AzureFunctions: How to create a new Azure Function with only with Commands

Creating an Azure Function in C# with Visual Studio Code involves a few steps.

Make sure you have the necessary tools installed, such as the Azure Functions Core Tools and the Azure Functions extension for Visual Studio Code.

Here's a step-by-step guide:

1. Install Prerequisites:

Install .NET SDK (if not already installed): https://dotnet.microsoft.com/es-es/download/dotnet/8.0

Install Azure Functions Core Tools: https://learn.microsoft.com/en-us/azure/azure-functions/functions-run-local

Install Azure Functions Extension for VSCode.

2. Open Visual Studio Code and install extension.

Go to the Extensions view (you can press Ctrl + Shift + X), and search for "**Azure Functions**"

Install the extension published by Microsoft.

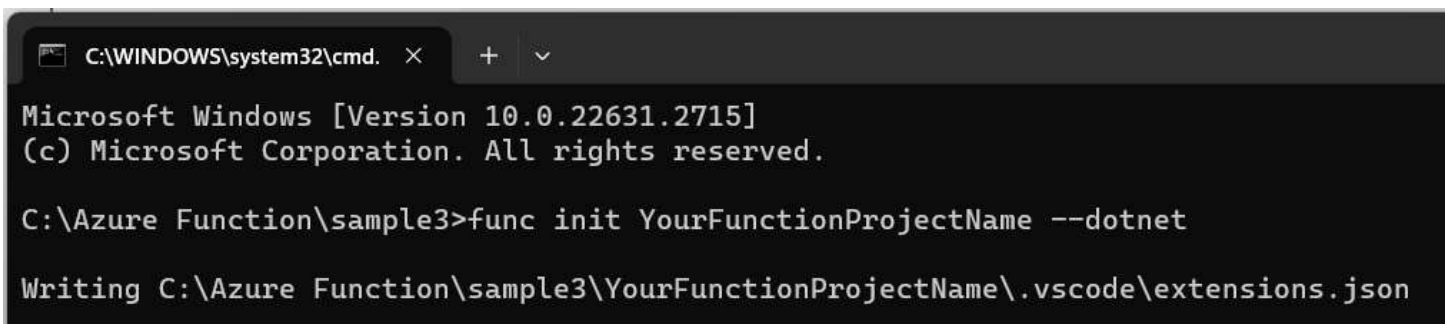
3. Create a new Azure Functions Project:

Open a terminal in Visual Studio Code.

Run the following command to create a new Azure Functions project:

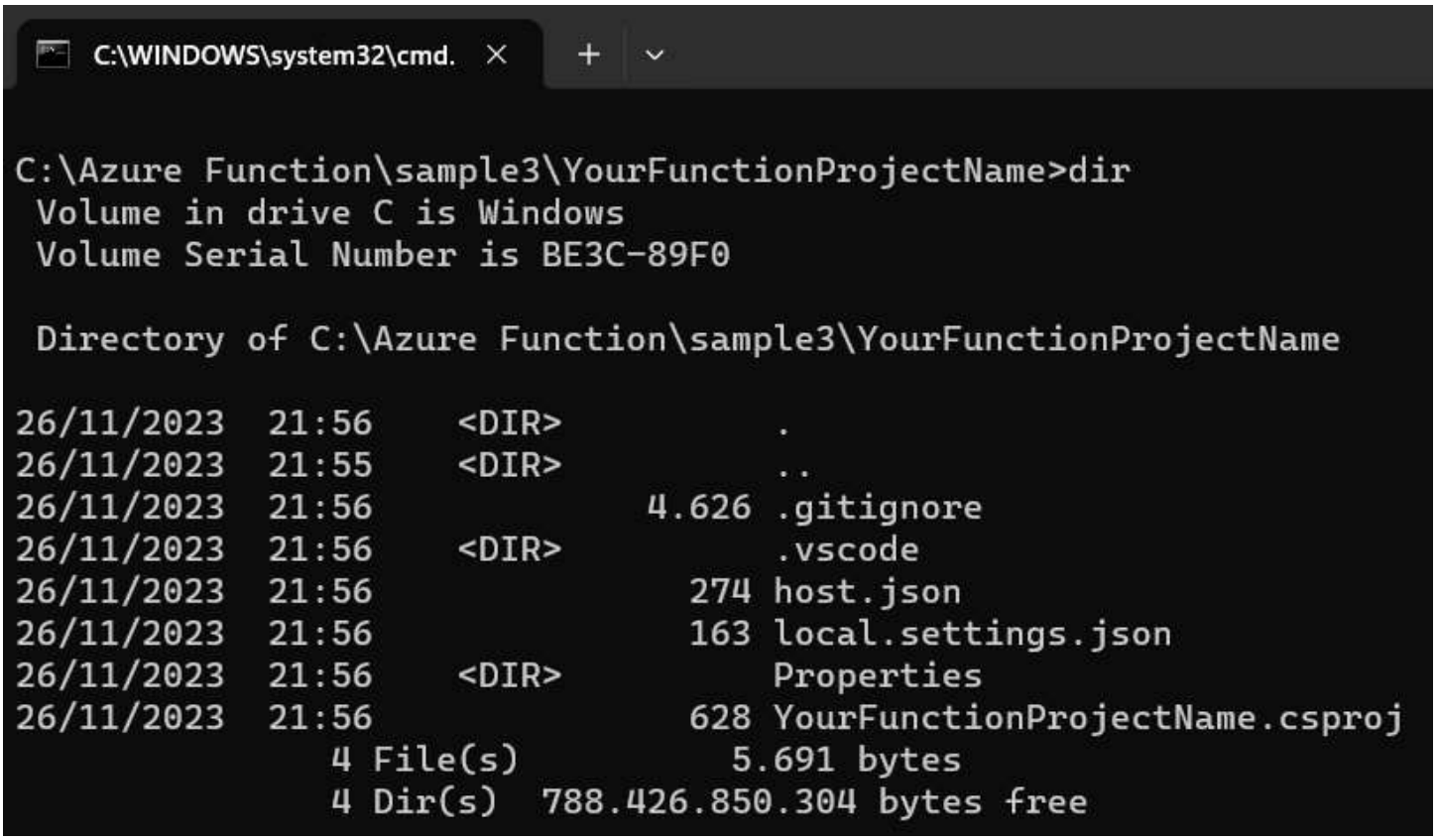
```
func init YourFunctionProjectName --dotnet
```

Replace YourFunctionProjectName with the desired name for your project.



Navigate to the Project Directory:

```
cd YourFunctionProjectName
```



Create a New Azure Function:

Run the following command to create a new Azure Function:

```
func new
```

Follow the prompts to select the template. Choose "HTTP trigger" for a simple example.

```
C:\Azure Function\sample3\YourFunctionProjectName>func new
Use the up/down arrow keys to select a template:
QueueTrigger
HttpTrigger
BlobTrigger
TimerTrigger
KafkaTrigger
KafkaOutput
DurableFunctionsOrchestration
SendGrid
EventHubTrigger
ServiceBusQueueTrigger
ServiceBusTopicTrigger
EventGridTrigger
CosmosDBTrigger
IotHubTrigger
DaprPublishOutputBinding
DaprServiceInvocationTrigger
DaprTopicTrigger
```

