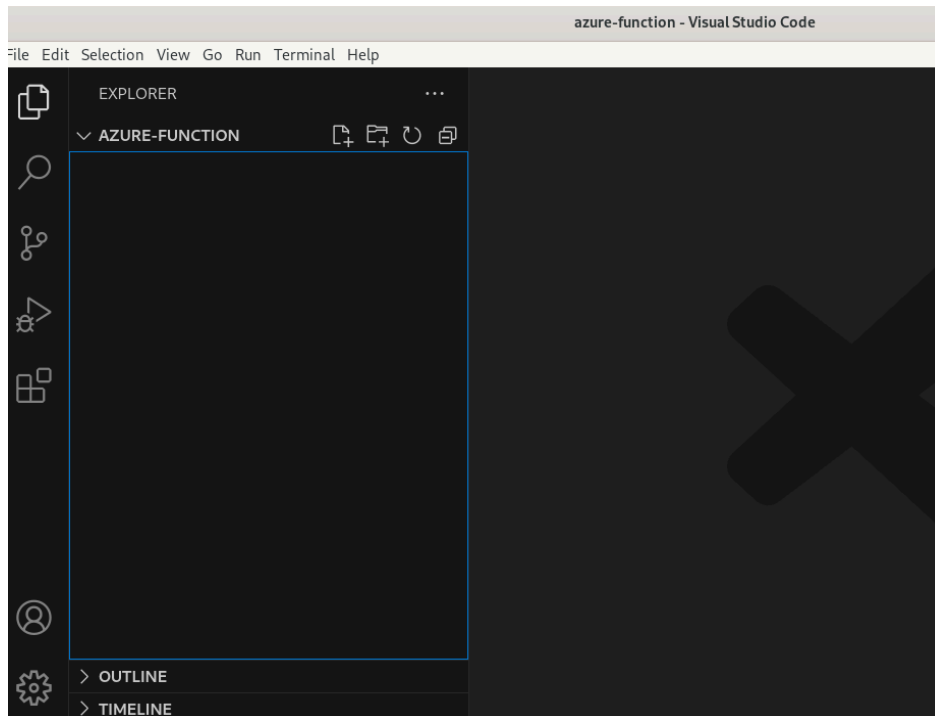




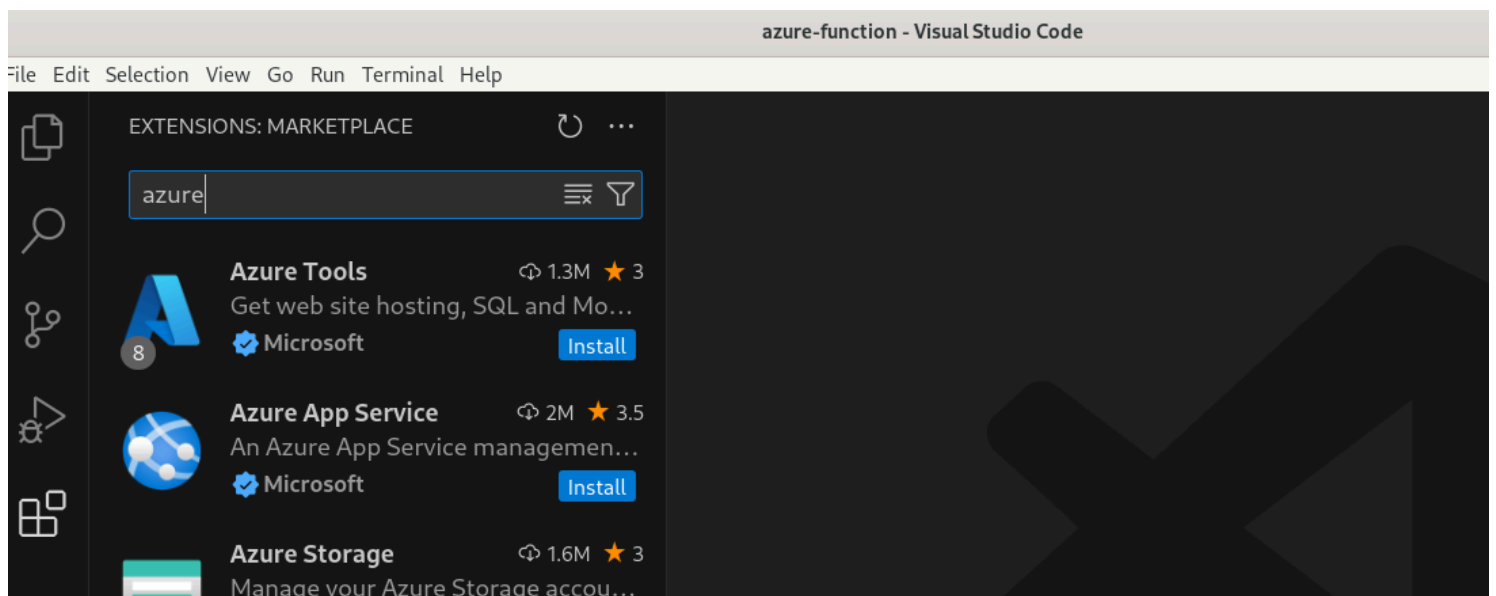
CÓMO CREAR Y DESPLEGAR UNA APLICACIÓN SERVERLESS CON FUNCTIONS DE AZURE

PARTE 1: CREACIÓN Y EJECUCIÓN LOCAL

1. Creamos la carpeta del proyecto y la abrimos en Visual Studio Code



