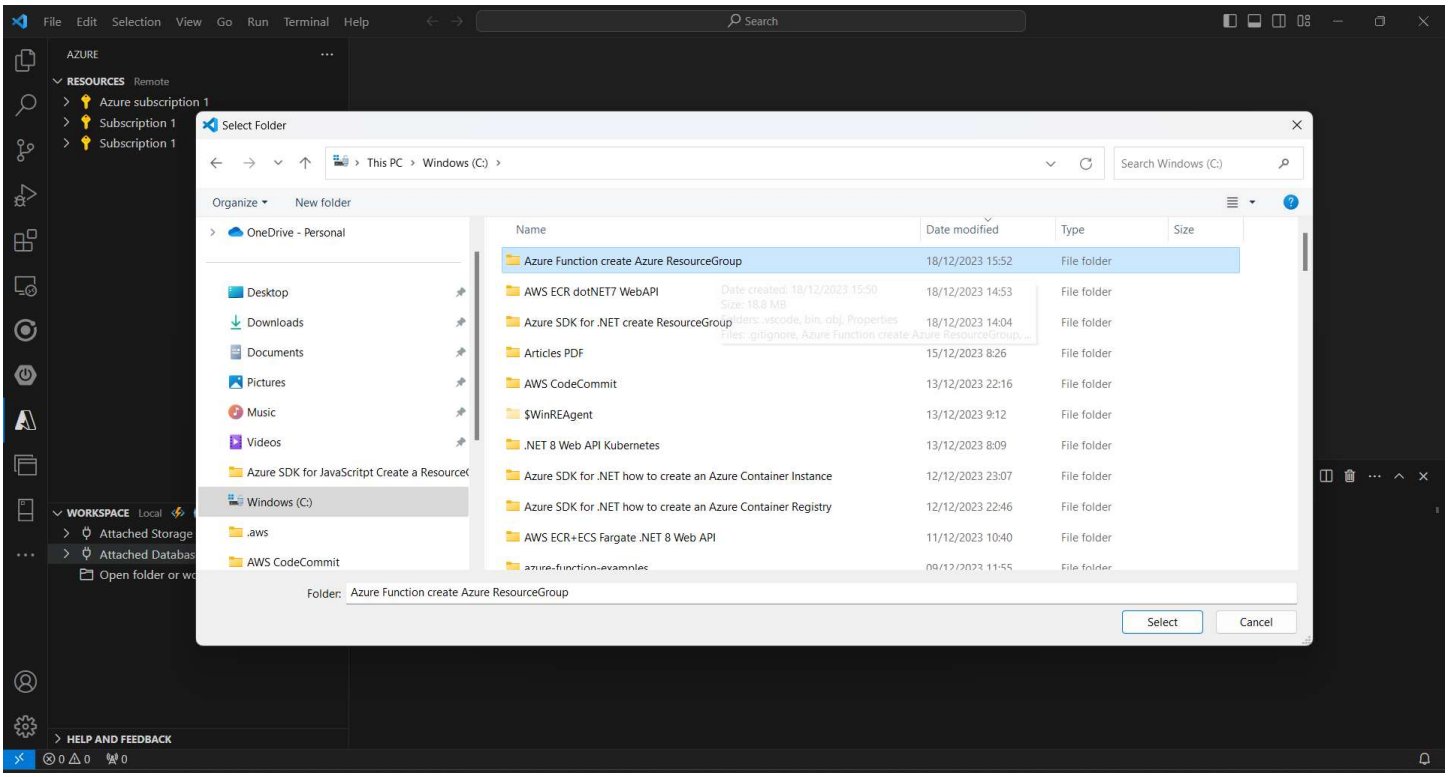
Run VSCode and create a new Azure Function