We set the new Azure Function name:

```
C:\Azure Function\sample3\YourFunctionProjectName>func new
Use the up/down arrow keys to select a template:Function name: myNewFunction
```

```
C:\Azure Function\sample3\YourFunctionProjectName>func new
Use the up/down arrow keys to select a template:Function name: myNewFunction
myNewFunction

Creating dotnet function...
The function "myNewFunction" was created successfully from the "HttpTrigger" template.

C:\Azure Function\sample3\YourFunctionProjectName>
```

```
C:\Azure Function\sample3\YourFunctionProjectName>dir
 Volume in drive C is Windows
 Volume Serial Number is BE3C-89F0

 Directory of C:\Azure Function\sample3\YourFunctionProjectName

26/11/2023  22:00    <DIR>          .
26/11/2023  21:55    <DIR>          ..
26/11/2023  21:56             4.626 .gitignore
26/11/2023  21:56    <DIR>          .vscode
26/11/2023  21:56               274 host.json
26/11/2023  21:56               163 local.settings.json
26/11/2023  22:00             1.315 myNewFunction.cs
26/11/2023  21:56    <DIR>          Properties
26/11/2023  21:56               628 YourFunctionProjectName.csproj
               5 File(s)          7.006 bytes
               4 Dir(s)  788.419.620.864 bytes free
```

We also can set in the commands: the Azure **FunctionName**, the template, the **authorization level**, and the **runtime**.

```
func init YourFunctionProjectName --dotnet
cd YourFunctionProjectName
func new --name MyNewFunction --template "HTTP trigger" --authlevel "anonymous" --runtime dotn
```

4. Azure Function C# source code.

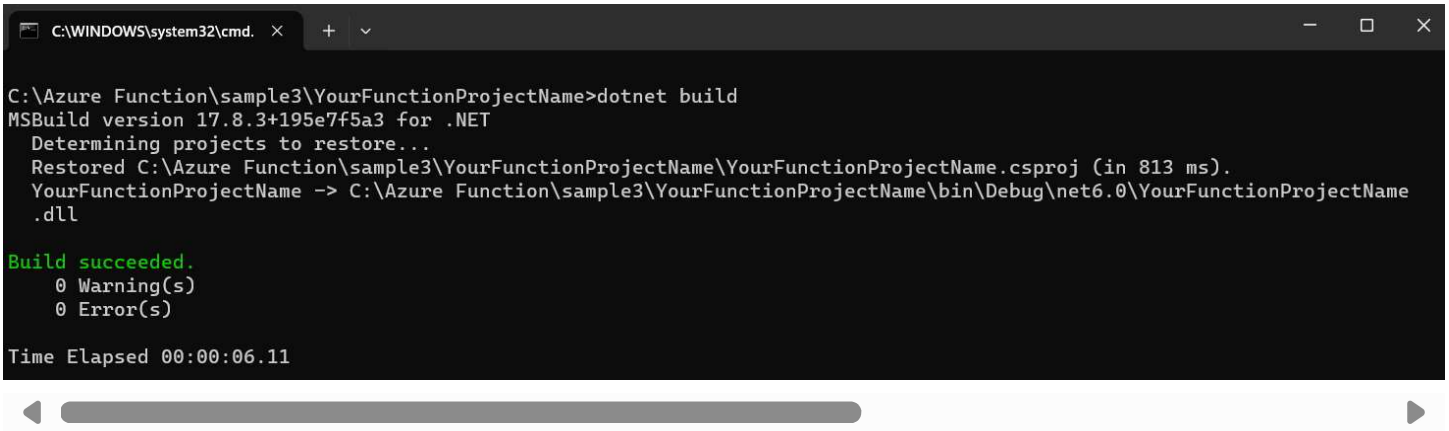
```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace YourFunctionProjectName
{
    public static class myNewFunction
    {
        [FunctionName("myNewFunction")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpReques
            ILogger log)
        {
```

```
            log.LogInformation("C# HTTP trigger function processed a request.");

            string name = req.Query["name"];

            string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
            dynamic data = JsonConvert.DeserializeObject(requestBody);
            name = name ?? data?.name;

            string responseMessage = string.IsNullOrEmpty(name)
                ? "This HTTP triggered function executed successfully. Pass a name in the quer
                : $"Hello, {name}. This HTTP triggered function executed successfully.";

            return new OkObjectResult(responseMessage);
        }
    }
}
```

5. Build the Project.

Run the following command to build the project:

```
dotnet build
```

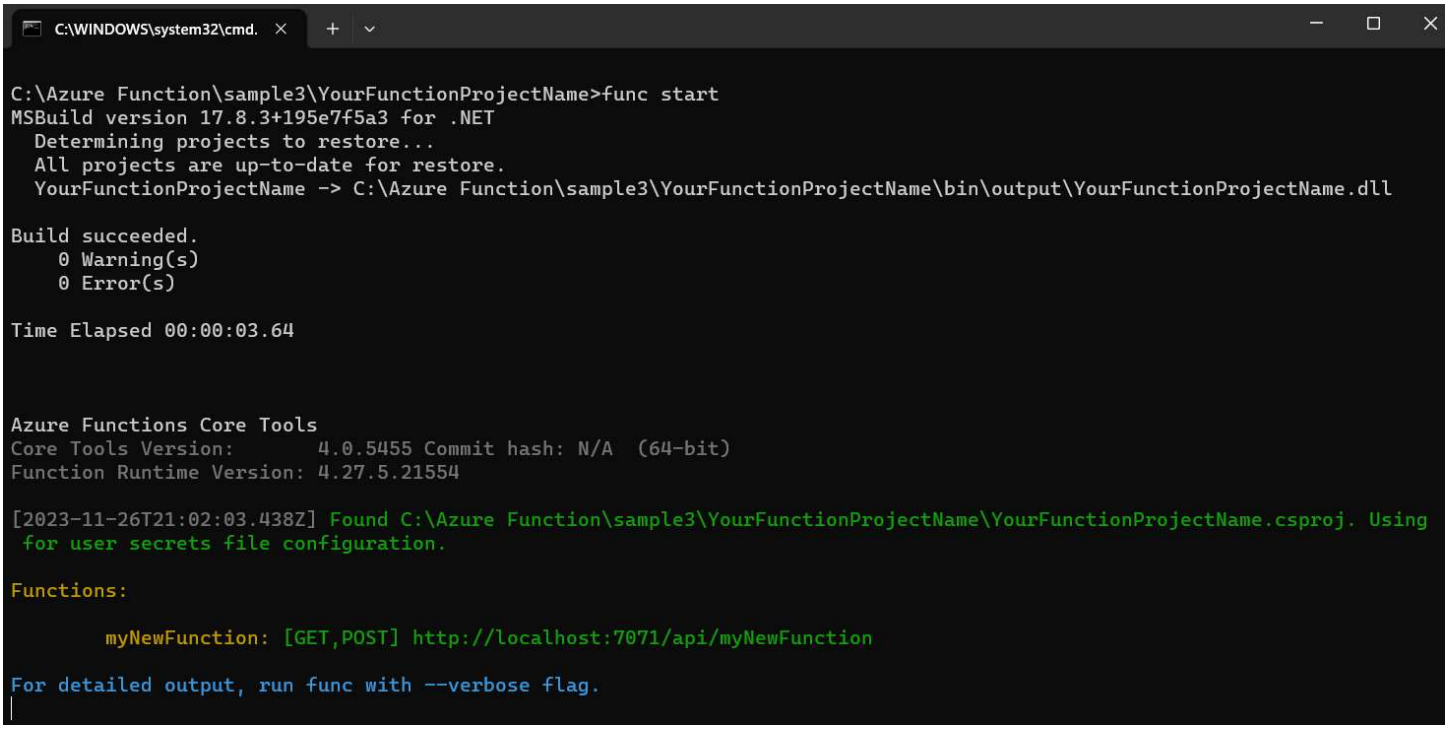


6. Run the Function Locally:

Run the following command to start the function locally:

```
func start
```

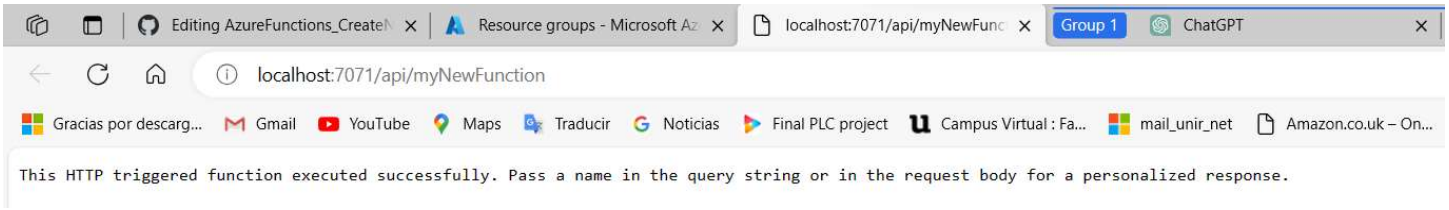
This will start a local development server.



