FunctionGraph

Pasos iniciales

Edición 01

Fecha 2023-05-08

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registradas y permisos

El logotipo HUAWEI y otras marcas registradas de Huawei pertenecen a Huawei Technologies Co., Ltd.
Todas las demás marcas registradas y los otros nombres comerciales mencionados en este documento son propiedad de sus respectivos titulares.

Aviso

Es posible que la totalidad o parte de los productos, las funcionalidades y/o los servicios que figuran en el presente documento no se encuentren dentro del alcance de un contrato vigente entre Huawei Cloud y el cliente. Las funcionalidades, los productos y los servicios adquiridos se limitan a los estipulados en el respectivo contrato. A menos que un contrato especifique lo contrario, ninguna de las afirmaciones, informaciones ni recomendaciones contenidas en el presente documento constituye garantía alguna, ni expresa ni implícita.

Huawei está permanentemente preocupada por la calidad de los contenidos de este documento; sin embargo, ninguna declaración, información ni recomendación aquí contenida constituye garantía alguna, ni expresa ni implícita. La información contenida en este documento se encuentra sujeta a cambios sin previo aviso.

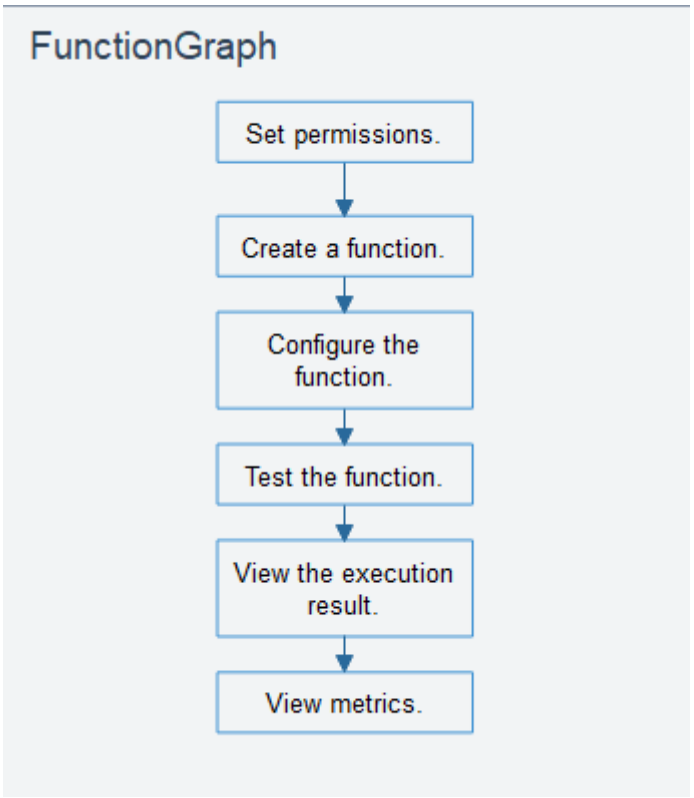
Índice

1 Introducción

Procedimiento general

FunctionGraph permite ejecutar código sin aprovisionar ni gestionar servidores, mientras garantiza altos niveles de disponibilidad y escalabilidad. Todo lo que se necesita hacer es cargar el código y establecer las condiciones de ejecución: FunctionGraph se encarga del resto. Usted paga solo por lo que usa y no se le cobra cuando su código no se está ejecutando.

Para crear rápidamente una función usando FunctionGraph haga lo siguiente:



1. Establecer permisos: Asegúrese de que tiene los permisos **FunctionGraph Administrator**.
2. Crear una función: Cree una función desde cero o usando el código de ejemplo o una imagen de contenedor.

3. Configurar la función: Configure la fuente del código o modificar otros parámetros.
4. Probar la función: Cree un evento de prueba para depurar la función.
5. Ver el resultado de la ejecución: En la página de detalles de la función, vea el resultado de la ejecución basado en el evento de prueba configurado.
6. Ver métricas: En la página de ficha **Monitoring** de la página de detalles de función, vea métricas de función.

2 Creación de una función desde cero

Introducción

Esta sección describe cómo crear y probar rápidamente una función de HelloWorld en la consola de FunctionGraph.