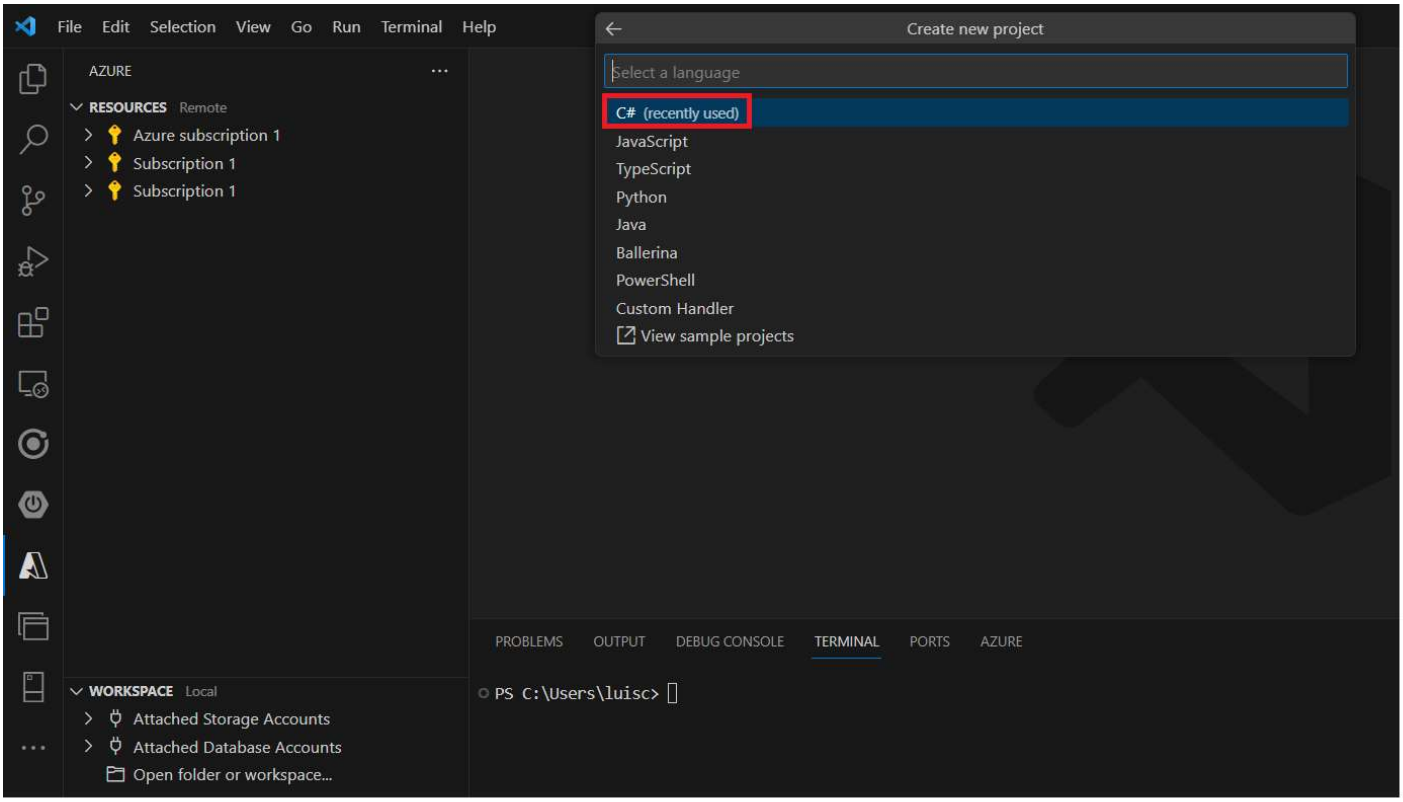
We select **Create Function...**



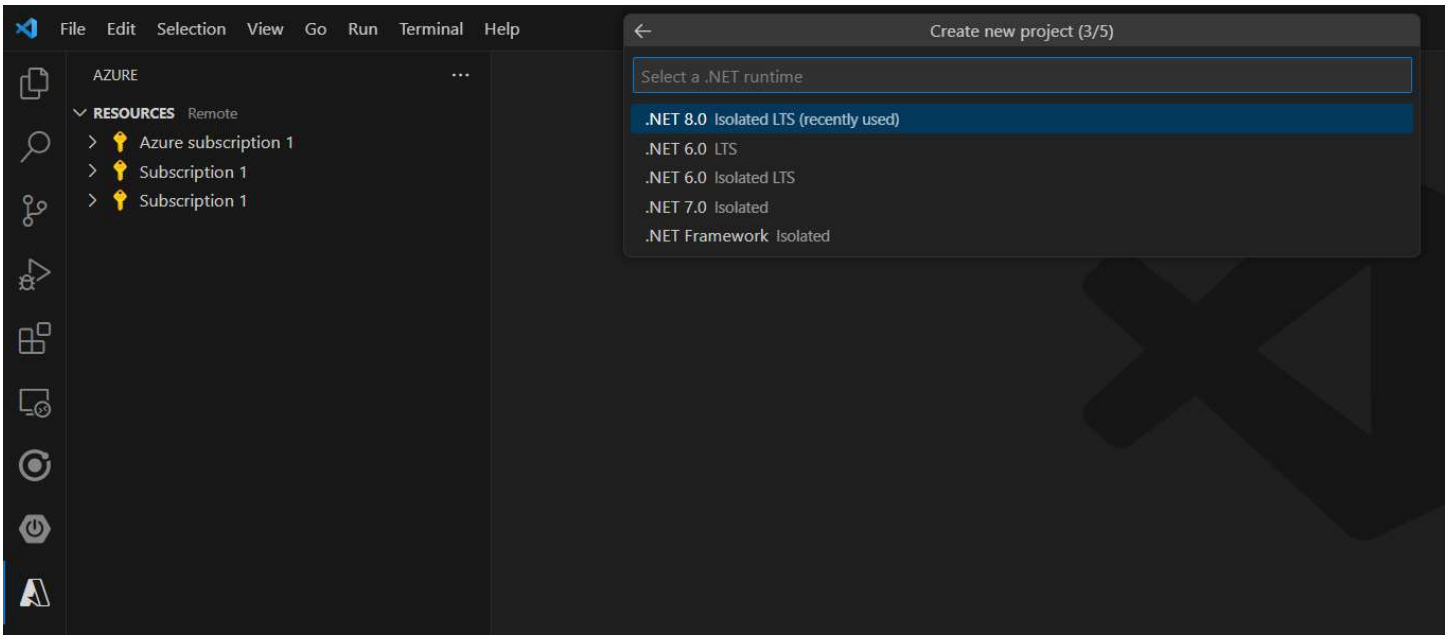
We select the folder to place the new Azure function