7. Test the Function:

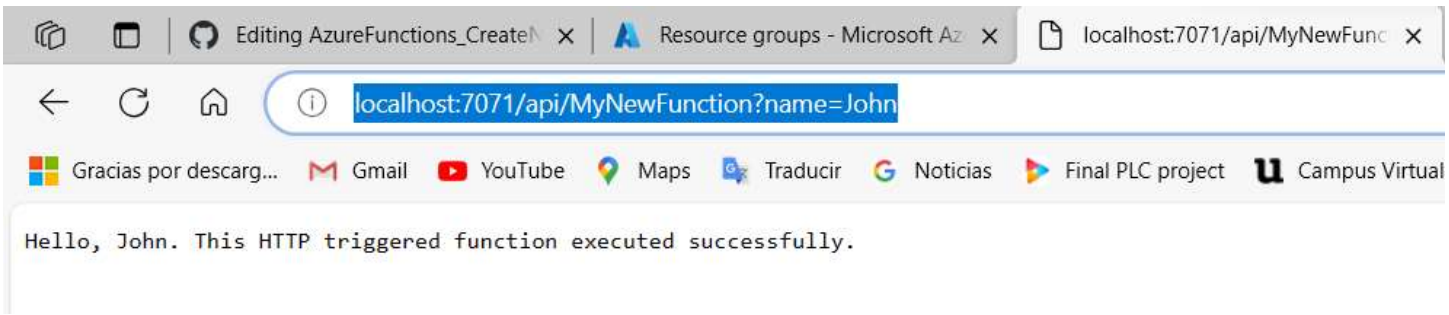
Open a web browser or a tool like Postman and send a request to the locally running function.

The URL will be displayed in the terminal.



We can also invoke the Azure Function setting the parameter "name"

http://localhost:7071/api/MyNewFunction?name=John



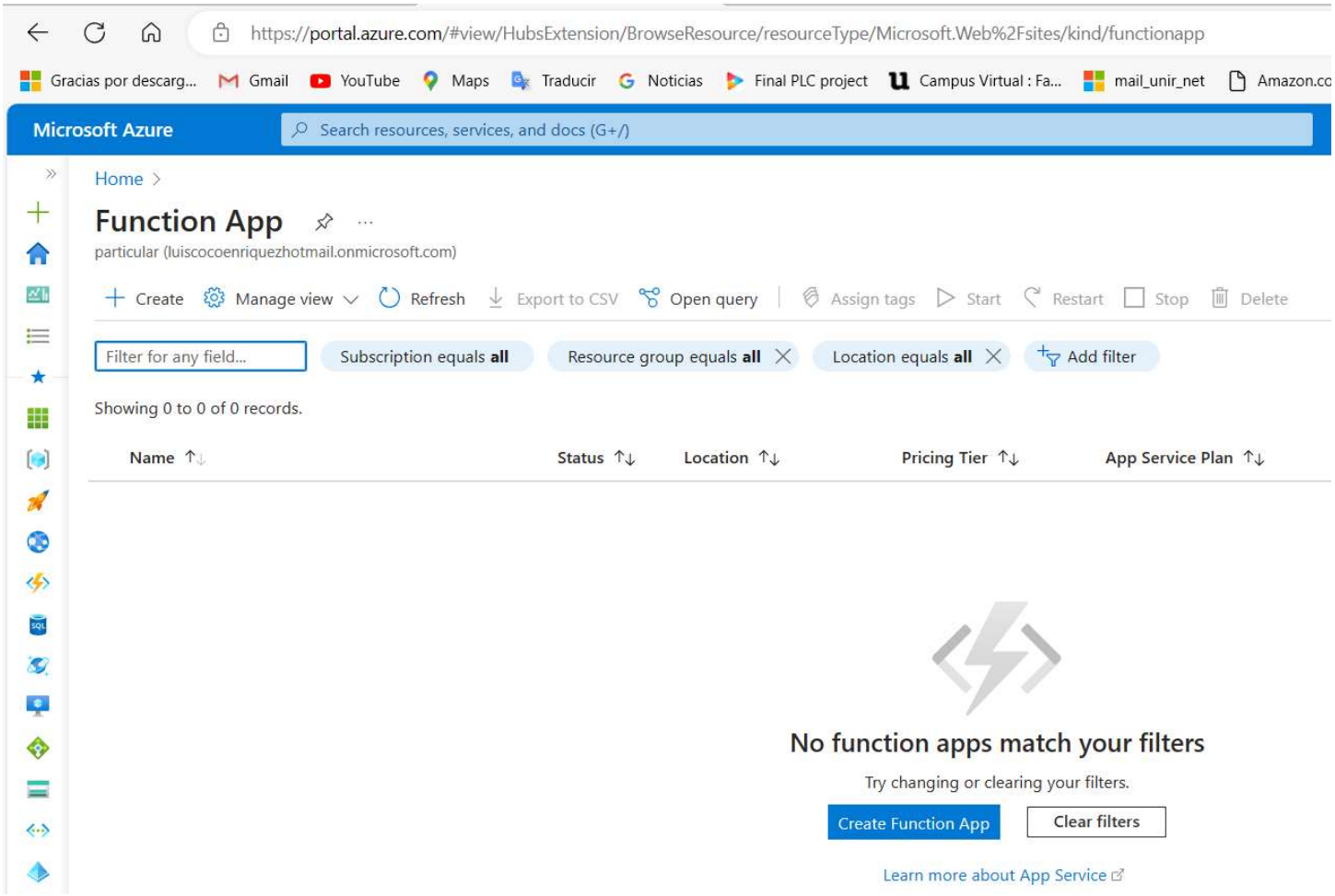
8. Publish to Azure (Optional):

Make sure you have an Azure Function App created in the Azure portal.

