**Creación de una aplicación de funciones en Azure Portal**

Las funciones se hospedan en un contexto de ejecución denominado **aplicación de funciones**. Puede definir aplicaciones de función para agrupar y estructurar de manera lógica las funciones y un recurso informático en Azure. En nuestro ejemplo de las escaleras mecánicas, crearía una aplicación de función para hospedar el servicio de temperatura del engranaje impulsor de las escaleras. Es necesario tomar algunas decisiones con el fin de crear la aplicación de función: deberá elegir un plan de servicio y seleccionar una cuenta de almacenamiento compatible.

**Selección de un plan de servicio**

Las aplicaciones de funciones pueden usar uno de los siguientes planes de hospedaje:

- Plan de consumo
- Plan Premium
- Plan dedicado (App Service)

Al usar la plataforma de aplicaciones sin servidor de Azure, elija el **plan de consumo**. El plan proporciona escalado automático y se factura solo cuando se ejecutan las funciones. El plan de consumo viene con un período de tiempo de espera configurable para ejecutar una función. De manera predeterminada es de 5 minutos, pero se puede configurar para tener un tiempo de espera de hasta 10 minutos.

El **plan Premium** también escala dinámicamente los recursos para satisfacer la demanda, pero puede especificar un número mínimo de instancias de máquina virtual (VM) para mantenerlas activas y reducir los llamados "arranques en frío". El plan Premium también permite que sus funciones se conecten a redes virtuales y se ejecuten dentro de ellas. Al igual que el plan Dedicado, el tiempo de espera predeterminado para las aplicaciones de un plan Premium es de 30 minutos, pero básicamente se pueden ejecutar durante un tiempo ilimitado (en función de la disponibilidad del servidor).

El **plan Dedicado (App Service)** le permite evitar períodos de tiempo de espera al ejecutar la función continuamente en una máquina virtual que defina. Cuando se usa un plan de App Service, usted es responsable de administrar los recursos de la aplicación en los que se ejecuta la función, por lo que técnicamente no se trata de un plan sin servidor. Pero puede ser una mejor opción si ya tiene un exceso de recursos de App Service disponibles en los que también ejecutar las funciones.

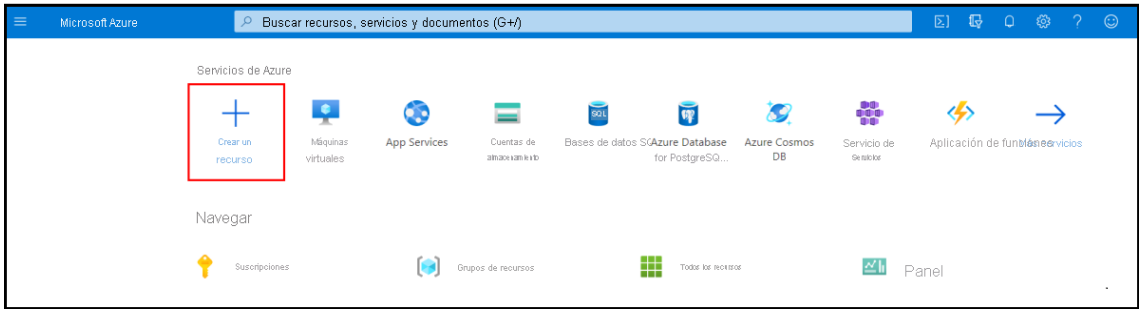
**Requisitos de la cuenta de almacenamiento**

Cuando se crea una aplicación de función, se debe vincular a una cuenta de almacenamiento. Puede seleccionar una cuenta existente o crear una. La aplicación de función usa esta cuenta de almacenamiento para las operaciones internas, como el registro de las ejecuciones de función y la administración de los desencadenadores de ejecución. En el **plan de consumo**, esta cuenta de almacenamiento también es donde se almacenan el código de función y el archivo de configuración.

**Creación de una aplicación de función**

Vamos a crear una aplicación de función en Azure Portal. La finalización de este módulo supone un pequeño coste en la cuenta de Azure.