2. Vamos a extensiones e instalamos Azure Tools

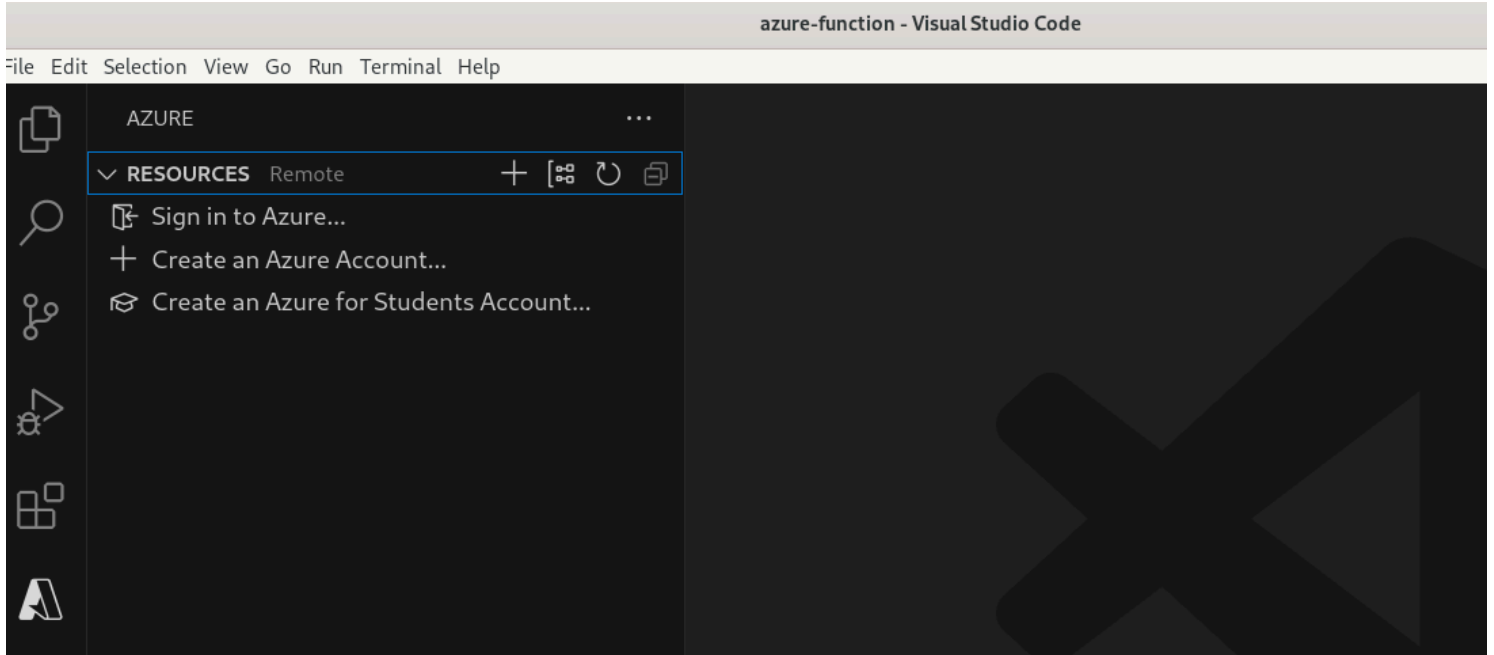




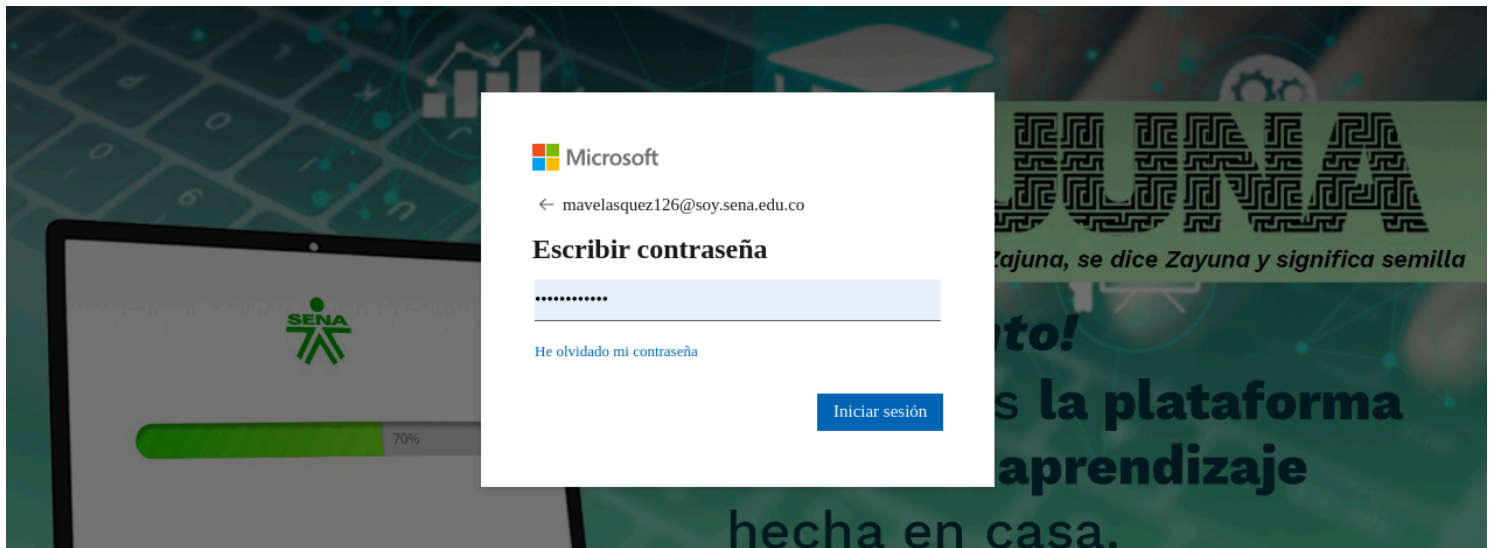
Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2024
Instructor: Germán Alberto Angarita Henao