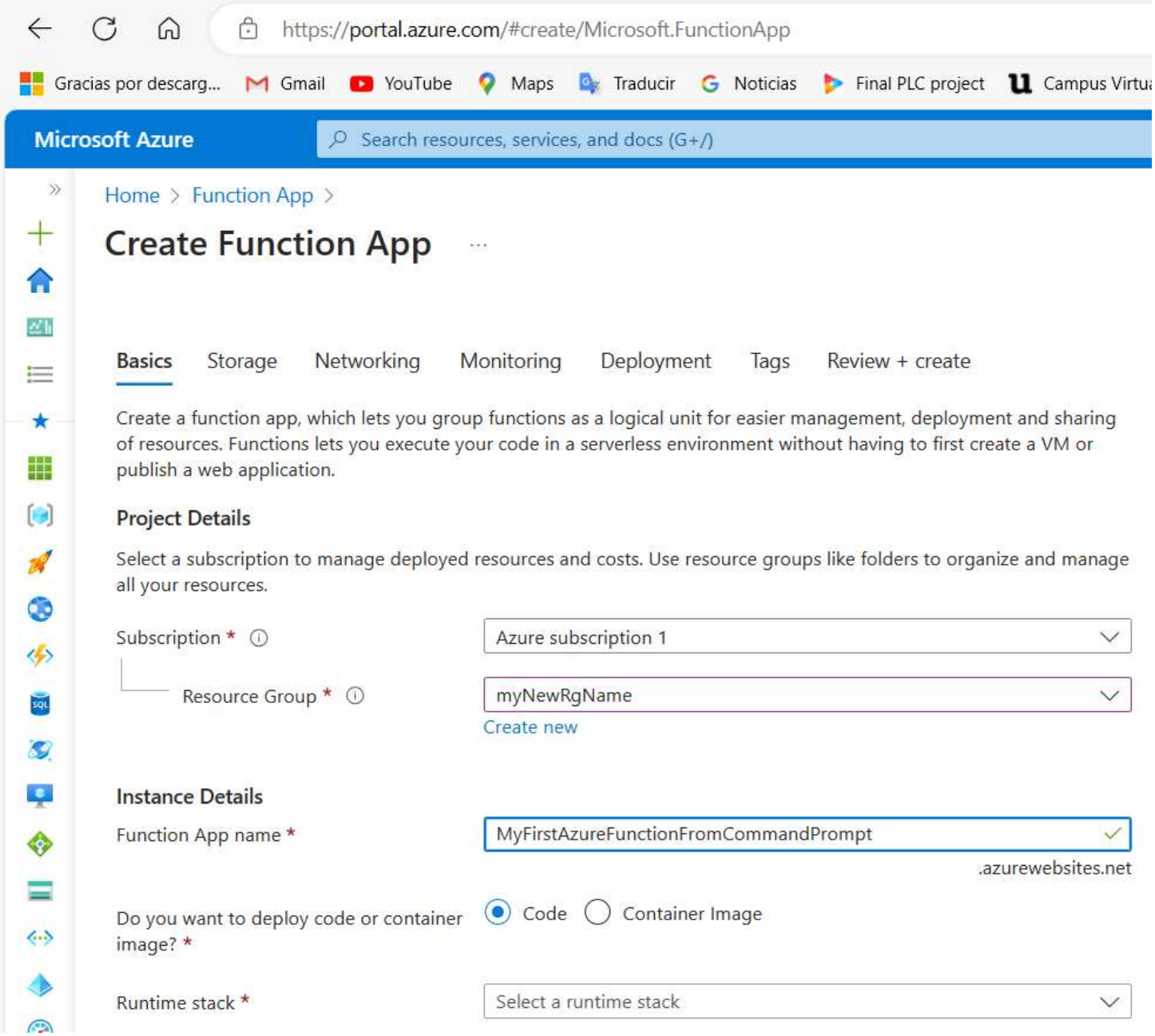
If not, you'll need to create one.

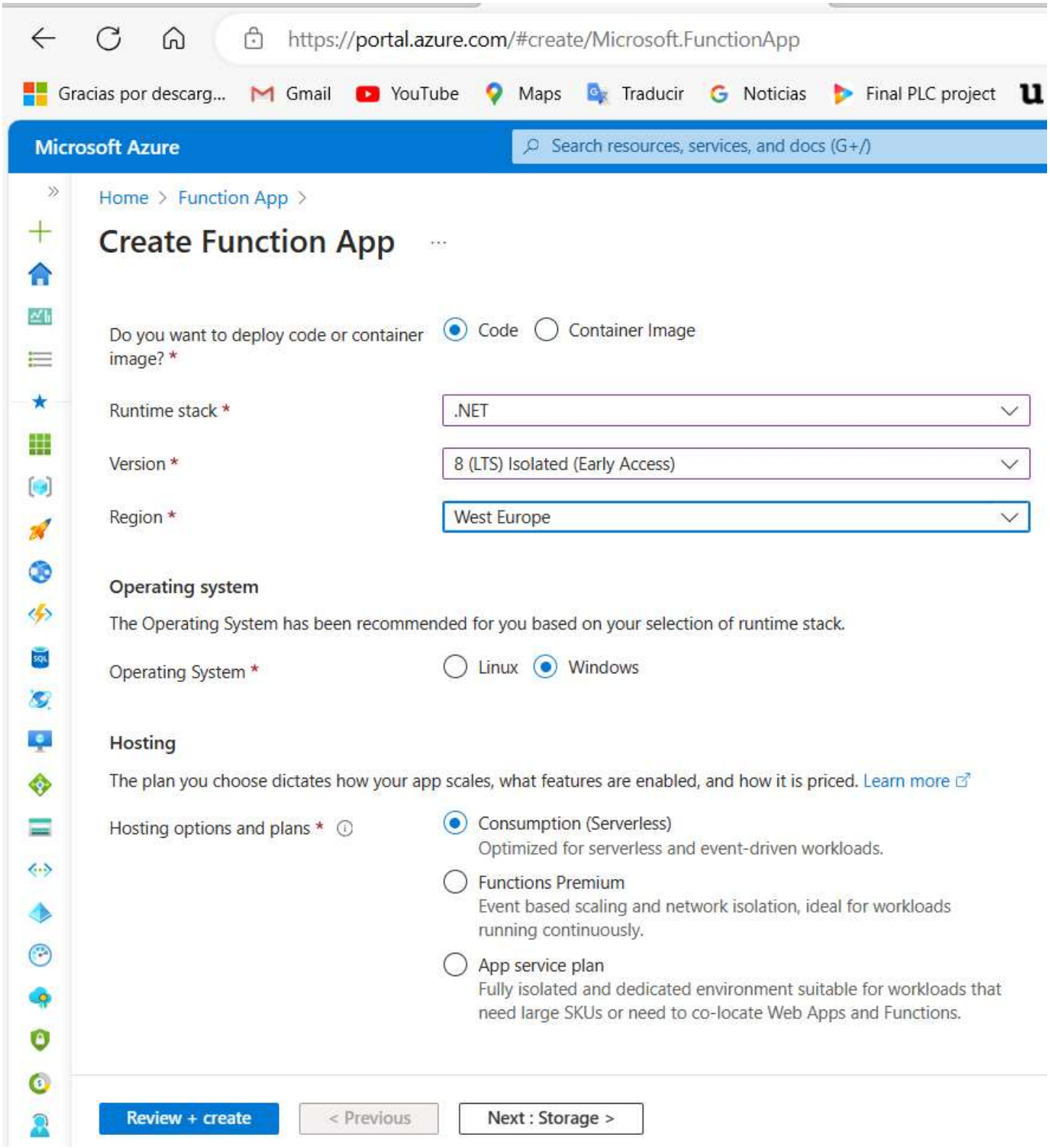
The name you provide here is what you'll use in the func azure functionapp publish command.