1. Inicie sesión en Azure Portal con su cuenta de Azure.
2. En **Servicios de Azure**, seleccione **Crear un recurso**.




Aparece el panel **Crear un recurso** .

3. En el menú, busque y seleccione **Function App**. Seleccione el botón **Crear** . Aparece el panel **Crear aplicación de funciones** .
4. Seleccione el botón de radio **Consumo** y, a continuación, seleccione el botón **Seleccionar** .

| **Configuración** | **Valor** |
|---|---|
| **Detalles del proyecto** | |
| Suscripción | Su suscripción |
| Grupo de recursos | Seleccione el vínculo **Crear nuevo** y escriba un nombre para el grupo de recursos. |
| **Detalles de la instancia** | |
| Nombre de la aplicación de funciones | Escriba un nombre de aplicación único globalmente, que se convierte en parte de la dirección URL base del servicio. Por ejemplo, puede asignarle un nombre **escalator-functions-xxx**, donde puede reemplazar xxx por sus iniciales y un número. Los caracteres válidos son a-z, 0-9 y - |
| Pila en tiempo de ejecución | Node.js (lenguaje que se usa para implementar los ejemplos de la función de este ejercicio). |
| Versión | Aceptar *el valor predeterminado* |
| Región | Seleccione una ubicación geográfica próxima a usted. En un sistema de producción, se recomienda seleccionar una ubicación cercana a los clientes o consumidores de la función. |
| **Sistema operativo** | |

| Configuración | Valor |
|---|---|
| Sistema operativo | Aceptar *el valor predeterminado* |

5. En la pestaña **Aspectos básicos** , escriba los valores siguientes para cada configuración.
6. Seleccione **Siguiente: Almacenamiento**. En la pestaña **Almacenamiento** , escriba los valores siguientes para cada configuración.

| Configuración | Valor |
|---|---|
| **Almacenamiento** | |
| Cuenta de almacenamiento | Seleccione el vínculo **Crear nuevo** y escriba un nombre para la cuenta de almacenamiento y seleccione **Aceptar**. |

7. Seleccione **Revisar y crear** y, luego, **Crear**. La implementación tarda unos minutos. Cuando se complete la implementación, recibirá una notificación.

**Comprobación de la aplicación de función de Azure**

1. Cuando se complete la implementación, seleccione **Ir al recurso**. Aparece el panel Aplicación de funciones para la función escalator.
2. En la sección **Essentials** , seleccione el vínculo **Dominio predeterminado** para abrirlo en un explorador. Aparece una página web de Azure predeterminada con un mensaje que indica que la aplicación de Functions está en funcionamiento.

**Incorporación de lógica a la aplicación de funciones**

Vamos a continuar con el ejemplo de transmisión por engranaje y a agregar la lógica para el servicio de temperatura. En concreto, vamos a recibir datos de una solicitud HTTP.

**Requisitos de la función**

En primer lugar, es necesario definir algunos requisitos de la lógica:

- Las temperaturas de 0 a 25 grados deben marcarse como **OK**.
- Las temperaturas superiores a 25 hasta 50 grados deben marcarse como **PRECAUCIÓN**.
- Las temperaturas superiores a 50 grados deben marcarse como **PELIGRO**.

**Incorporación de una función a la aplicación de funciones**

Como se ha explicado en la unidad anterior, Azure proporciona plantillas que ayudan en la creación de funciones. En esta unidad se usa la plantilla HttpTrigger para implementar el servicio de temperatura.

1. En el ejercicio anterior, implementó la aplicación de funciones y la abrió. Si aún no está abierto, puede abrirlo desde la página Principal seleccionando **Todos los recursos** y, a continuación, seleccionando la aplicación de funciones, con un nombre similar a **escalator-functions-xxx**.

2. En la pantalla Aplicación de funciones, en la pestaña **Funciones** y **Crear en Azure Portal**, seleccione **Crear función**. Aparece el panel **Crear función** .
3. En **Seleccionar una plantilla**, seleccione *desencadenador HTTP* y seleccione **Siguiente**.
4. Deje el **nombre** de la función como *HttpTrigger1* y nivel de **autorización** como *Función* y seleccione **Crear**. La función **HttpTrigger1** se crea y se muestra en el panel Función **HttpTrigger1** .
5. Seleccione la pestaña **Código y prueba** . Se abre el editor de código, que muestra el contenido del archivo de código *index.js* para la función. El código predeterminado que la plantilla HTTP ha generado de forma automática aparece en el siguiente fragmento de código.