3. Vamos a Azure Tools y hacemos click en Sign in to Azure

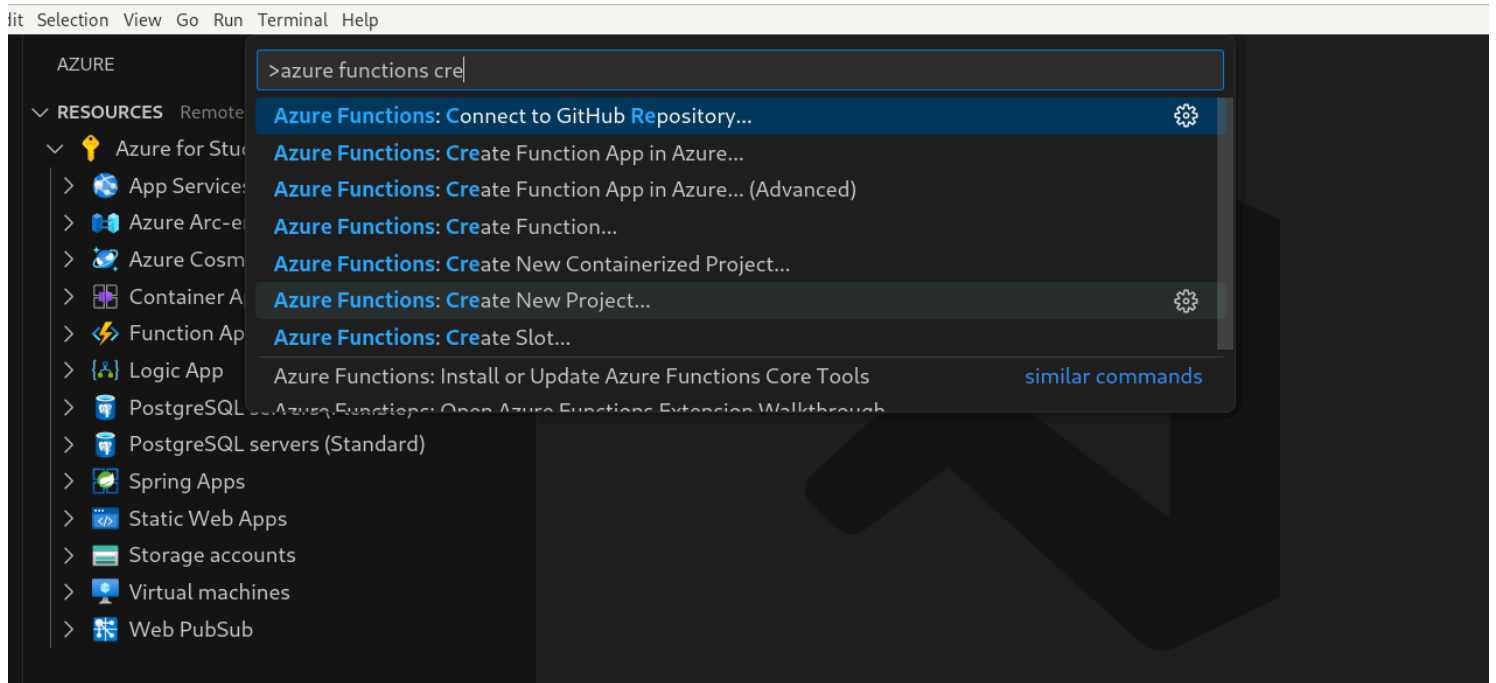


4. Se abre el navegador y hacemos login con nuestras credenciales de Azure(correo soyseña en nuestro caso)

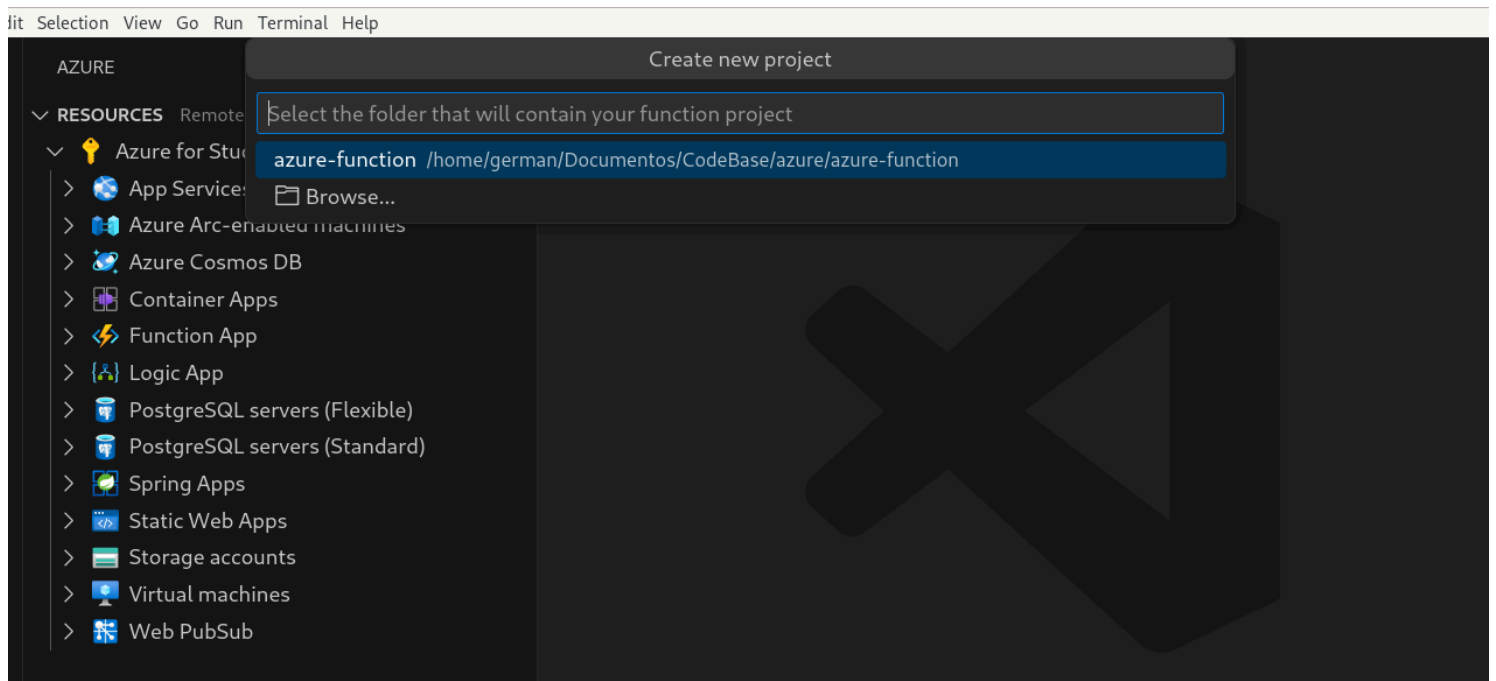




5. Una vez logueados, presionamos F1 y ejecutamos el comando **Azure Functions: Create New Project.**



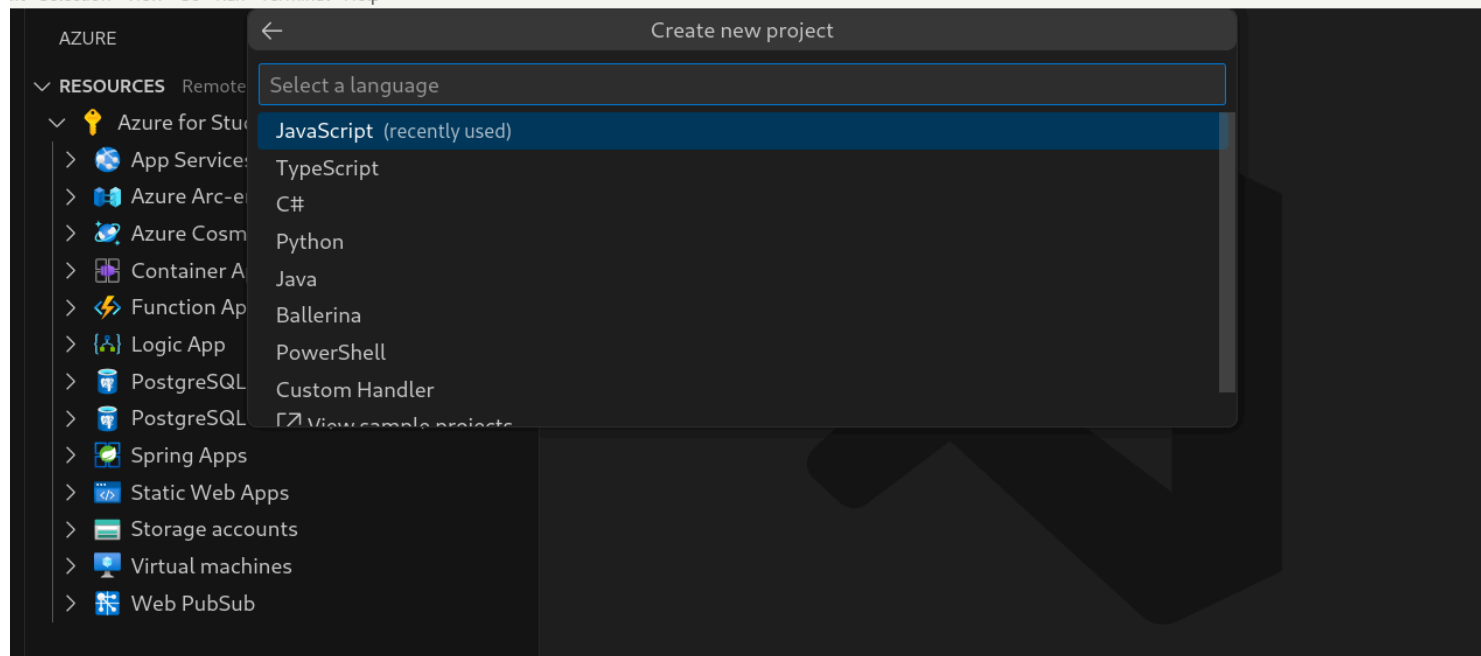
6. Seleccionamos la carpeta creada al inicio como la carpeta del proyecto