For creating a new Azure Function open this service in Azure Portal and press on the "Create Function App" button:



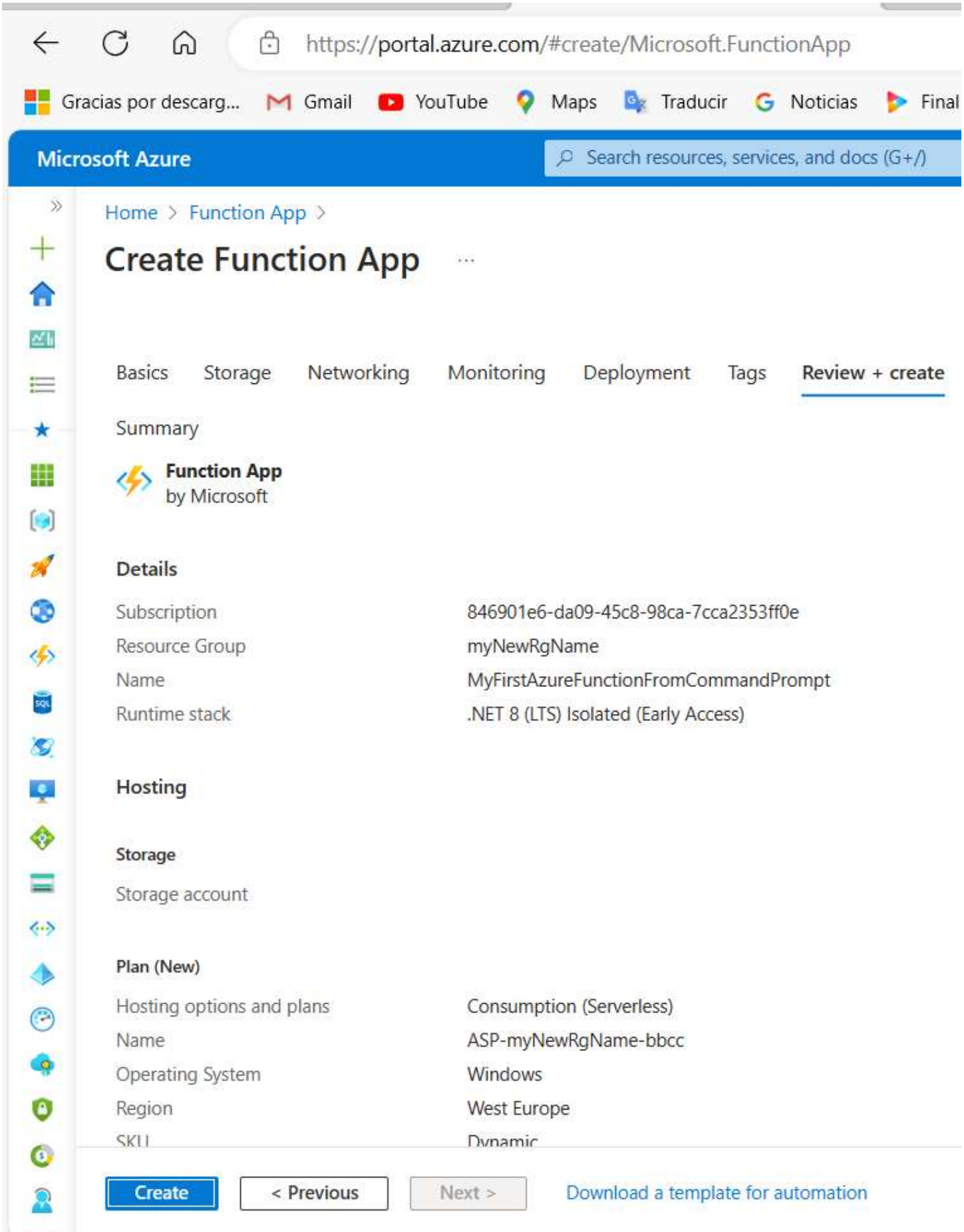
The set the input data requested for creating the new Azure Function:


https://portal.azure.com/#create/Microsoft.FunctionApp
Gracias por descarg...
Gmail
YouTube
Maps
Traducir
Noticias
Final PLC project
Campus Virtua
Microsoft Azure
Search resources, services, and docs (G+/)
Home > Function App >
Create Function App
Basics
Storage
Networking
Monitoring
Deployment
Tags
Review + create
Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.
Project Details
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
Azure subscription 1
Resource Group *
myNewRgName
Create new
Instance Details
Function App name *
MyFirstAzureFunctionFromCommandPrompt
.azurewebsites.net
Do you want to deploy code or container image? *
Code
Container Image
Runtime stack *
Select a runtime stack

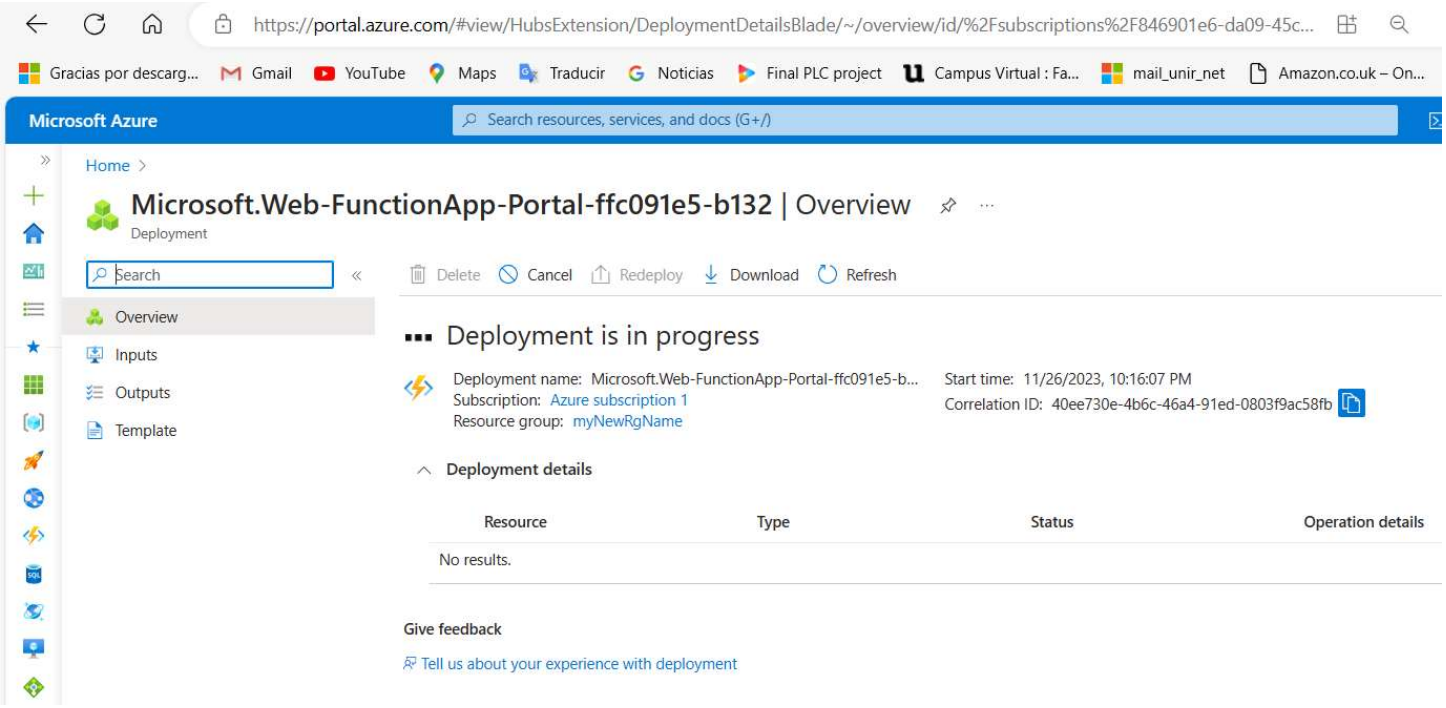


Then press the button "Review + create"

