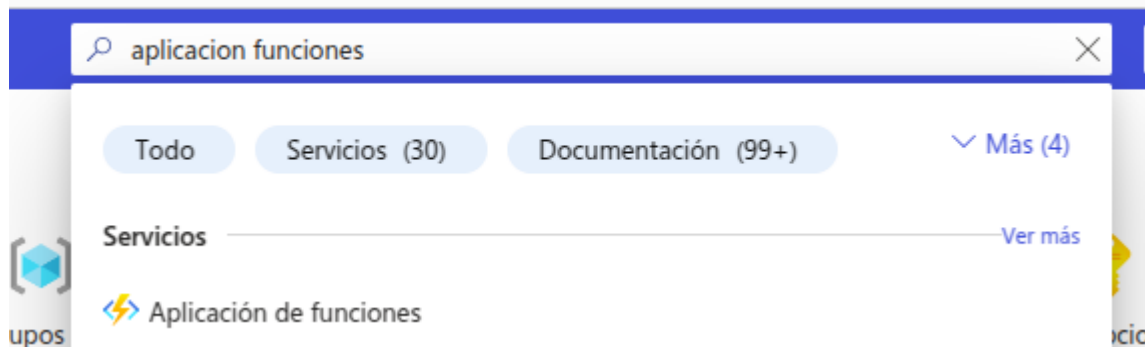


Configuración básica de Azure Functions

1. Buscar el servicio de Azure Functions.



2. Creamos un nuevo servicio.



3. Seleccionamos el plan "Consumo", que es el de plan mas sencillo y apto para la capa gratuita.

Planes de hospedaje	<input type="radio"/> Consumo flexible	<input checked="" type="radio"/> Consumo
	Obtenga alta escalabilidad con opciones de proceso, redes virtuales y facturación de pago por uso.	Pague por los recursos de proceso cuando se ejecuten las funciones (pago por uso).
Escalar a cero	✓	✓
Comportamiento de escalado	Controlado por eventos rápidos	Event-driven
Redes virtuales	✓	-
Proceso dedicado y prevención del arranque en frío	Opcional con Always Ready	-
Escalabilidad horizontal máxima (instancias)	1000	200

4. Se nos mostrará la siguiente ventana.

Datos básicos Storage Redes Supervisión Implementación Etiquetas Revisar y crear

Cree una aplicación de funciones, que permite agrupar funciones como una unidad lógica para facilitar la administración, la implementación y el uso compartido de los recursos. Fuctions permite ejecutar el código en un entorno sin servidor sin necesidad de crear primero una VM ni de publicar una aplicación web.

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * ⓘ

Grupo de recursos * ⓘ
[Crear nuevo](#)

Detalles de instancia

Nombre de la aplicación de funciones *
azurewebsites.net

☒ Probar un nombre de host predeterminado único seguro (versión preliminar). [Más información sobre esta actualización](#)

Pila del entorno en tiempo de ejecución *

Versión *

Región *

Sistema operativo * ☒ Linux ☐ Windows

5. Seleccionamos la suscripción que tenemos y creamos un nuevo Grupo de recurso para este servicio.

carpetas para organizar y administrar todos los recursos.

Suscripción * ⓘ

Grupo de recursos * ⓘ
[Crear nuevo](#)

Detalles de instancia

Nombre de la aplicación de funciones *
azurewebsites.net

Pila del entorno en tiempo de ejecución

Un grupo de recursos es un contenedor que tiene los recursos relacionados de una solución de Azure.

Nombre *

✓

Aceptar

Cancelar

6. Le colocamos el nombre a la instancia que crearemos, luego seleccionamos el entorno de ejecución y la versión que tendrá, para este ejemplo se usa Nodejs. Debemos seleccionar una región, puede ser cualquiera (de preferencia la misma que API Management si lo usamos). Y finalmente seleccionamos el sistema operativo, para este ejemplo dejaremos Windows.