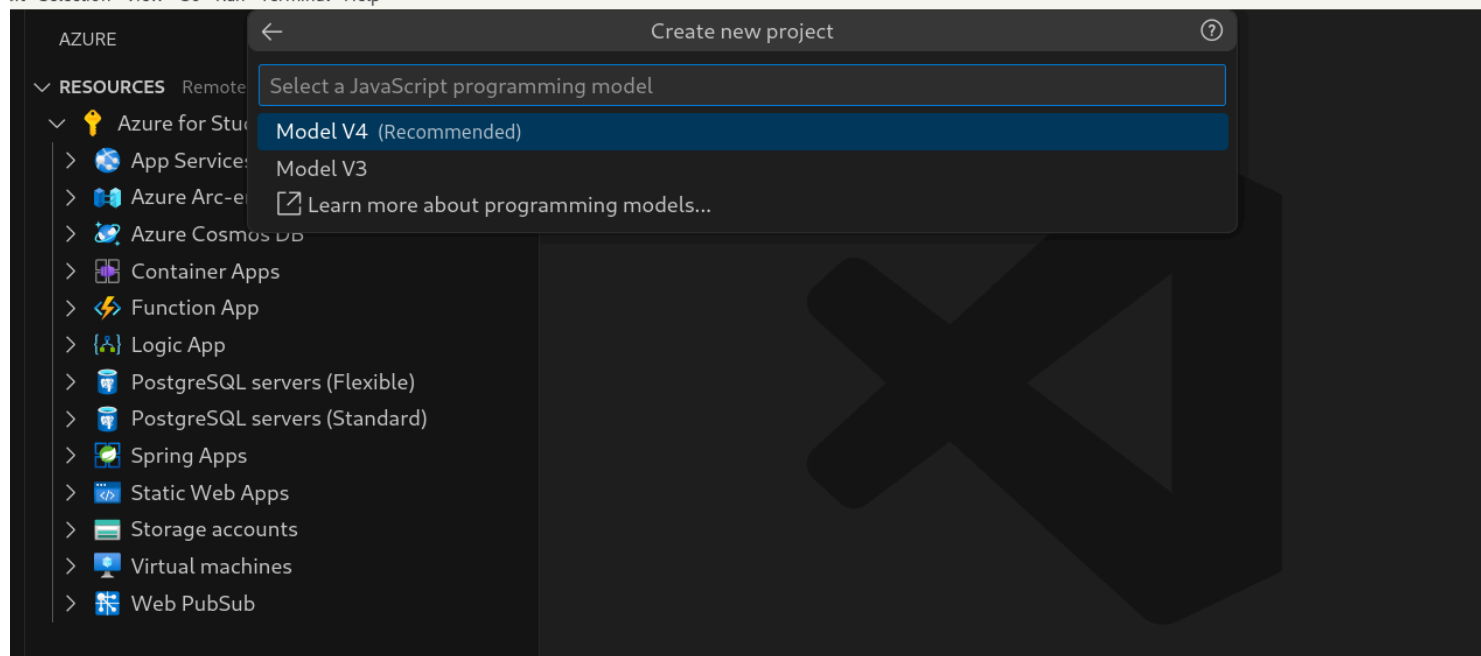


7. Seleccionamos el lenguaje de la función serverless

File Selection View Go Run Terminal Help

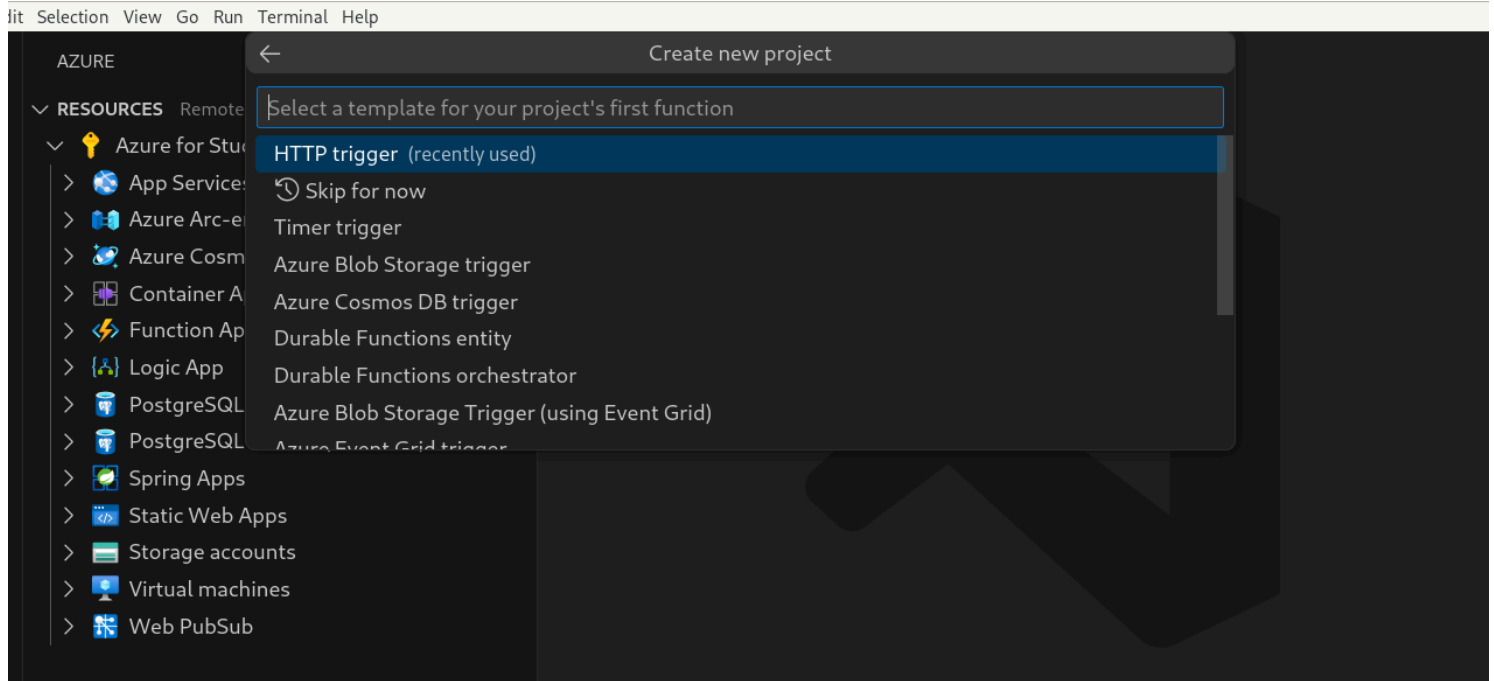


File Selection View Go Run Terminal Help

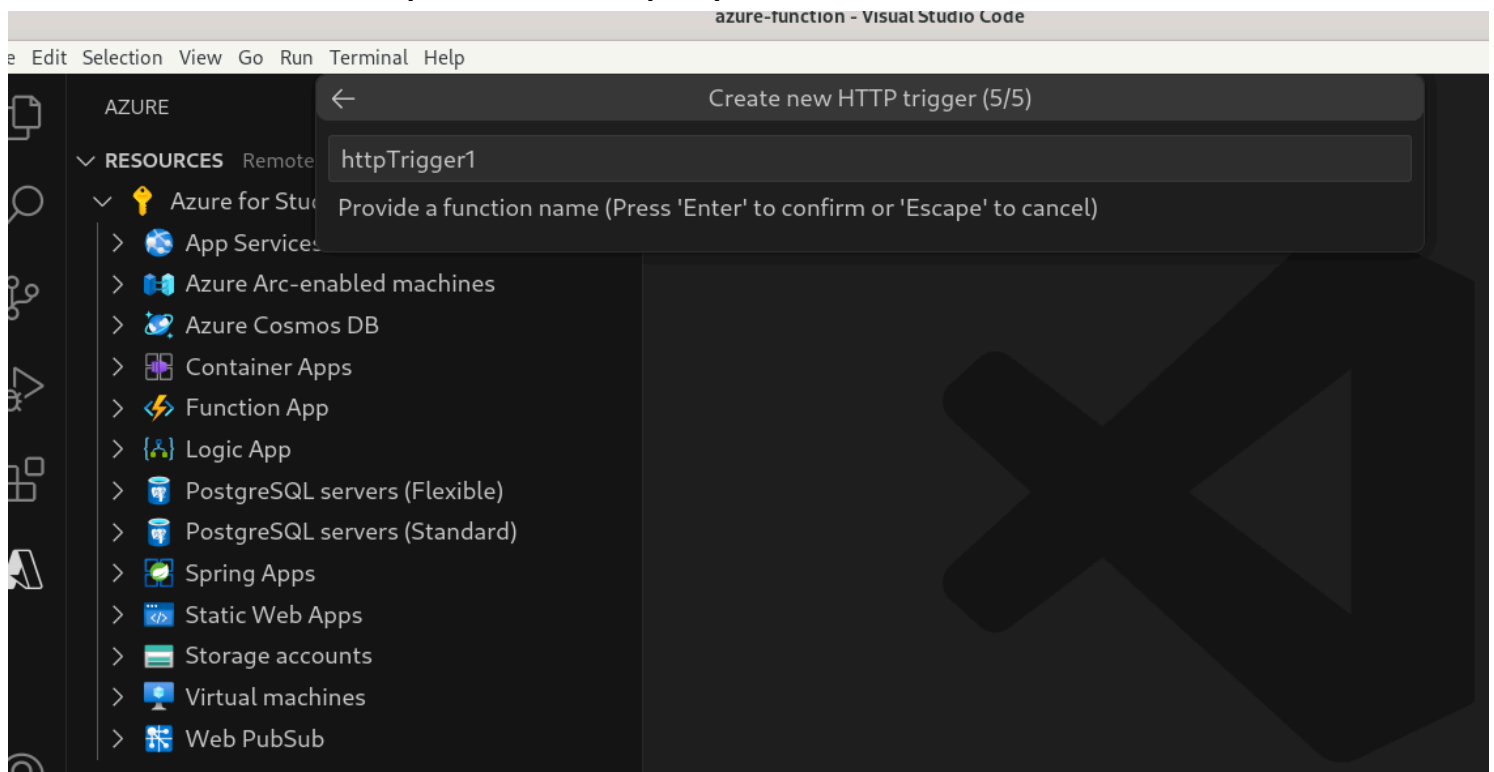




8. Seleccionamos la primera opción como la plantilla del proyecto a usar



9. Seleccionamos el nombre predeterminado que aparece en la barra como nombre de la función





10. En nuestra carpeta de proyecto ya aparecen los archivos de la función serverless

httpTrigger1.js - azure-function - visual studio code

File Edit Selection View Go Run Terminal Help

EXPLORER

- AZURE-FUNCTION
 - .vscode
 - node_modules
 - src
 - functions
 - JS httpTrigger1.js**
 - JS index.js
 - .funcignore
 - .gitignore
 - host.json
 - local.settings.json
 - package-lock.json
 - package.json

```
src > functions > JS httpTrigger1.js > ...
1  const { app } = require('@azure/functions');
2
3  app.http('httpTrigger1', {
4    methods: ['GET', 'POST'],
5    authLevel: 'anonymous',
6    handler: async (request, context) => {
7      context.log('Http function processed