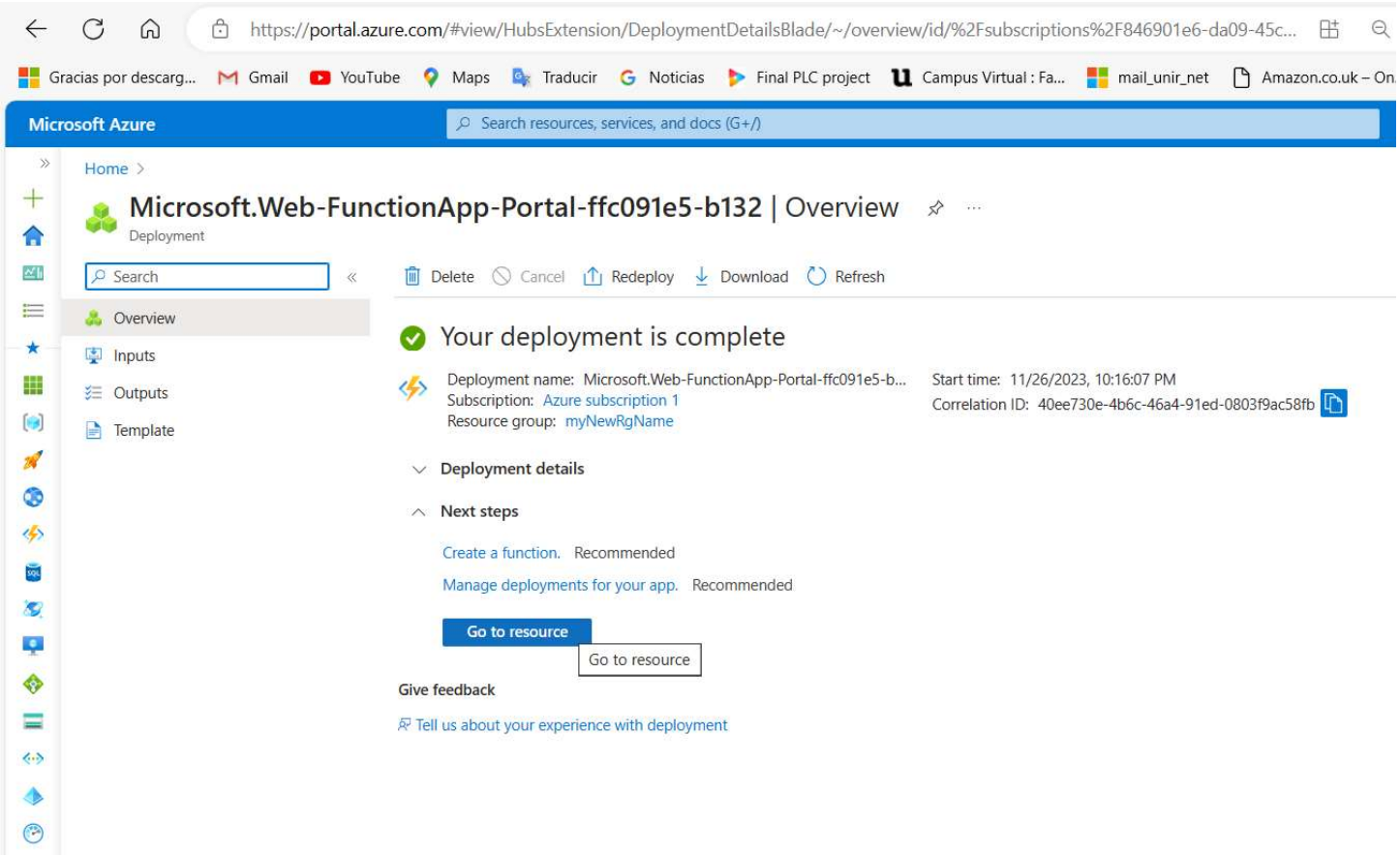
Then press the button "Create"

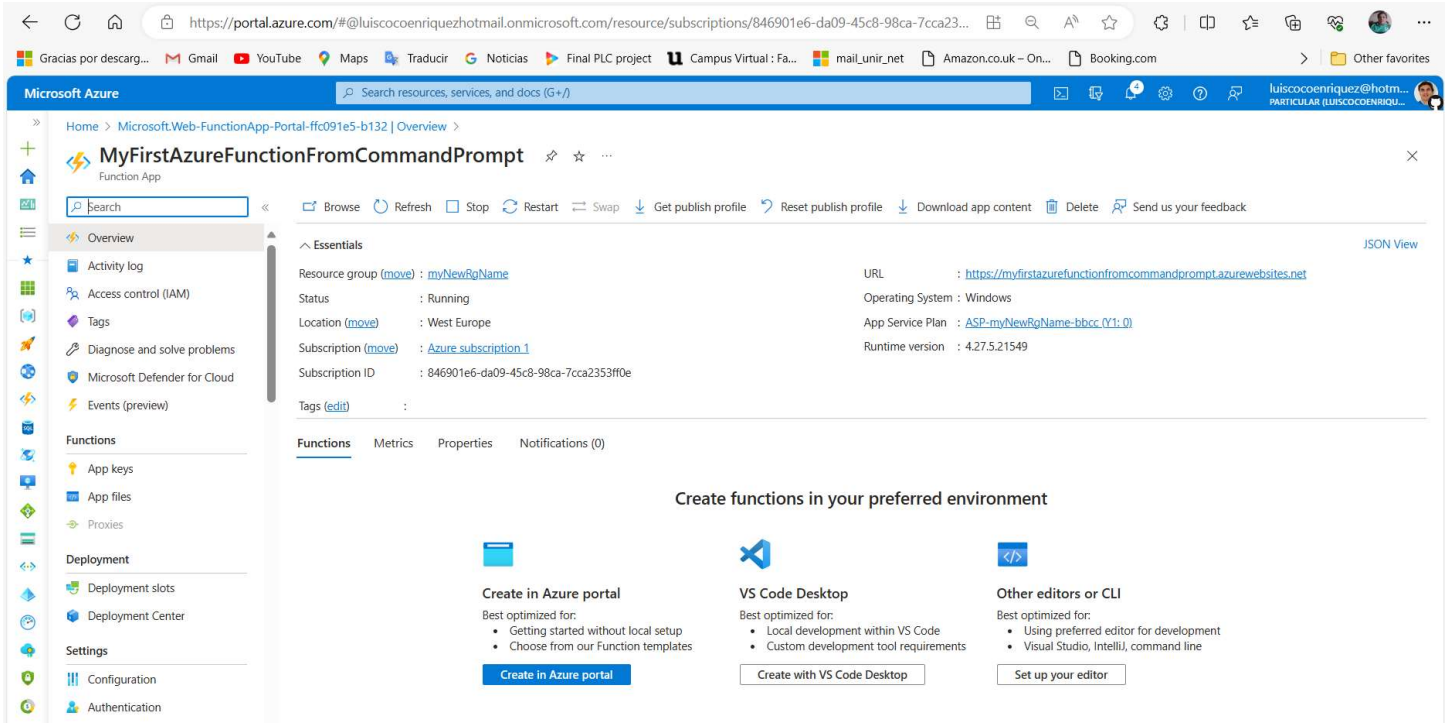


Now the new Azure Function deployment is in progress



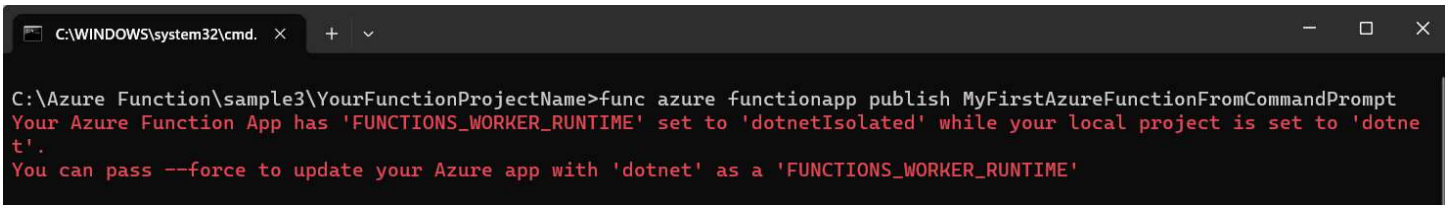
When the deployment is finished you can navigate to the new resource, press the "Go to resource" button:





If you want to deploy your function to Azure, you can run the following command:

```
func azure functionapp publish MyFirstAzureFunctionFromCommandPrompt
```



This error is telling you that there is a mismatch between the **FUNCTIONS_WORKER_RUNTIME** setting in your Azure Function App and your local project.