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


  const name = (req.query.name || (req.body && req.body.name));
  const responseMessage = name
    ? "Hello, " + name + ". This HTTP triggered function executed successfully."
    : "This HTTP triggered function executed successfully. Pass a name on the query string or in the request body for a personalized response.";


  context.res = {
    // status: 200, /* Defaults to 200 */
    body: responseMessage
  };
}
```

La función espera que se pase un nombre mediante la cadena de consulta de la solicitud HTTP o como parte del cuerpo de la solicitud. La función responde devolviendo el mensaje **Hello, <name>. Esta función desencadenada por HTTP se ejecutó correctamente**. Se devuelve el *nombre* que se envió en la solicitud.

En la lista desplegable archivo de origen, seleccione **function.json** para ver la configuración de la función, que debería tener un aspecto similar al código siguiente.

```
{
 "bindings": [
  {
   "authLevel": "function",
```

```
    "type": "httpTrigger",
    "direction": "in",
    "name": "req",
    "methods": [
      "get",
      "post"
    ]
  },
  {
    "type": "http",
    "direction": "out",
    "name": "res"
  }
 ]
}
```

Este archivo de configuración declara que la función se ejecuta cuando recibe una solicitud HTTP. El enlace de salida declara que la respuesta se envía como una respuesta HTTP.

**Probar la función**

**Sugerencia**

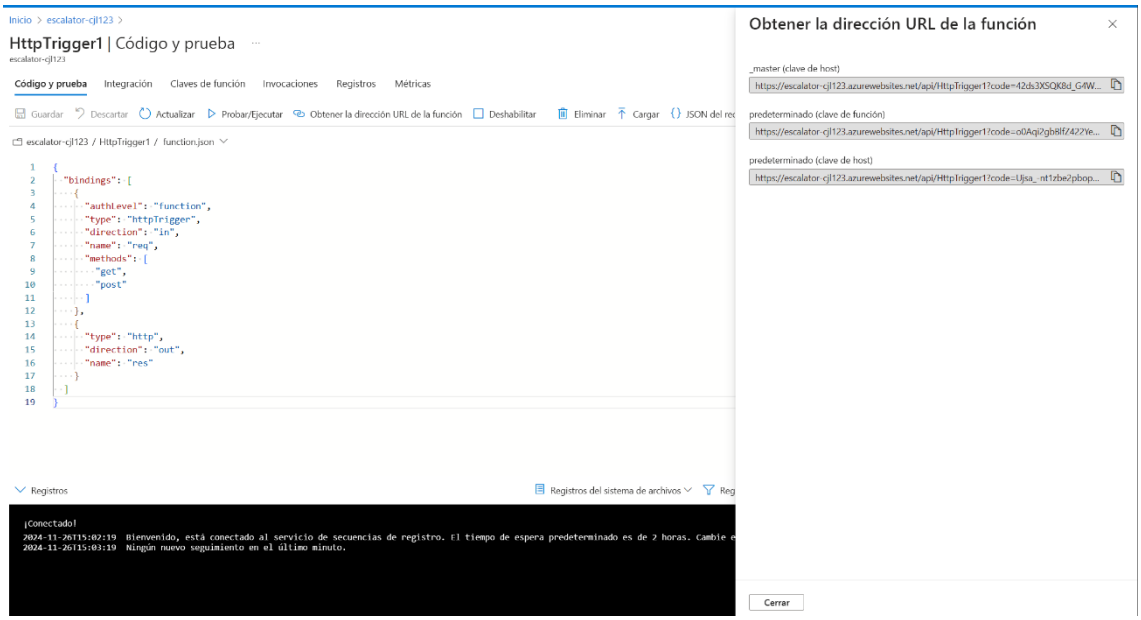
**cURL** es una herramienta de línea de comandos que se puede usar para enviar o recibir archivos. Se incluye con Linux, macOS y Windows 10 y puede descargarse para la mayoría de los demás sistemas operativos. cURL admite numerosos protocolos, como HTTP, HTTPS, FTP, FTPS, SFTP, LDAP, TELNET, SMTP, POP3, etc. Para obtener más información, consulte los vínculos siguientes:

- **https://en.wikipedia.org/wiki/CURL**
- **https://curl.haxx.se/docs/**

Para probar la función, puede enviar una solicitud HTTP a la dirección URL de la función con cURL en la línea de comandos.

1. Expanda el cuadro **Registros** de la parte inferior del panel de la función de desencadenador. Seleccione **Registros del sistema** de archivos en la lista desplegable de la parte superior del marco Registros. El cuadro Registros debe empezar a acumular notificaciones de seguimiento cada minuto.
2. Para buscar la dirección URL del punto de conexión de la función, en la barra de comandos, seleccione **Obtener dirección URL de la función**, como se muestra

en la siguiente imagen. Guarde el vínculo _**master (clave de host)** seleccionando el icono *Copiar en el Portapapeles* al final de la dirección URL. Almacene el vínculo en el Bloc de notas o en una aplicación similar para usarlo más adelante.




3. Abra un símbolo del sistema y ejecute cURL para enviar una solicitud HTTP a la dirección URL de la función. Recuerde usar la dirección URL que copió en el paso anterior.

curl "<your-https-url>"

**Sugerencia**

Es posible que tenga que encapsular la dirección URL entre comillas para evitar problemas con los caracteres especiales que pueda tener esta dirección.
Si usa Windows, ejecute cURL en el símbolo del sistema. PowerShell tiene un comando *curl* , pero es un alias para Invoke-WebRequest, que no es el mismo que cURL.

La respuesta debería ser similar a la siguiente.

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

Ahora pase un nombre en la solicitud. Para ello, debe agregar un parámetro de cadena de consulta denominado name a la dirección URL. En el ejemplo siguiente se agrega el parámetro name=Azure de cadena de consulta .

curl "<your-https-url>&name=Azure"

La respuesta debería ser similar a la siguiente.

Hello, Azure. This HTTP triggered function executed successfully.