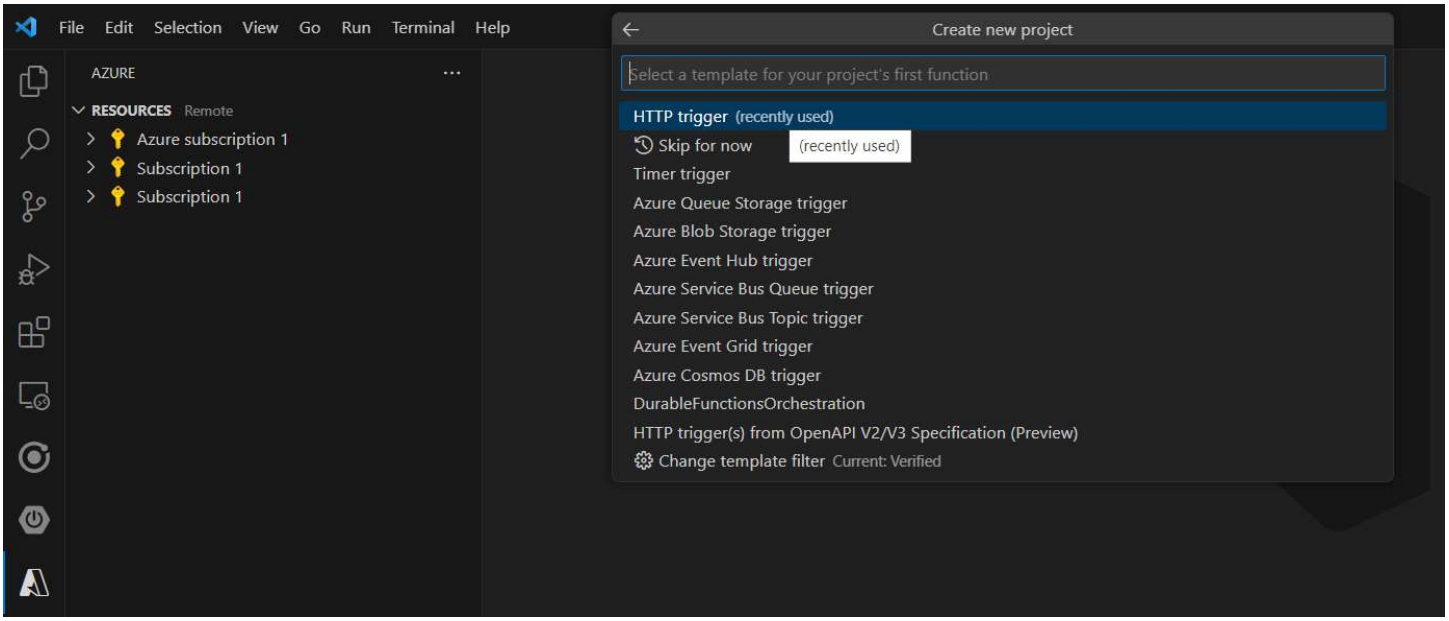
Select the C# language



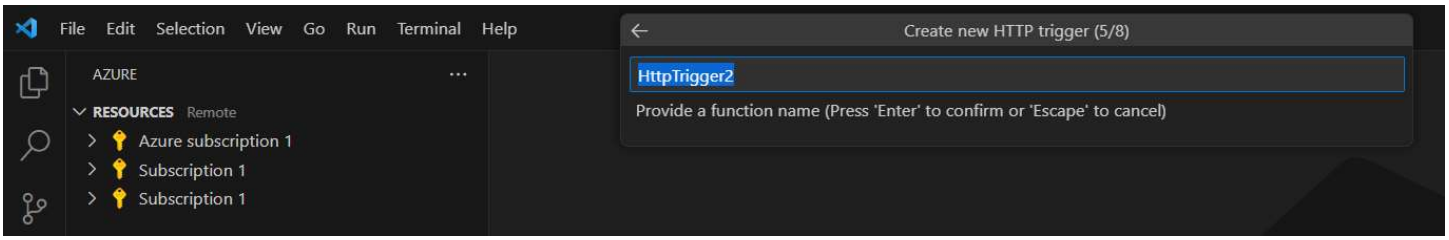