8
9      const name = request.query.get('name'
10
11      return { body: `Hello, ${name}!` };
12    }
13  });
14
```

11. Presionamos F5 para ejecutar la función en local, seleccionamos Use Local Emulator

httpTrigger1.js - azure-function - visual studio code

File Edit Selection View Go Run Terminal Help

EXPLORER

- AZURE-FUNCTION
 - .vscode
 - node_modules
 - src
 - functions
 - JS httpTrigger1.js**
 - JS index.js
 - .funcignore
 - .gitignore
 - host.json
 - local.settings.json
 - package-lock.json
 - package.json

```
src > functions > JS httpTrigger1.js > ...
1  const { app } = require('@azure/functions');
2
3  app.http('httpTrigger1', {
4    methods: ['GET', 'POST'],
5    authLevel: 'anonymous',
6    handler: async (request, context) => {
7      context.log('Http function processed
8
9      const name = request.query.get('name'
10
11      return { body: `Hello, ${name}!` };
12    }
13  });
14
```

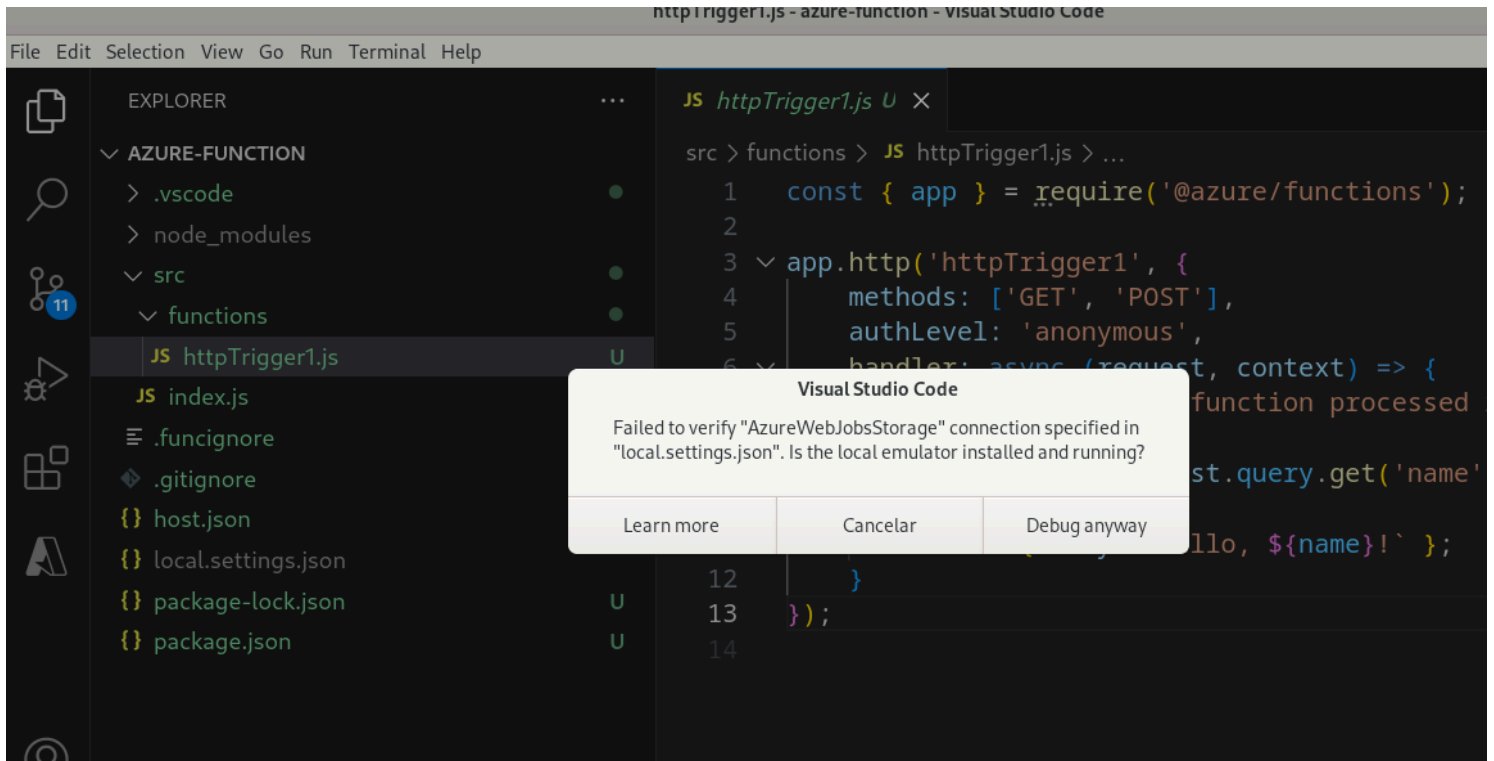
Visual Studio Code

In order to proceed, you must connect a storage account for internal use by the Azure Functions runtime.