La función se ha ejecutado correctamente y ha devuelto el nombre que ha pasado en la solicitud.

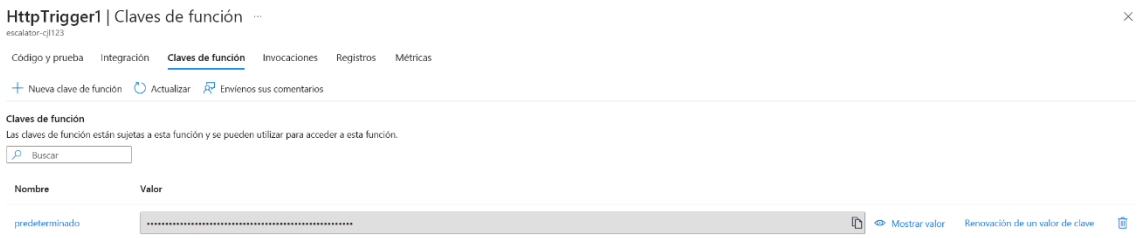
**Protección de desencadenadores de HTTP**

Los desencadenadores HTTP permiten usar claves de API para bloquear a los llamadores desconocidos pidiendo una clave como parte de la solicitud. Al crear una función, seleccione el nivel de *autorización*. De forma predeterminada, el nivel se establece en *Function*, que requiere una clave de API específica de función. También se puede establecer en *Administrador* para usar una clave "maestra" global o *Anónima* para indicar que no se requiere ninguna clave. También puede cambiar el nivel de autorización en las propiedades de la función tras su creación.

Dado que especificó *Function* al crear esta función, debe proporcionar la clave al enviar la solicitud HTTP. Puede enviarlo como un parámetro de cadena de consulta denominado code. O bien, use el método preferido y páselo como un encabezado HTTP denominado x-functions-key.

1. Para buscar las teclas de función, abra el menú **Código y prueba** seleccionando el nombre de la función (por ejemplo, *HttpTriger1*) en la pestaña **Funciones** del menú **Información general** . A continuación, seleccione la pestaña **Claves de función** .

2. De forma predeterminada, el valor de la clave de función está oculto. Para mostrar el valor predeterminado de la clave de función, seleccione **Mostrar valor**. Copie el contenido del campo **Valor** en el Portapapeles y, a continuación, almacene esta clave en el Bloc de notas o en una aplicación similar para su uso posterior.




3. Para probar la función con la clave de función, abra un símbolo del sistema y ejecute cURL para enviar una solicitud HTTP a la dirección URL de la función. Reemplace <your-function-key> por el valor de la clave de función que guardó y reemplace <your-https-url> por la dirección URL de la función.

curl --header "Content-Type: application/json" --header "x-functions-key: <your-function-key>" --request POST --data "{\"name\": \"Azure Function\"}" <your-https-url>

4. Revise el comando cURL y compruebe que tiene los siguientes valores:

    - Ha agregado un valor de encabezado Content-Type de tipo application/json.
    - Ha pasado la tecla de función como el valor de encabezado x-functions-key.
    - Ha usado una solicitud POST.
    - Se ha pasado la función de Azure con la dirección URL de la función.

5. Compruebe los registros.

El panel **Código y prueba** debe abrir una sesión que muestre la salida del archivo de registro (asegúrese de que **los registros del sistema** de archivos están seleccionados en

la lista desplegable de la parte superior del panel **Registros** ). El archivo de registro se actualiza con el estado de la solicitud, cuyo aspecto debe ser similar al siguiente:

2022-02-16T22:34:10.473 [Information] Executing 'Functions.HttpTrigger1' (Reason='This function was programmatically called via the host APIs.', Id=4f503b35-b944-455e-ba02-5205f9e8b47a)

2022-02-16T22:34:10.539 [Information] JavaScript HTTP trigger function processed a request.

2022-02-16T22:34:10.562 [Information] Executed 'Functions.HttpTrigger1' (Succeeded, Id=4f503b35-b944-455e-ba02-5205f9e8b47a, Duration=114ms)

**Adición de lógica de negocios a la función**

Agreguemos la lógica a la función, para comprobar las lecturas de temperatura que recibe y establecer un estado para cada una.

La función espera una matriz de lecturas de temperatura. El siguiente fragmento de código JSON es un ejemplo del cuerpo de la solicitud que enviaremos a nuestra función. El nombre de la matriz puede ser ligeramente diferente para JavaScript o PowerShell, pero cada entrada de la matriz tiene un identificador, marca de tiempo y temperatura.