Detalles de instancia

Nombre de la aplicación de funciones * ✓
 .azurewebsites.net

☐ Probar un nombre de host predeterminado único seguro (versión preliminar). [Más información sobre esta actualización](#)

Pila del entorno en tiempo de ejecución *

Versión *

Región *

Sistema operativo * ☐ Linux ☒ Windows

7. Las otras configuraciones podemos dejarlo como están por defecto si deseamos, solo debemos asegurarnos que en “Redes” esté habilitado el acceso público.

Datos básicos Storage **Redes** Supervisión Implementación Etiquetas Revisar y crear

Las Aplicaciones de funciones se pueden aprovisionar con la dirección de entrada pública a Internet o aislada en una red virtual de Azure. Las Aplicaciones de funciones también pueden aprovisionarse con el tráfico saliente capaz de llegar a los puntos de conexión de una red virtual, registrarse por grupos de seguridad de red o verse afectadas por rutas de red virtual. De forma predeterminada, la aplicación está abierta a Internet y no puede acceder a ninguna red virtual. Estos aspectos también se pueden cambiar una vez que se haya aprovisionado la aplicación. [Más información](#)

Habilitar el acceso público * ☐ ☒ Activado ☐ Desactivado

8. Las otras configuraciones podemos dejarlo como están por defecto si deseamos, por lo cual podemos darle directamente en “Revisar y Crear”.

Detalles

Suscripción	deb02758-46a5-4ed6-bded-5980e63093fb
Grupo de recursos	funciones
Nombre	funcionesCarga
Pila del entorno en tiempo de ejecución	Node.js 20 LTS

Hospedaje

Almacenamiento (nuevo)

Cuenta de Storage	funcionesac5f
-------------------	---------------

Plan (nuevo)

Opciones y planes de hospedaje	Consumo
Nombre	ASP-funciones-b863
Sistema operativo	Windows
Región	Central US
SKU	Dynamic

Supervisión (nuevo)

Application Insights	Habilitado
Nombre	funcionesCarga
Región	Central US

Implementación

Autenticación básica	Deshabilitado
Implementación continua	No habilitado/configurado tras crear la aplicación

9. Al momento de crearse se nos mostrará lo siguiente y damos click en “Ir a recurso”.

✓ Se completó la implementación



Nombre de implementación : Microsoft.Web-FunctionApp-Portal-89b1d997-8399

Suscripción : [MiSuscripción](#)

Grupo de recursos : [funciones](#)

> Detalles de implementación

✓ Pasos siguientes

[Ir al recurso](#)

10. Se nos mostrará una ventana con las especificaciones del servicio creado. Nos dirigimos a “Funciones” y damos click en “Crear Función”.

Creación de funciones en e



Crear en Azure Portal

Optimizado para:

- Introducción sin configuración local
- Elija entre nuestras plantillas de función

Crear función



Escritorio de VS Code

Optimizado para:

- Desarrollo local en VS Code
- Requisitos de la herramienta de c

Crear con VS Code escritorio

11. En la ventana que se nos muestra seleccionamos “HTTP” trigger y damos click en siguiente.

Crear función

1

Seleccionar una plantilla

2

Detalles de la plantilla

Use una plantilla para crear una función. Los desencadenadores describen el tipo de eventos que invocan las funciones. [Más información](#)

Nombre	Desencadenador
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in the body or query string

12. Le colocamos un nombre a la función y click en Crear.

1

Seleccionar una plantilla

2

Detalles de la plantilla

Plantilla de función

HTTP trigger

Nombre de función *

CargarImagen

Authorization level * ⓘ

Function

13. Se nos mostrará la función creada y damos click para configurarlo.

Funciones Métricas Propiedades Notificaciones (0)

+ Crear {} Configuración del entorno local Actualizar

Nombre
CargarImagen

14. Se nos mostrará el contenido de la función y donde podemos editar el código.

CargarImagen | Código y prueba ...