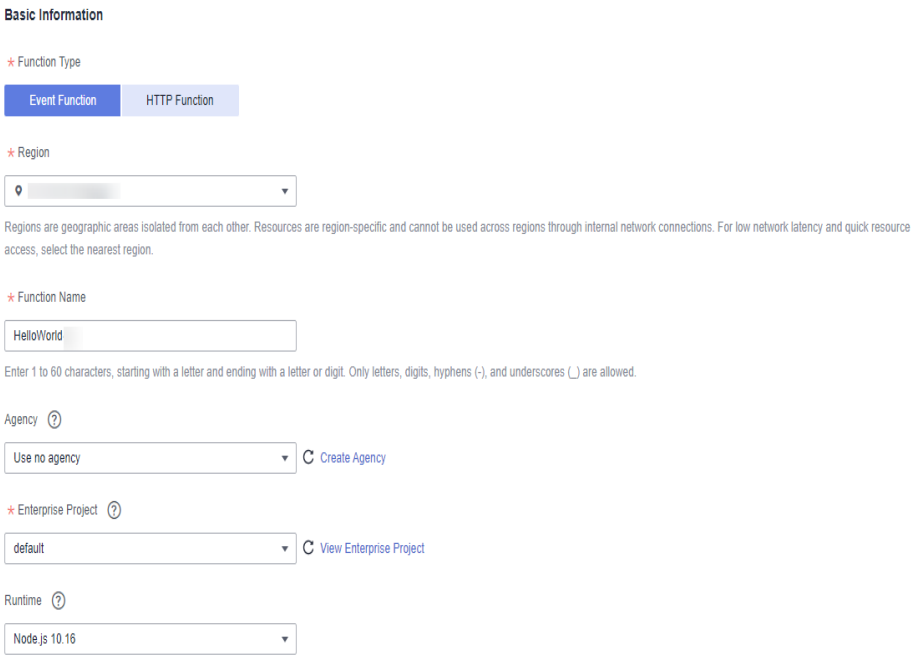
Paso 1: Preparar el entorno

Para realizar las operaciones descritas en esta sección, asegúrese de que tiene los permisos **FunctionGraph Administrator**, esto es, los permisos completos de FunctionGraph. Para obtener más información, vea la **Gestión de permisos**.

Paso 2: Crear una función

1. Inicie sesión en **la consola de FunctionGraph**. En el panel de navegación, seleccione **Functions** > **Function List**.
2. Haga clic en **Create Function** en la esquina superior derecha y elija **Create from scratch**.
3. En la página mostrada, establezca **Function Name** en **HelloWorld**, conserve los valores predeterminados para otros parámetros y haga clic en **Create Function**. Para obtener más información, véase **Figura 2-1**.

Figura 2-1 Configuración de información básica



4. Configure el código fuente, copie el siguiente código en la ventana de código y haga clic en **Deploy**.

   El código de ejemplo permite obtener eventos de prueba e imprimir información de eventos de prueba.

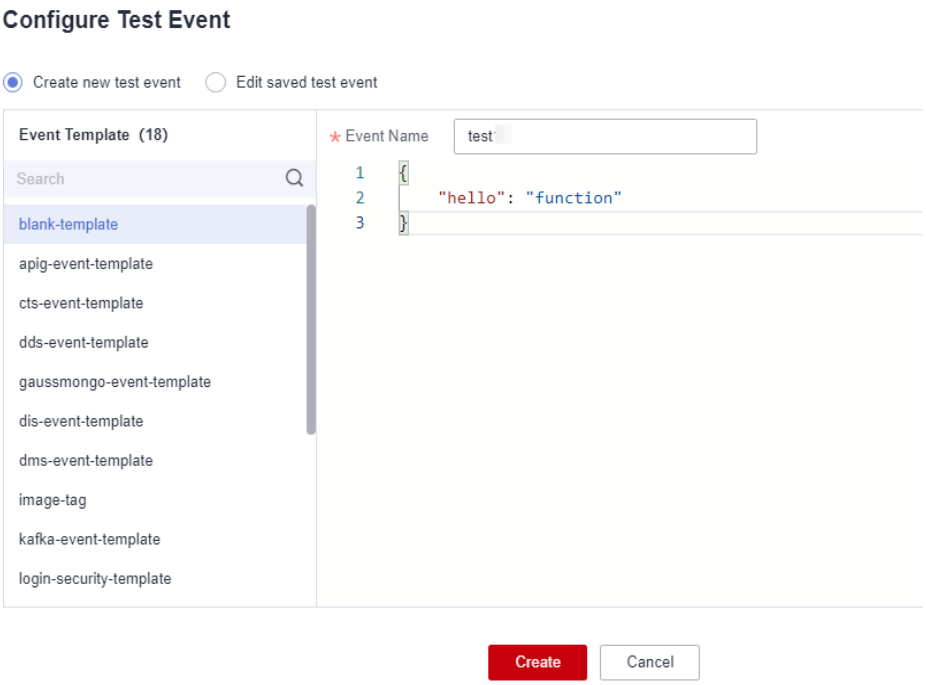
```
exports.handler = function (event, context, callback) {
    const error = null;
    const output = `Hello message: ${JSON.stringify(event)}`;
    callback(error, output);
}
```