```
{
  "Readings": [
    {
      "driveGearId": 1,
      "timestamp": 1534263995,
      "temperature": 23
    },
    {
      "driveGearId": 3,
      "timestamp": 1534264048,
      "temperature": 45
    },
    {
      "driveGearId": 18,
      "timestamp": 1534264050,
      "temperature": 55
    }
  ]
```

}

Reemplacemos el código predeterminado de la función por el siguiente código que implementa la lógica de negocios.

En el panel de funciones **HttpTrigger1** , abra el archivo **index.js** y reemplácelo por el código siguiente. Después de realizar este cambio, en la barra de comandos, seleccione **Guardar** para guardar las actualizaciones en el archivo.

```
module.exports = function (context, req) {
  context.log('Drive Gear Temperature Service triggered');
  if (req.body && req.body.readings) {
    req.body.readings.forEach(function(reading) {


      if(reading.temperature<=25) {
        reading.status = 'OK';
      } else if (reading.temperature<=50) {
        reading.status = 'CAUTION';
      } else {
        reading.status = 'DANGER'
      }
      context.log('Reading is ' + reading.status);
    });


    context.res = {
      // status: 200, /* Defaults to 200 */
      body: {
        "readings": req.body.readings
      }
    };
  }
  else {
    context.res = {
      status: 400,
      body: "Please send an array of readings in the request body"
```