funcionesCarga

Código y prueba






Integración



Claves de función

Invocaciones

Registros

Métricas

 Guardar  Descartar  Actualizar  Prueba/ejecución  Obtener la dirección URL de la función ☐ Deshabilitar


 funcionesCarga / CargarImagen / index.js 

```
1 module.exports = async function (context, req) {
2   context.log('JavaScript HTTP trigger function processed a request.');
```

```
3
4   const name = (req.query.name || (req.body && req.body.name));
5   const responseMessage = name
6     ? "Hello, " + name + ". This HTTP triggered function executed successfully."
7     : "This HTTP triggered function executed successfully. Pass a name in the que
8
9   context.res = {
10     // status: 200, /* Defaults to 200 */
11     body: responseMessage
12   };
13 }
```

15. Podemos dar Click en “Obtener la dirección URL de la función” para probar el código de la función.

Integración Registros Métricas

ón  Obtener la dirección URL de la función ☐

Obtener dirección URL de función



_master (Clave de host)

<https://funcionescarga.azurewebsites.net/api/CargarImagen?code=nHMJ5ihNE2ssEHR...> 

default (Clave de host)

<https://funcionescarga.azurewebsites.net/api/CargarImagen?code=iJoYFVsXUT5ujShtA...> 

default (Tecla de función)

<https://funcionescarga.azurewebsites.net/api/CargarImagen?code=eSh3HeL4xKAPPe0...> 

16. Al ingresar a la URL se nos mostrará la respuesta de la función.

This HTTP triggered function executed successfully. Pass a name in the query string the request body for a personalized response.

17. Podemos dar click en “Prueba/Ejecución” para probar también.

Prueba/ejecución

Entrada Salida

Proporcione parámetros para probar la solicitud HTTP. Los resultados se pueden encontrar en la pestaña Salida.

Método HTTP *

Clave *

Parámetros de consulta

Nombre	Valor
<input type="text"/>	<input type="text"/>

Encabezados

Nombre	Valor
<input type="text"/>	<input type="text"/>

Cuerpo

```
1 {
2   "name": "Azure"
3 }
```

Prueba/ejecución

Entrada **Salida**

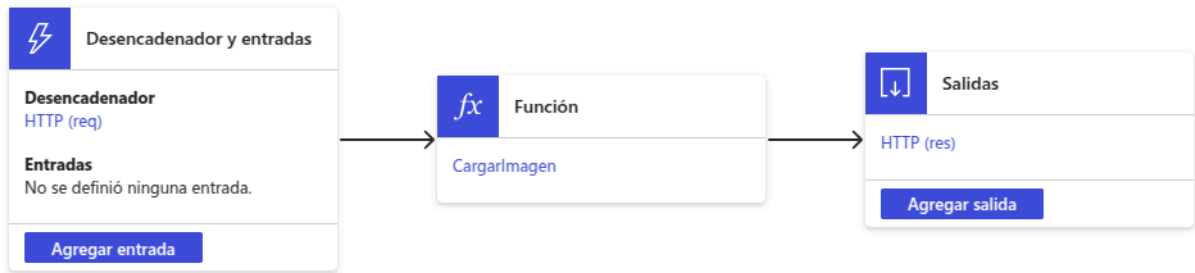
Código de respuesta HTTP

Contenido de respuesta HTTP

Hello, Azure. This HTTP triggered function executed successfully.

Conectar Azure Función con Blob Storage

18. Lo primero que haremos será conectar Blob Storage con nuestra función, para lo cual damos click en “Integración” y se nos mostrará lo siguiente.



19. Damos click en “Agregar Salida” y se nos mostrará lo siguiente.

Agregar salida ✕

Para empezar, seleccione el tipo de enlace de salida que desea agregar.

Tipo de enlace * ⓘ Azure Blob Storage ▼

Azure Blob Storage detalles

Storage account connection * ⓘ AzureWebJobsStorage ▼
[Nuevo](#)

Blob parameter name * ⓘ outputBlob

Path * ⓘ outcontainer/{rand-guid}

20. En tipo de enlace dejamos “Azure Blob Storage”. En “Storage account connection” seleccionamos nuevo y elegimos un Almacenamiento de Blob Storage que previamente debimos haber creado.

Para empezar, seleccione el tipo de enlace de salida que desea agregar.