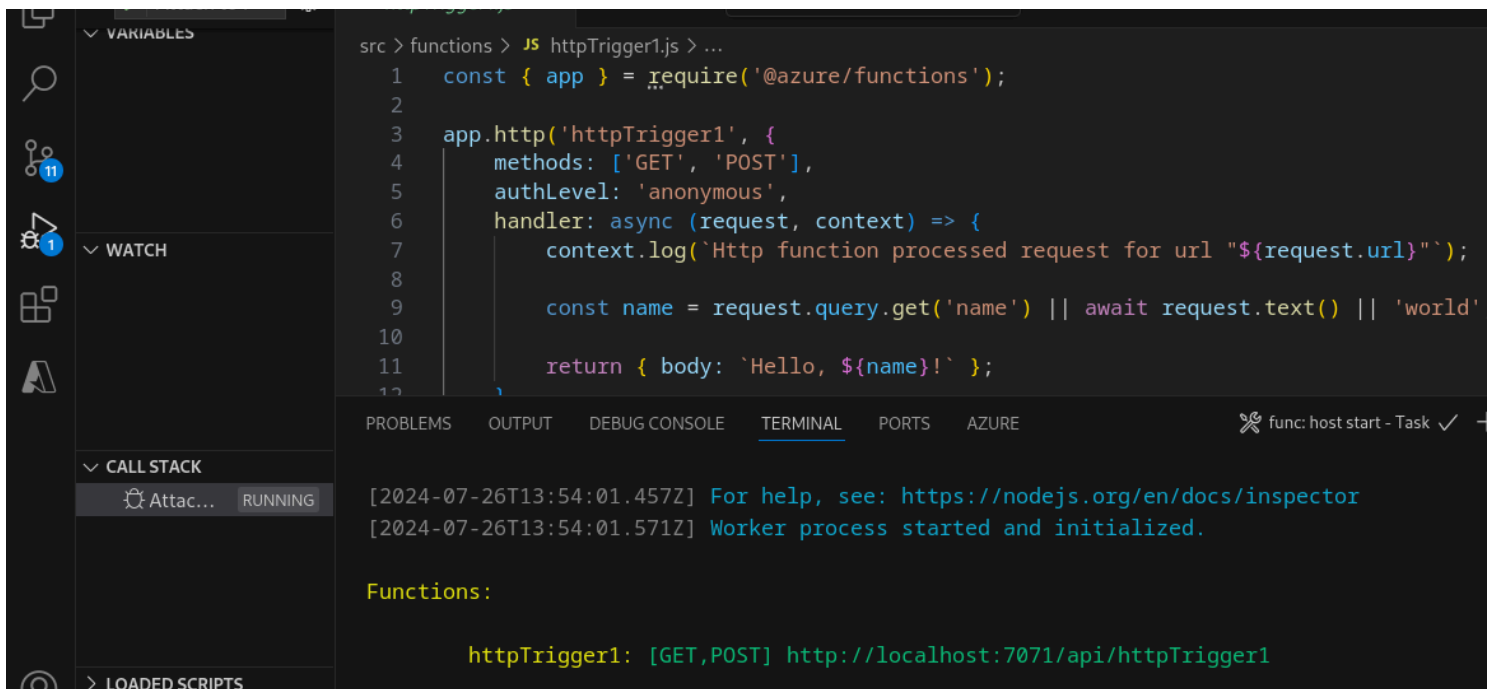
Use Local Emulator Cancelar Connect Storage Account