```
    };
  }
  context.done();
};
```

La lógica que se agrega es sencilla. Recorremos en iteración la matriz y establecemos el estado como **CORRECTO**, **PRECAUCIÓN** o **PELIGRO** en función del valor del campo de temperatura. A continuación, devolvemos la matriz de lecturas con un campo de estado agregado a cada entrada.

Observe las instrucciones Log cuando expanda **Registros** en la parte inferior del panel. Cuando se ejecute la función, estas instrucciones agregarán los mensajes en la ventana Registros.

**Prueba de la lógica de negocios**

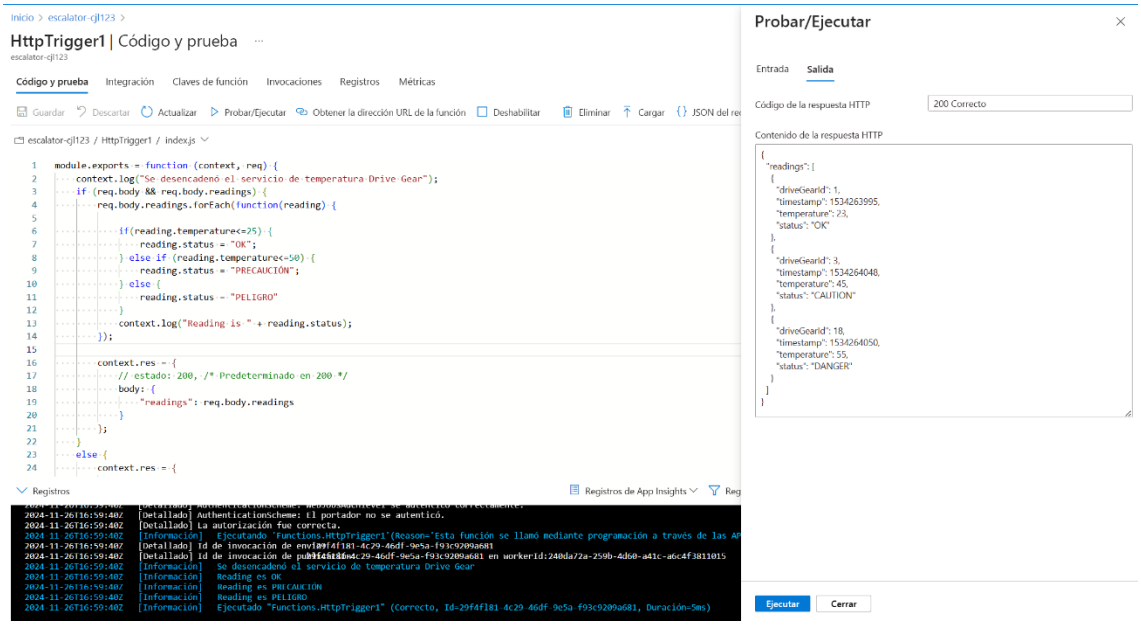
Vamos a usar la característica **Test/Run** en *Desarrollador>Código + Prueba* para probar nuestra función.

1. En la pestaña **Código y prueba** , seleccione **Probar/Ejecutar**. En la pestaña **Entrada**, reemplace el contenido del cuadro de texto **Cuerpo** por el código siguiente para crear la solicitud de ejemplo.

```
{
  "readings": [
    {
      "driveGearId": 1,
      "timestamp": 1534263995,
      "temperature": 23
    },
    {
      "driveGearId": 3,
      "timestamp": 1534264048,
      "temperature": 45
    },
    {
      "driveGearId": 18,
      "timestamp": 1534264050,
      "temperature": 55
    }
```

```
  ]
}
```

2. Seleccione **Ejecutar**. La pestaña **Salida** muestra el contenido y el código de respuesta HTTP. Para ver los mensajes de registro, abra la pestaña **Registros** en el menú desplegable del panel (si todavía no está abierto). En la imagen siguiente se muestra una respuesta de ejemplo en el panel de salida y los mensajes del panel **Registros** .




La pestaña **Salida** muestra que un campo de estado se agregó correctamente a cada una de las lecturas.