Select the .NET runtime



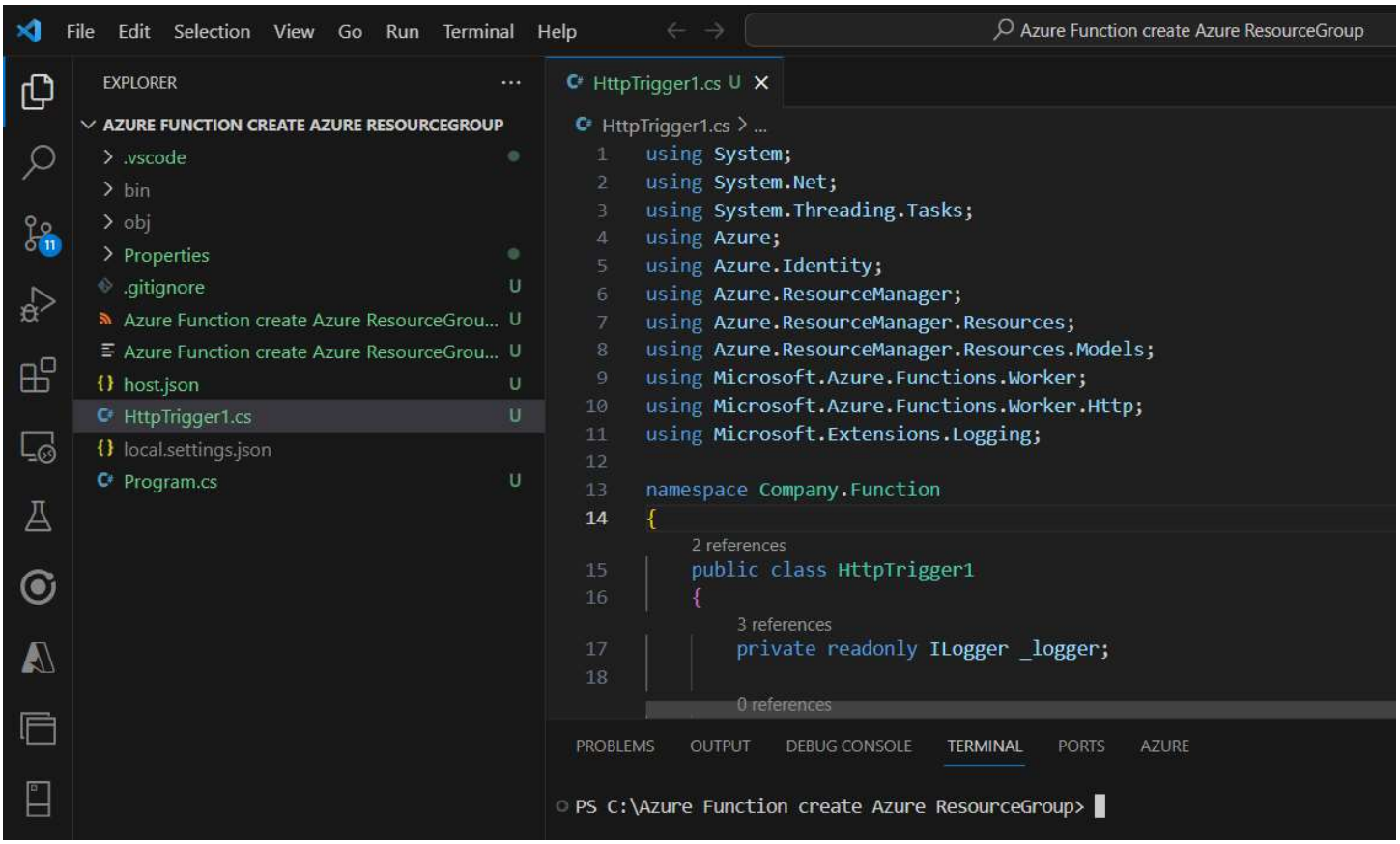
We select the Http Triggered Azure Function