Tipo de enlace * ⓘ Azure Blob Storage ▼

Azure Blob Storage detalles

Storage account connection * ⓘ AzureWebJobsStorage ▼
[Nuevo](#)

Blob parameter name * ⓘ

Path * ⓘ

Nueva conexión de la cuenta de almacenamiento

Cuenta de almacenamiento * conferenciasemis ▼

Aceptar Cancelar

21. En “Blob parameter name” podemos dejar el nombre por defecto (al dejar “outputBlob” interactuamos con el código con este mismo nombre para almacenar en Blob). En “Path” debemos colocar lo siguiente:

- a. Nombre del contenedor donde se guardarán las imágenes, si dicho contenedor no existe se creará de manera automática, pero sin permisos públicos. De preferencia crear un contenedor publico y colocar el nombre del contenedor. (fotos)
- b. Lo que se encuentra dentro de “{” “}” es el nombre con que se guardará la imagen, por defecto tiene “rand-guid”, el cual es un nombre aleatorio. Para este ejemplo colocamos “fileName”, el cual será un valor que vendrá desde el JSON que se envía desde el frontend, para ello se crea una variable en la función para almacenar el valor.

Blob parameter name * ⓘ	<input type="text" value="outputBlob"/>
Path * ⓘ	<input type="text" value="fotos/{fileName}"/>

22. Finalmente nos queda de esta forma y damos click en crear.

Agregar salida



Para empezar, seleccione el tipo de enlace de salida que desea agregar.

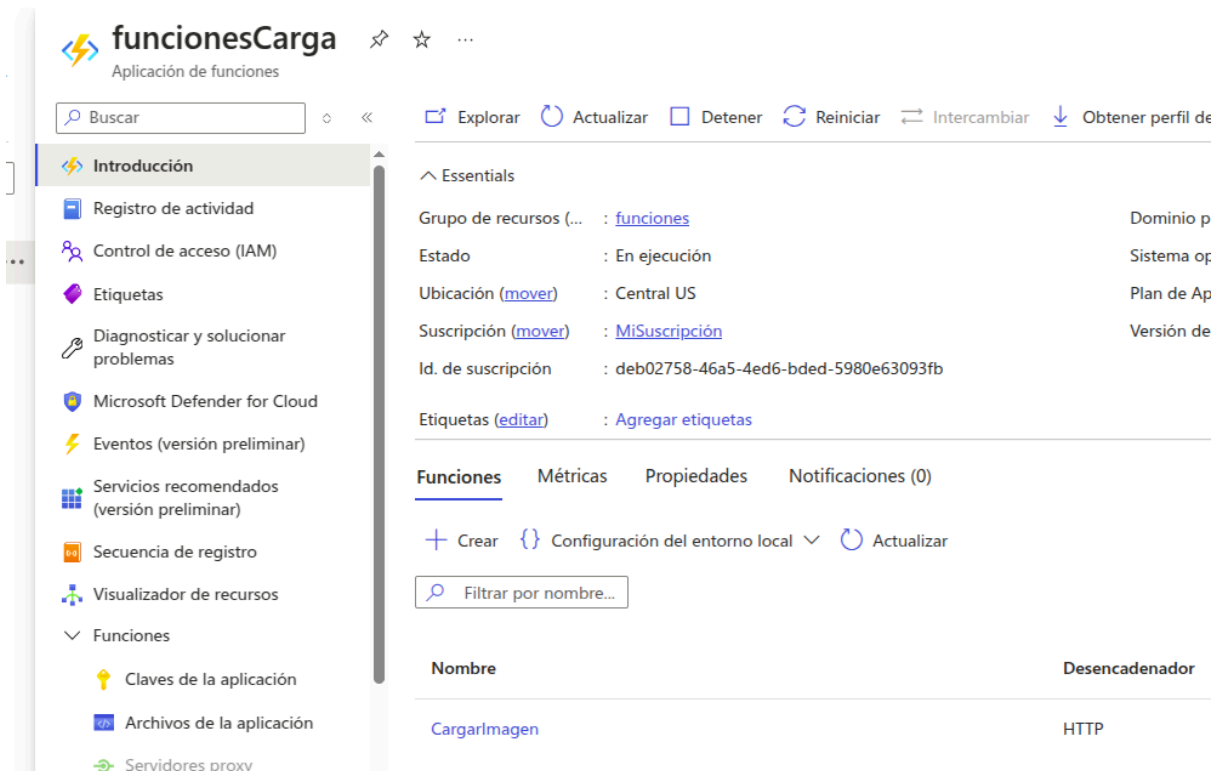
Tipo de enlace * ⓘ	<input type="text" value="Azure Blob Storage"/>
Azure Blob Storage detalles	
Storage account connection * ⓘ	<input type="text" value="conferenciasemis_STORAGE (new)"/>
	Nuevo
Blob parameter name * ⓘ	<input type="text" value="outputBlob"/>
Path * ⓘ	<input type="text" value="fotos/{fileName}"/>