Paso 3: Probar la función

1. En la página de detalles de la función, haga clic en **Test**. En el cuadro de diálogo que se muestra, cree un evento de prueba.
2. Seleccione **blank-template**, establezca **Event Name** en **test**, modifique el evento de prueba de la siguiente manera y haga clic en **Create**.

```
{
    "hello": "function"
}
```


Figura 2-2 Configuración de un evento de prueba

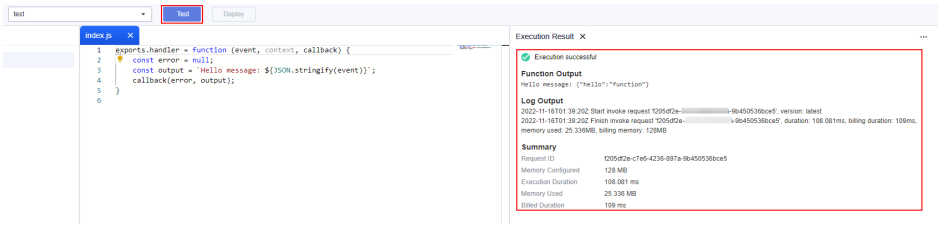


Paso 4: Ver el resultado de la ejecución

Haga clic en **Test** y vea el resultado de la ejecución a la derecha.

- **Function Output**: muestra el resultado de retorno de la función.
- **Log Output**: muestra los logs de ejecución de la función.
- **Summary**: muestra la información clave de los logs.

Figura 2-3 Consulta del resultado de la ejecución



NOTA

Se puede mostrar un máximo de 2 KB de logs. Para obtener más información de log, véase la **Consulta de logs de funciones**.

Paso 5: Ver métricas de monitoreo

En la página de detalles de la función, haga clic en la ficha **Monitoring**.

- En la página de ficha **Monitoring**, elija **Metrics** y seleccione un intervalo de tiempo (como 5 minutos, 15 minutos o 1 hora) para consultar la función.
- Se muestran las siguientes métricas: invocaciones, errores, duración (incluidas las duraciones máxima, media y mínima) y aceleración.

Paso 6: Eliminar una función

1. En la página de detalles de la función, elija **Operation** > **Delete function** en la esquina superior derecha.
2. En el cuadro de diálogo mostrado, haga clic en **OK** para liberar recursos.

3 Creación de una función mediante una plantilla

Introducción

FunctionGraph proporciona plantillas para completar automáticamente el código y ejecutar configuraciones de entorno cuando crea una función, lo que le ayuda a crear aplicaciones rápidamente.

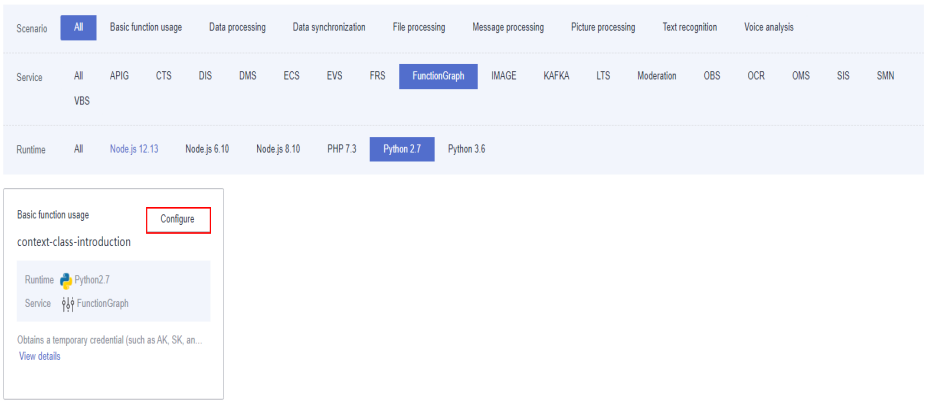
Paso 1: Preparar el entorno

Para realizar las operaciones descritas en esta sección, asegúrese de que tiene los permisos **FunctionGraph Administrator**, esto es, los permisos completos de FunctionGraph. Para obtener más información, vea la **Gestión de permisos**.

Paso 2: Crear una función

1. Inicie sesión en **la consola de FunctionGraph**. En el panel de navegación, seleccione **Functions** > **Function List**.
2. Haga clic en **Create Function** en la esquina superior derecha y elija **Select template**.
3. Seleccione la plantilla que se muestra en **Figura 3-1** y haga clic en **Configure**.