Provide the Function name



This is the project folders structure



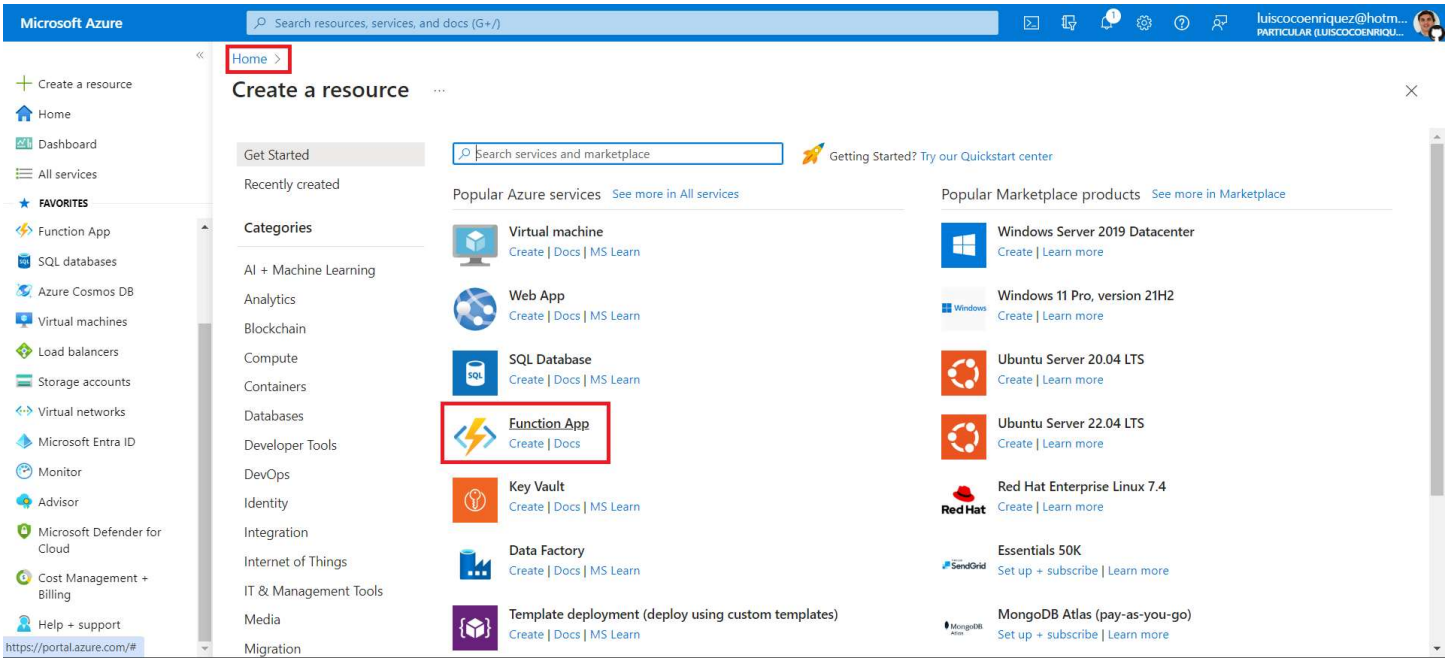
Now we input the C# source code:

```
using System;
using System.Net;
using System.Threading.Tasks;
using Azure;
using Azure.Identity;
using Azure.ResourceManager;
using Azure.ResourceManager.Resources;
using Azure.ResourceManager.Resources.Models;
using Microsoft.Azure.Functions.Worker;
using Microsoft.Azure.Functions.Worker.Http;
using Microsoft.Extensions.Logging;

namespace Company.Function
{
    public class HttpTrigger1
    {
        private readonly ILogger _logger;

        public HttpTrigger1(ILoggerFactory loggerFactory)
        {
            _logger = loggerFactory.CreateLogger<HttpTrigger1>();
        }

        [Function("HttpTrigger1")]
        public async Task<HttpResponseData> Run([HttpTrigger(AuthorizationLevel.Function, "get
        {
            _logger.LogInformation("C# HTTP trigger function processed a request.");

            var query = System.Web.HttpUtility.ParseQueryString(req.Url.Query);
            string resourceGroupName = query["resourceGroupName"];

            if (string.IsNullOrEmpty(resourceGroupName))
            {
                var badRequestResponse = req.CreateResponse(HttpStatusCode.BadRequest);
                await badRequestResponse.WriteStringAsync("Please pass a resourceGroupName on
                return badRequestResponse;
            }

            try
            {
                string subscriptionId = Environment.GetEnvironmentVariable("AZURE_SUBSCRIPTION
                var credential = new DefaultAzureCredential();
                var armClient = new ArmClient(credential, subscriptionId);

                string location = "westeurope"; // You can also make this a parameter
                var resourceGroupData = new ResourceGroupData(location);

                var operation = await armClient.GetDefaultSubscription().GetResourceGroups().C
                var resourceGroup = operation.Value;

                var response = req.CreateResponse(HttpStatusCode.OK);
                await response.WriteStringAsync($"Resource group {resourceGroupName} created i
```