12. En caso de que se muestre la siguiente ventana, seleccionamos Debug anyway

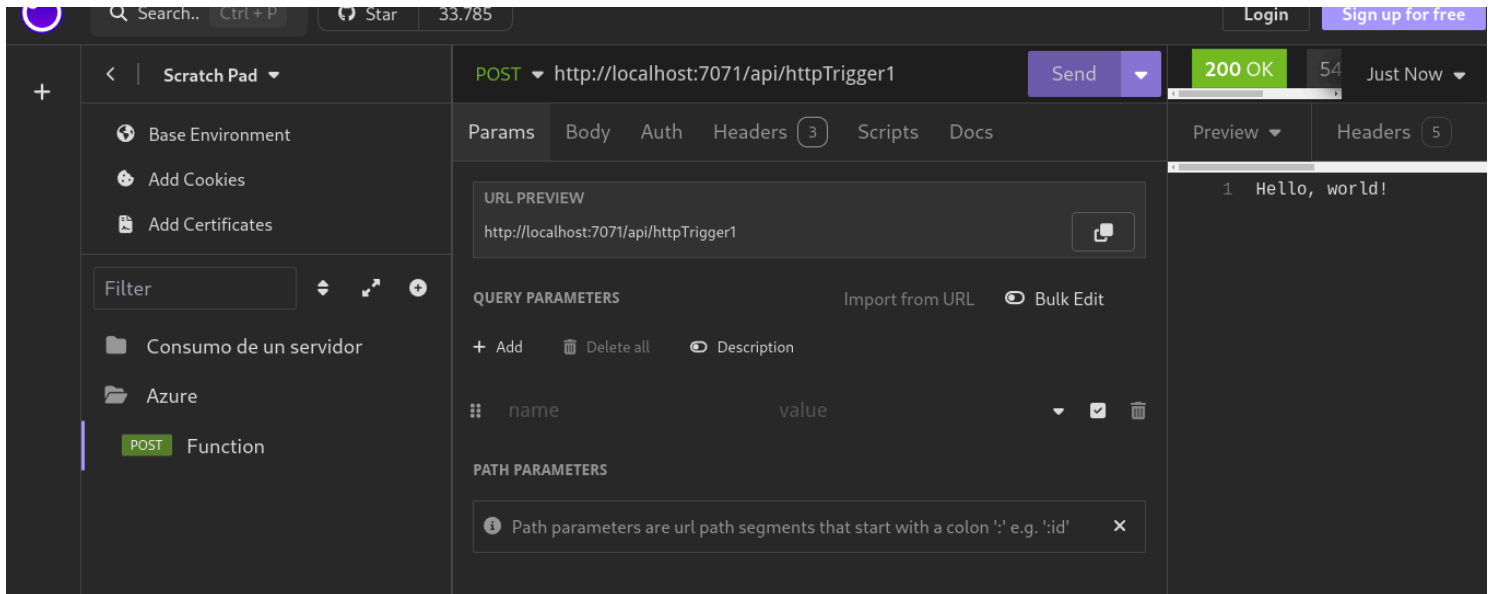


13. Nuestra función ahora se ejecuta localmente y muestra su URL local en la terminal



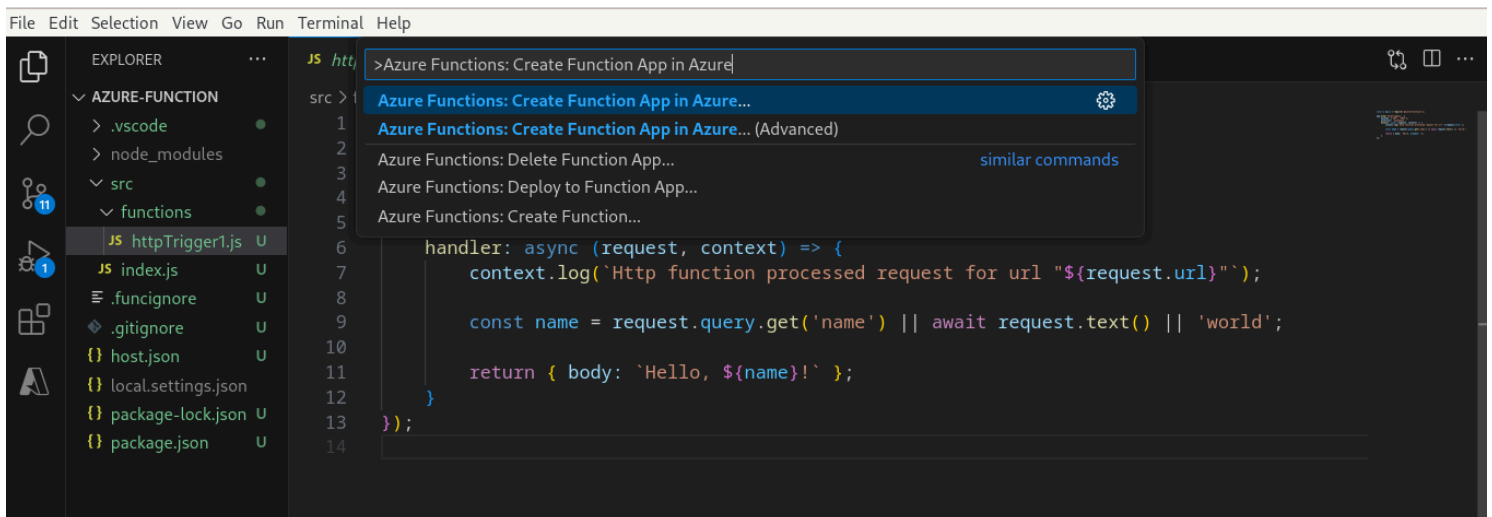


14. Si nos dirigimos a la URL de la función en local con nuestro cliente REST favorito, obtendremos el retorno de la función que en este caso es un “Hola, mundo!”



PARTE 2: CREACIÓN DE LA FUNCIÓN EN LA NUBE DE AZURE

1. Presionamos F1 y usamos el comando **Azure Functions: Create Function App in Azure**





2. ASignamos un nombre único a la función, este nombre identificará a nuestra función en la nube de Azure

File Edit Selection View Go Run Terminal Help

Create new Function App in Azure (1/3)

src > function-16542-adso

Enter a globally unique name for the new function app. (Press 'Enter' to confirm or 'Escape' to cancel)

```
methods: ['GET', 'POST'],
authLevel: 'anonymous',
handler: async (request, context) => {
  context.log(`Http function processed request for url "${request.url}"`);

  const name = request.query.get('name') || await request.text() || 'world';

  return { body: `Hello, ${name}!` };
};
```

3. Seleccionamos el entorno de ejecución

File Edit Selection View Go Run Terminal Help

Create new Function App in Azure (2/3)

src > Select a runtime stack.

Node.js 20 LTS (recently used)

Node.js 18 LTS

Change Azure Functions version Current: ~4

```
authLevel: 'anonymous',
handler: async (request, context) => {
  context.log(`Http function processed request for url "${request.url}"`);

  const name = request.query.get('name') || await request.text() || 'world';

  return { body: `Hello, ${name}!` };
};
```

4. Seleccionamos la región (preferiblemente más cercana a nuestra ubicación)

File Edit Selection View Go Run Terminal Help

Create new Function App in Azure (3/3)

src > Select a location for new resources.

East US 2 (recently used) Other

North Central US

West US

Japan West

West Central US

Australia Central

Australia Southeast

Korea South

South India

West India

```
st.url}"`);
) || 'world';
```



5. Esperamos que se cree nuestra función en la nube de Azure

```
src > functions > JS httpTrigger1.js > ...
1  const { app } = require('@azure/functions');
2
3  app.http('httpTrigger1', {
4    methods: ['GET', 'POST'],
5    authLevel: 'anonymous',
6    handler: async (request, context) => {
7      context.log(`Http function processed request for url "${request.url}"`);
8
9      const name = request.query.get('name') || await request.text() || 'world';
10
11      return { body: `Hello, ${name}!` };
12    }
13  });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

Create Function App "funcion-16542-adso" Creating storage account "funcion16542adso" in location "East US 2" with sku "Standard_LRS" ... (3/6)

6. Una vez creada la función en Azure, se nos notificará con Succeeded

```
src > functions > JS httpTrigger1.js > ...
1  const { app } = require('@azure/functions');
2
3  app.http('httpTrigger1', {
4    methods: ['GET', 'POST'],
5    authLevel: 'anonymous',
6    handler: async (request, context) => {
7      context.log(`Http function processed request for url "${request.url}"`);
8
9      const name = request.query.get('name') || await request.text() || 'world';
10
11      return { body: `Hello, ${name}!` };
12    }
13  });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

✓ Create Function App "funcion-16542-adso" Succeeded
Click to view resource

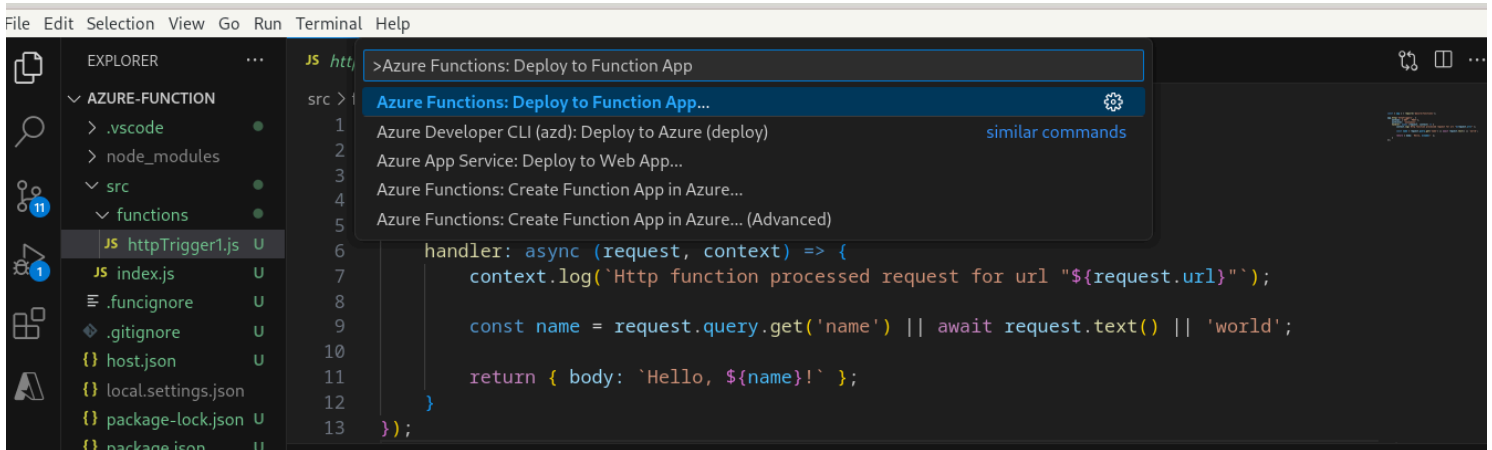
7. Si vamos a la plataforma de Azure, en la sección Aplicación de Funciones, veremos que nuestra función ha sido creada con éxito

Nombre	Estado	Ubicación	Plan de tarifa	Plan de App Service	Suscripción	Tipo de ...
funcion-16542-adso	En ejecución	East US 2	Dinámico	ASP-funcion-16542-adso-336d	Azure for Students	Function App