23. Para este punto hemos establecido conexión para usar Blob Storage, esto lo haremos en el código interactuando con “outputBlob”, si es que no lo modificamos.

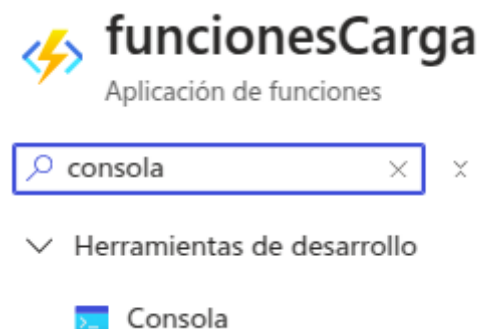
Instalar Librerías y crear Variables de Entorno en Azure Function

24. En ocasiones es necesario instalar librerías en nuestro entorno para poder interactuar con más comodidad, por ejemplo si necesitamos establecer una conexión con una base de datos externa a Azure.

25. Para conectarnos a una base de datos MYSQL de AWS, debemos instalar “mysql2”, por lo cual nos dirigimos a nuestro servicio de función.



26. Buscamos “Consola” e ingresamos.



27. Al ingresar a la consola nos dirigimos a la función que queremos instalar la librería y la instalamos.

```

C:\home\site\wwwroot>ls
CargarImagen
host.json

C:\home\site\wwwroot>cd CargarImagen
C:\home\site\wwwroot\CargarImagen>ls
function.json
index.js

C:\home\site\wwwroot\CargarImagen>npm init -y
Wrote to C:\home\site\wwwroot\CargarImagen\package.json:

{
  "name": "cargarimagen",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

```

```

C:\home\site\wwwroot\CargarImagen>ls
function.json
index.js
package.json

C:\home\site\wwwroot\CargarImagen>npm i mysql2
npm notice
npm notice New major version of npm available! 10.1.0 -> 11.2.0
npm notice Changelog: <https://github.com/npm/cli/releases/tag/v11.2.0>
npm notice Run `npm install -g npm@11.2.0` to update!
npm notice

added 13 packages, and audited 14 packages in 12s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\home\site\wwwroot\CargarImagen>_

```

28. Para agregar las variables de entorno buscamos “Variables de Entorno”.

 varia   

▼ Configuración





 Variables de entorno

 Configuración

29. Le damos en “Agregar” e ingresamos el valor, creamos las variables necesarias para conectar con la db. Y damos click en “Aplicar.”

Agregar o editar la configuración de la aplicación

Nombre *	<input type="text" value="DB_USER"/>
Valor	<input type="text" value="admin"/>
Configuración de ranura de implementación	<input type="checkbox"/>

DB_HOST	 Mostrar valor
DB_NAME	 Mostrar valor
DB_PASS	 Mostrar valor
DB_USER	 Mostrar valor

30. Con esto tenemos listo la librería “mysql2” y las variables de entorno para conectar con la DB.

31. Link de código de Ejemplo de carga de imagen en base 64 a Blob storage y almacenamiento en DB:
<https://github.com/Ayeser-Cristian/ConferenciaSemis1-1S2025/tree/main/server/azure>