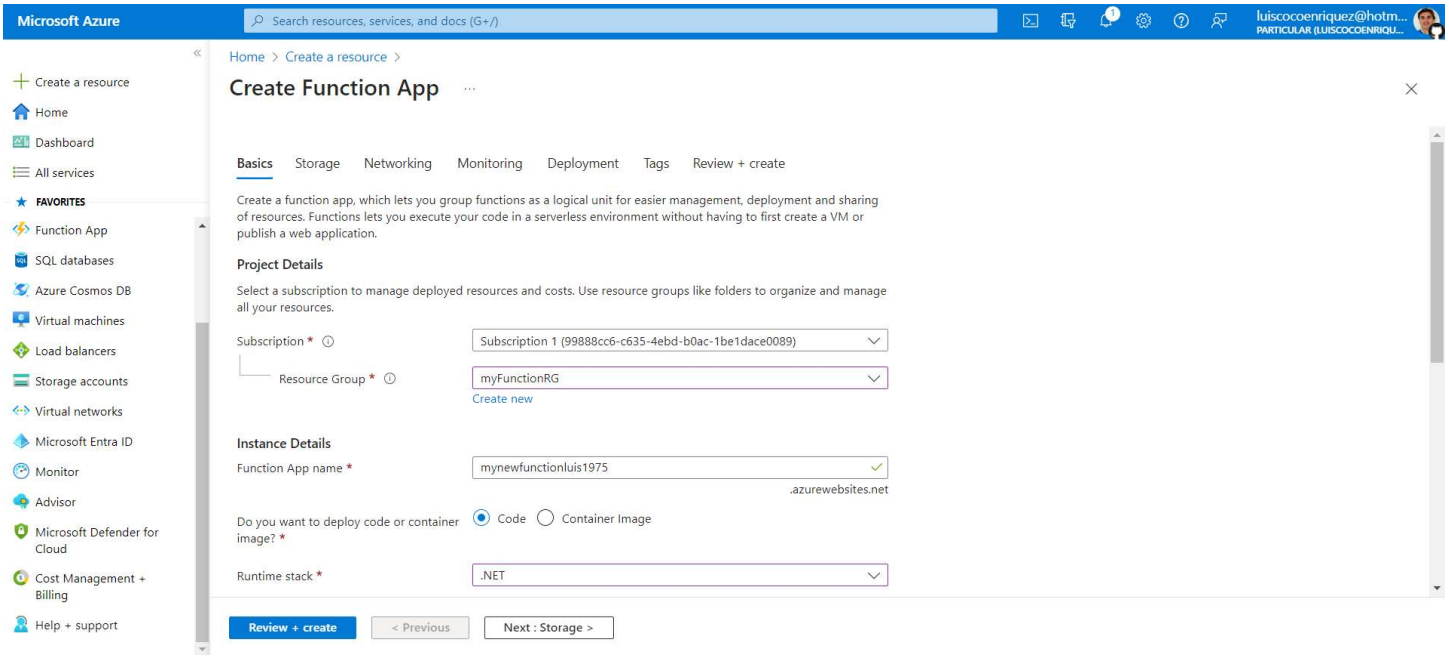
```
                return response;
            }
            catch (Exception ex)
            {
                _logger.LogError($"Error creating resource group: {ex.Message}");
                var errorResponse = req.CreateResponse(HttpStatusCode.InternalServerError);
                await errorResponse.WriteStringAsync("Error creating resource group");
                return errorResponse;
            }
        }
    }
}
```