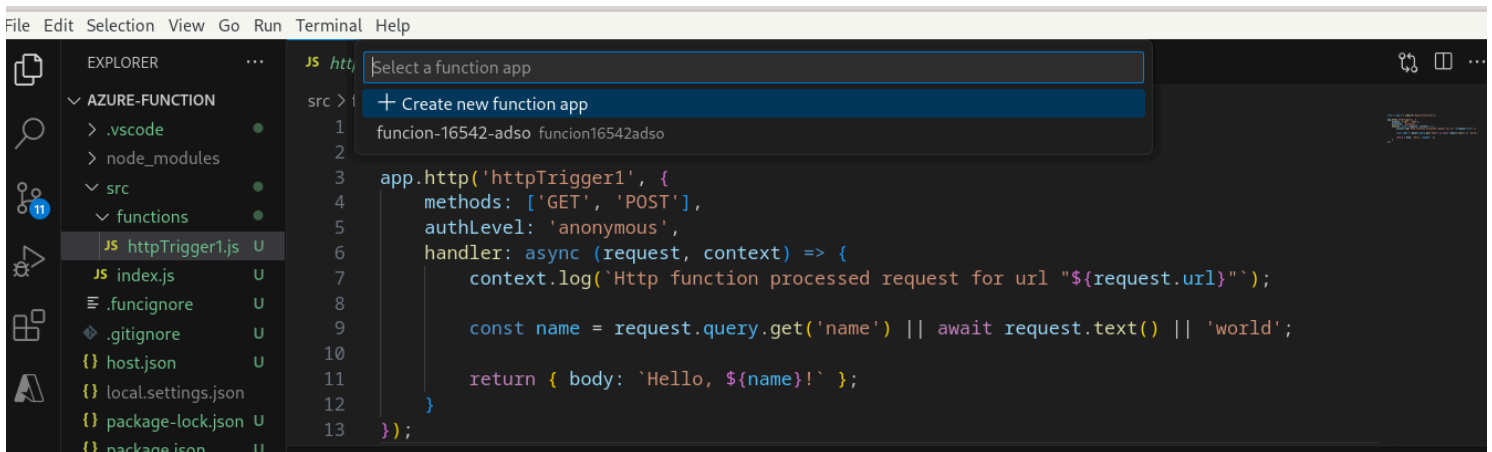


PARTE 3: DESPLIEGUE DE LA FUNCIÓN EN LA NUBE DE AZURE

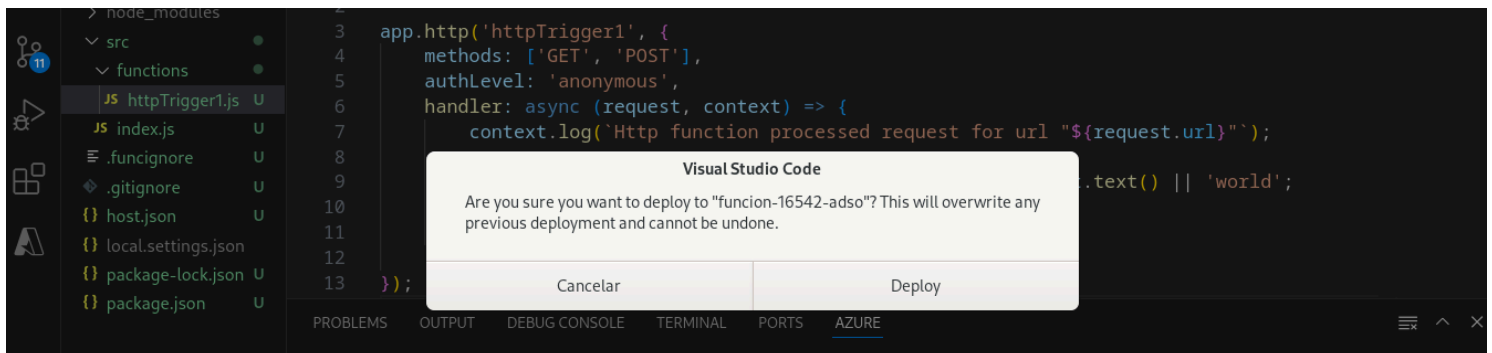
1. Presionamos F1 y ejecutamos el comando Azure Functions: Deploy to Function App



2. Seleccionamos la función creada anteriormente



3. En caso de mostrarse la siguiente ventana, hacemos click en Deploy





4. Cuando el despliegue esté hecho, se nos mostrará el siguiente mensaje en la parte inferior, damos click en View output

```
File Edit Selection View Go Run Terminal Help

EXPLORER
AZURE-FUNCTION
  .vscode
  node_modules
  src
    functions
      JS httpTrigger1.js U
      JS index.js U
  .funcignore U
  .gitignore U
  host.json U
  local.settings.json U
  package-lock.json U
  package.json U

src > functions > JS httpTrigger1.js > ...
1  const { app } = require('@azure/functions');
2
3  app.http('httpTrigger1', {
4    methods: ['GET', 'POST'],
5    authLevel: 'anonymous',
6    handler: async (request, context) => {
7      context.log(`Http function processed request for url "${request.url}"`);
8
9      const name = request.query.get('name') || await request.text() || 'world';
10
11      return { body: `Hello, ${name}!` };
12    }
13  });

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
* Executing task: npm install

up to date, audited 6 packages in 1s

found 0 vulnerabilities
* Terminal will be reused by tasks, press any key to restart

Deployment to "funcion-16542-adso" completed.
Source: Azure Functions Stream logs Upload settings View output
```

5. En la consola se mostrará la URL de la función en la internet

```
9:59:56 a. m. funcion-16542-adso: Preparing deployment for commit id 00d7127c0c...
9:59:57 a. m. funcion-16542-adso: Skipping build. Project type: Run-From-Zip
9:59:57 a. m. funcion-16542-adso: Skipping post build. Project type: Run-From-Zip
9:59:58 a. m. funcion-16542-adso: Triggering recycle (preview mode disabled).
9:59:58 a. m. funcion-16542-adso: Deployment successful.
10:00:07 a. m. funcion-16542-adso: Started postDeployTask "npm install (functions)".
10:00:23 a. m. funcion-16542-adso: Syncing triggers...
10:00:28 a. m. funcion-16542-adso: Querying triggers...
10:00:38 a. m. funcion-16542-adso: HTTP Trigger Urls:
httpTrigger1: https://funcion-16542-adso.azurewebsites.net/api/httptrigger1
```



Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2024
Instructor: Germán Alberto Angarita Henao



6. Si nos dirigimos a la URL de la función en internet con nuestro cliente REST favorito, obtendremos el retorno de la función que en este caso es un “Hola, mundo!”. Nuestro primer despliegue de una aplicación serverless ha sido terminado con éxito.

The screenshot shows a REST client interface with a dark theme. On the left is a sidebar with a search bar, a star icon, and a list of items including 'Scratch Pad', 'Base Environment', 'Add Cookies', 'Add Certificates', 'Consumo de un servidor', and 'Azure'. The 'POST Function' item is selected. The main area displays a POST request to 'https://funcion-16542-adso.azurewebsites.net/api/'. The 'Send' button is highlighted. The response status is '200 OK' and the body contains '1 Hello, world!'. Below the response, there are sections for 'QUERY PARAMETERS' and 'PATH PARAMETERS'. A tooltip at the bottom explains that path parameters are URL path segments starting with a colon, e.g., ':id'.