In Azure, the app is set to use **dotnetIsolated**, while your local project is set to **dotnet**.

To resolve this, you have a couple of options:

Update Local Project to Use dotnetIsolated:

Open your YourFunctionProjectName.csproj file.

Change the element to **net6.0-isolated** (or the version you are using).

Save the file.

Example:

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
```

```
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0-isolated</TargetFramework>
  </PropertyGroup>

</Project>
```

Pass --force to Update Azure App:

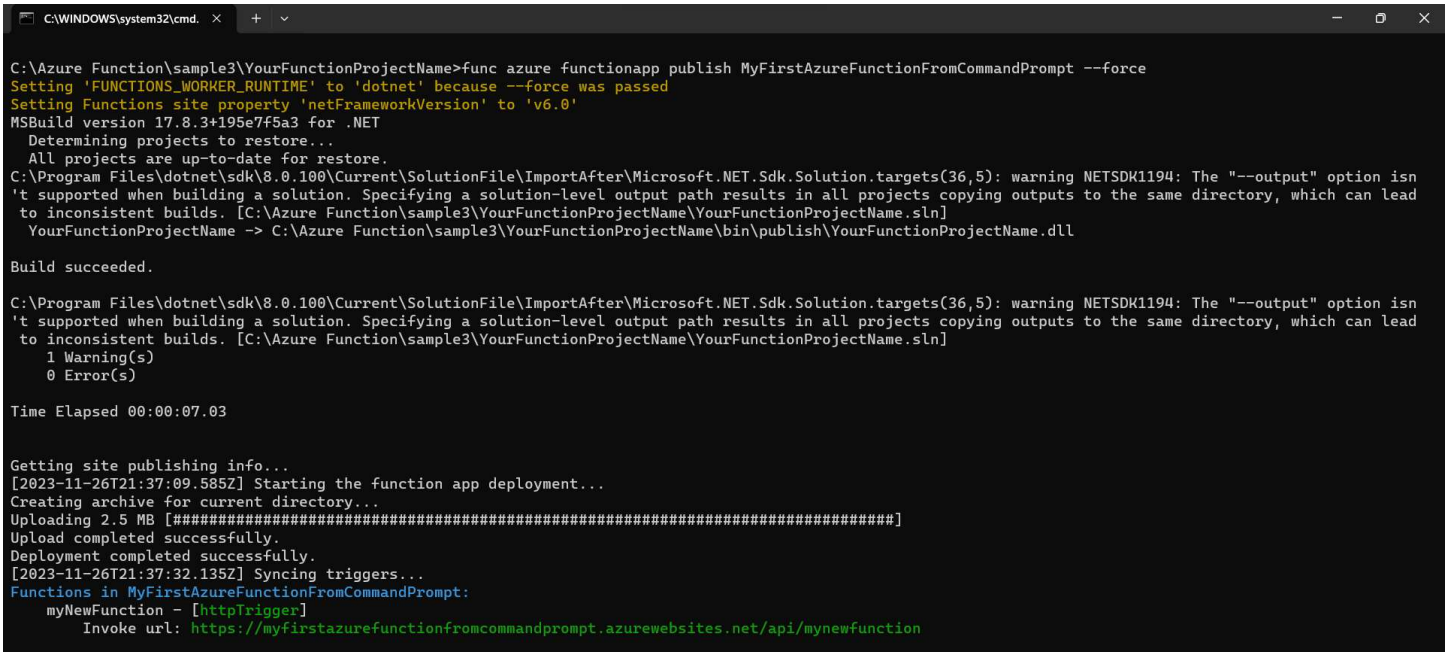
Run the following command, passing the **--force** option:

```
func azure functionapp publish MyFirstAzureFunctionFromCommandPrompt --force
```

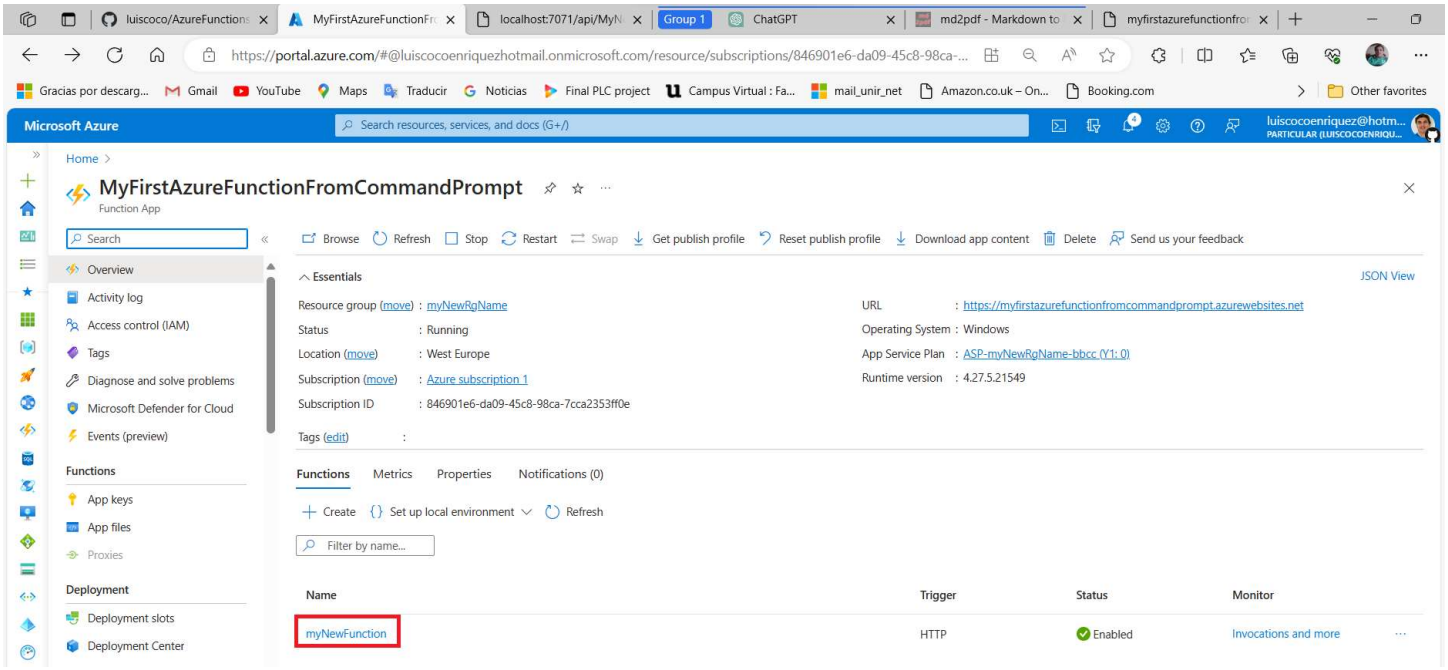
This will update your Azure Function App to use dotnet as the FUNCTIONS_WORKER_RUNTIME.

Choose the option that best fits your requirements.

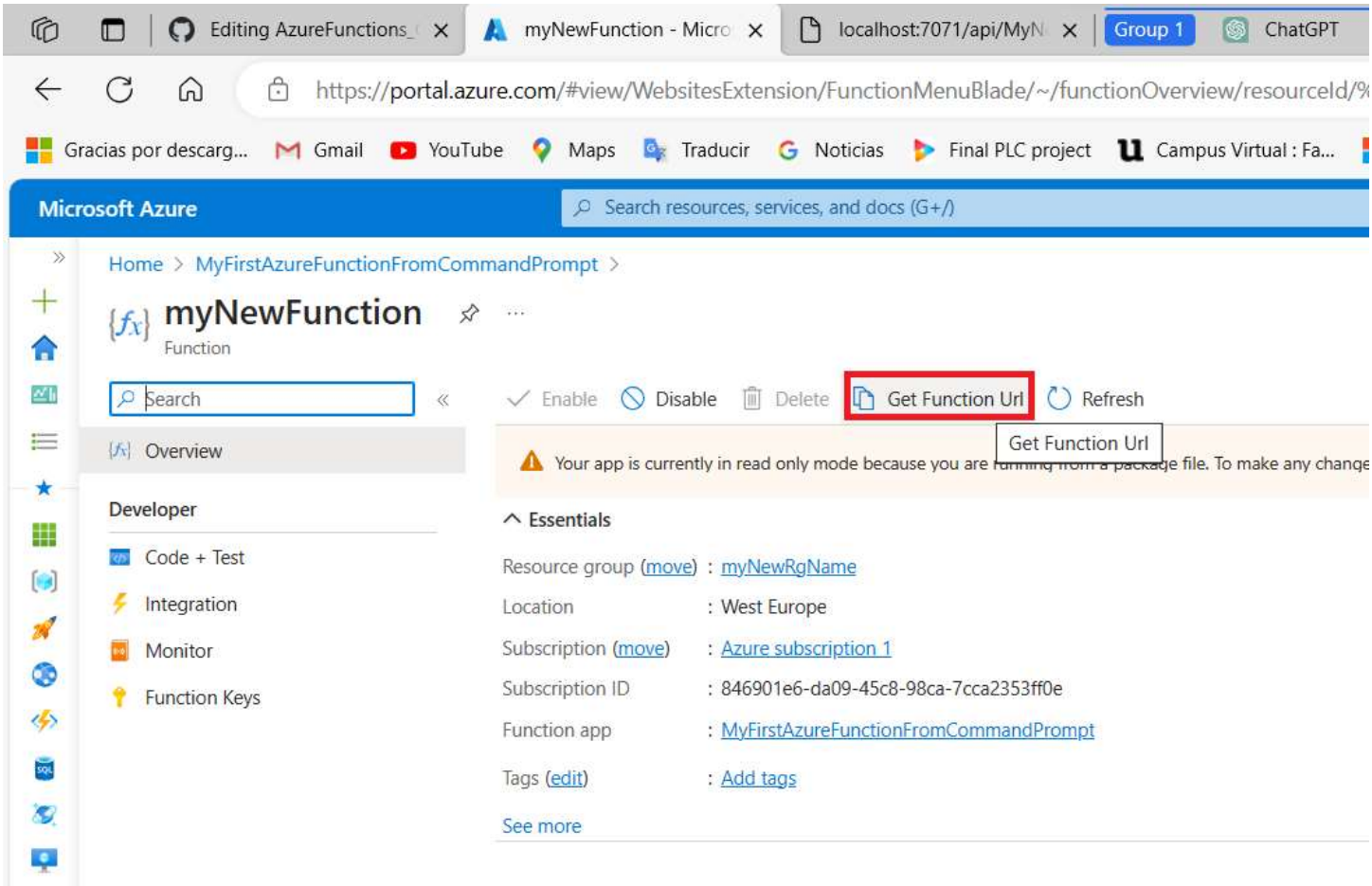
If you're using the isolated process model locally (net6.0-isolated), it's a good idea to update your Azure Function App to match.



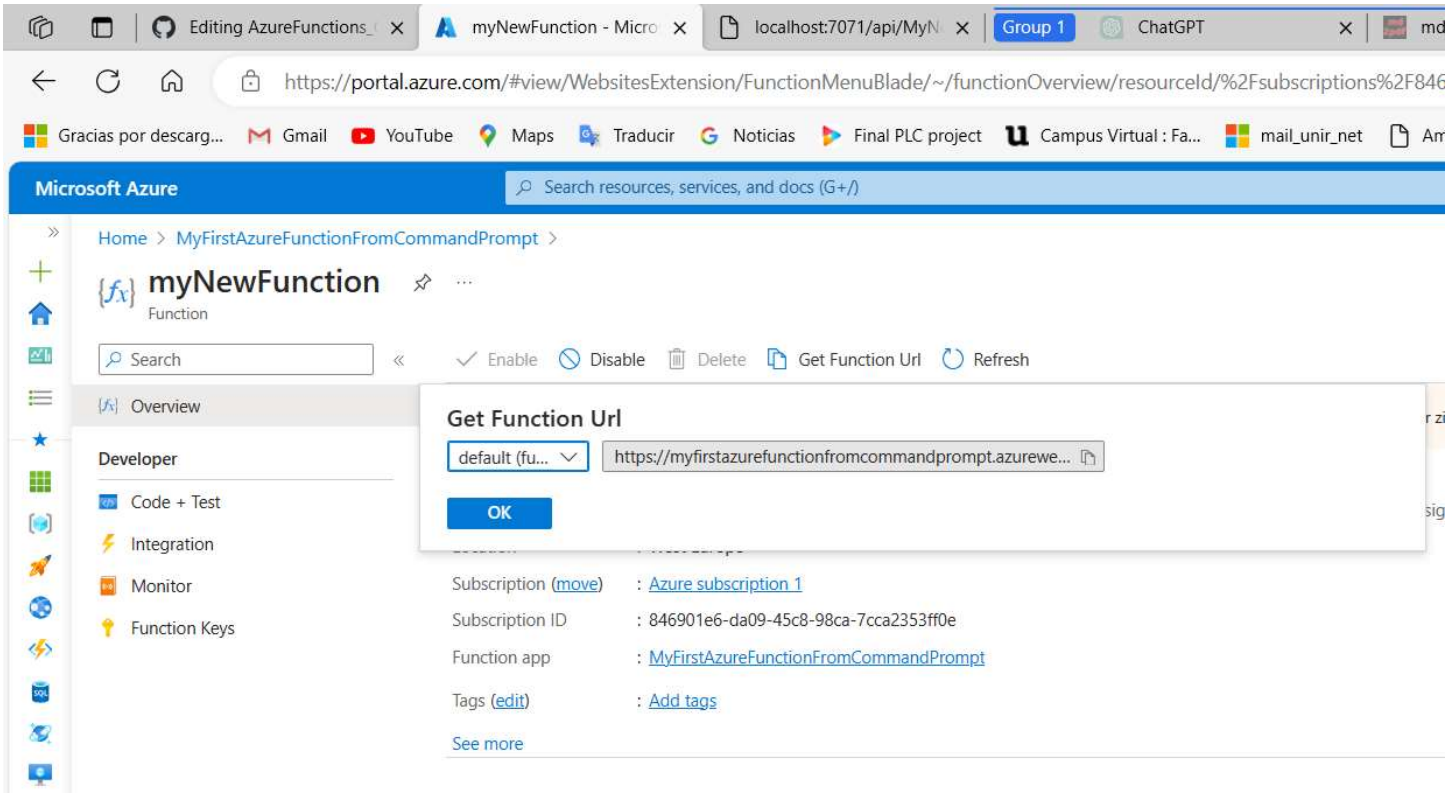
After the deployment go to Azure portal and navigate to the new Azure Function



Now we press in the "**Get Function Url**" option



Then we copy and paste the Azure Function Url in the internet web browser:



We check the output