2. In Azure Portal create a new Azure Function

In Azure Portal we navigate to the Functions App service



We input the new Function values: subscription, resourcegroup, region, function name, runtime stack, etc



3. In Azure Portal register a new application

Run the following Azure CLI commands:

Login to Azure:

```
az login
```

Create an Azure AD Application and Service Principal:

```
az ad app create --display-name "<your-app-name>"
```

Create a Secret for the Application:

```
az ad app credential reset --id <app-id> --append
```

Get Your Azure AD Tenant ID:

```
az account show --query tenantId -o tsv
```

Assign Required Permissions:

```
az role assignment create --assignee <app-id> ^
--role Contributor ^
--scope /subscriptions/<subscription-id>
```

4. Create the environmental variables in Azure Portal for the Azure Function

Go to the Azure Function in Azure Portal and in the Configuration menu option in the left menu create the following environmental variables:

Retrieve the Subscription (**AZURE_SUBSCRIPTION_ID**)

```
az account show --query id -o tsv
```

Retrieve the Client ID (**AZURE_CLIENT_ID**)

```
az ad app list --display-name "<your-app-name>" --query "[].appId" -o tsv
```

Retrieve Azure AD Tenant ID (**AZURE_TENANT_ID**)

```
az account show --query tenantId -o tsv
```

Retrieve the Secret (**AZURE_CLIENT_SECRET**)

```
az ad app credential reset --id <app-id> ^
--append ^
--credential-description "MyNewSecret" ^
--query password ^
-o tsv
```

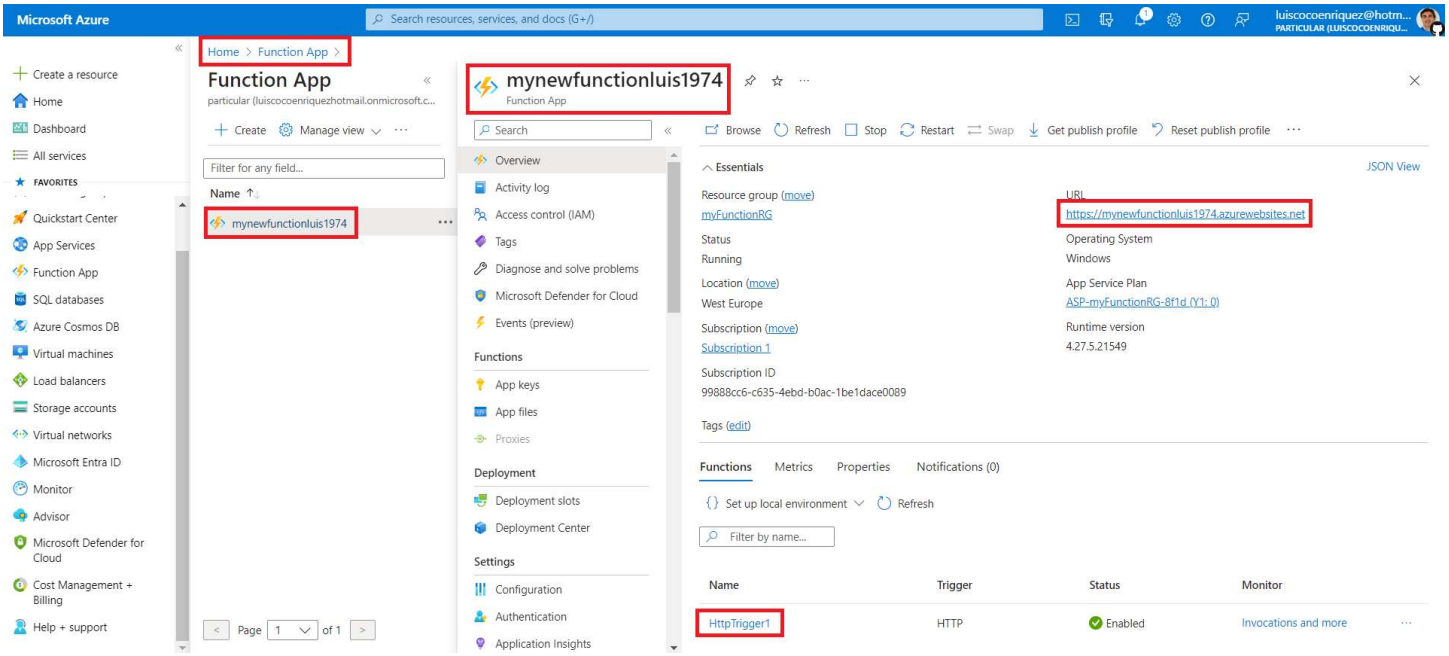
5. Deploy the Azure Function from VSCode to Azure Portal

We can deploy the Azure Function from the Terminal Window in VSCode running this command:

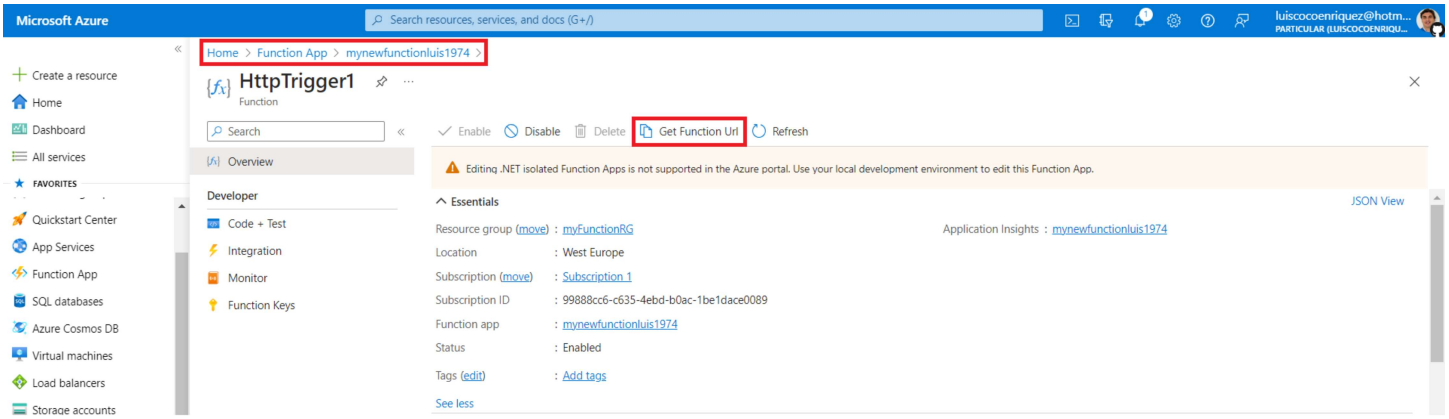
```
func azure functionapp publish <FunctionAppName>
```

6. Test/Verify the Azure Function

In Azure Portal we navigate to the Azure Function and we click on the "HttpTrigger1" link



We click on the option "**Get Function Url**"



And navigate in your internet web browser to the Function endpoint:

https://mynewfunctionluis1974.azurewebsites.net/api/HttpTrigger1?code=PIeNvNG0KeUvWu12oFMuMQ_bUKXDu_kNYJj0aG3LBSjKAzFu0Z0uuQ==&resourceGroupName=holaluis

Pay attention we added "**&resourceGroupName=holaluis**" at the end of the URL for setting the new Azure ResourceGroup name