Figura 3-1 Selección de una plantilla



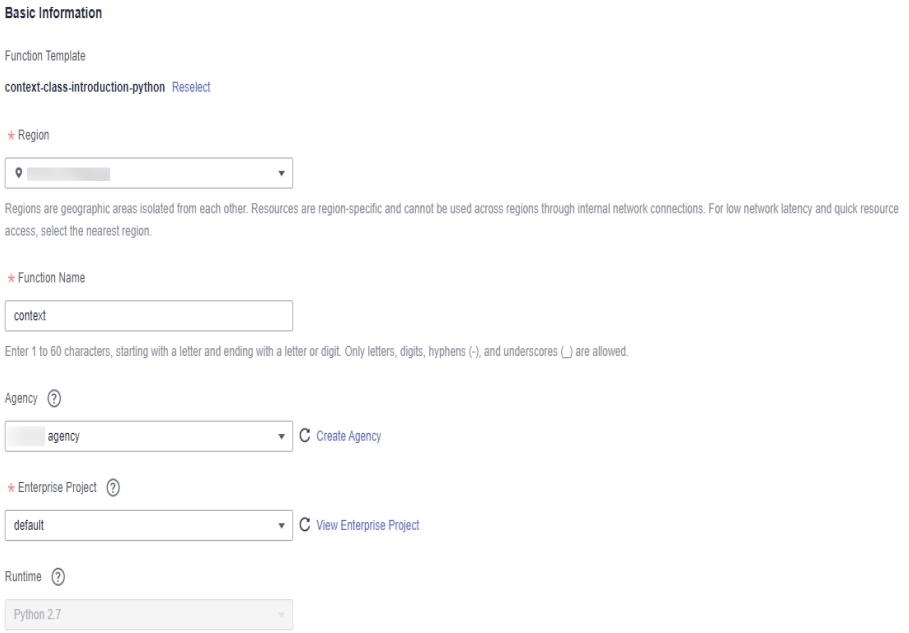
4. Establezca **Function Name** en **context** y seleccione cualquier delegación de la lista desplegable **Agency** y conserve los valores predeterminados para otros parámetros y haga clic en **Create Function**.

   NOTA

   Si no se configura ninguna delegación, se mostrará el siguiente mensaje cuando se active la función:

   ```
   Failed to access other services because no temporary AK, SK, or token has been obtained. Please set an agency.
   ```

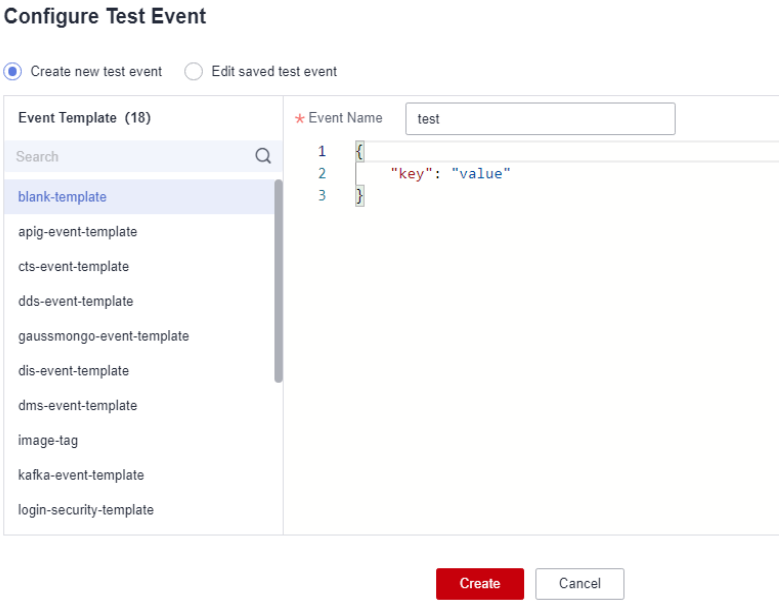
Figura 3-2 Configuración de información básica



Paso 3: Probar la función

1. En la página de detalles de la función, haga clic en **Test**. En el cuadro de diálogo que se muestra, cree un evento de prueba.
2. Seleccione **blank-template**, establezca **Event Name** en **test** y haga clic en **Create**.

Figura 3-3 Configuración de un evento de prueba



Paso 4: Ver el resultado de la ejecución

Haga clic en **Test** y vea el resultado de la ejecución a la derecha.

- **Function Output**: muestra el resultado de retorno de la función.
- **Log Output**: muestra los logs de ejecución de la función.
- **Summary**: muestra la información clave de los logs.

NOTA

Se puede mostrar un máximo de 2 KB de logs. Para obtener más información de log, véase la **Consulta de logs de funciones**.

Paso 5: Ver métricas de monitoreo

En la página de detalles de la función, haga clic en la ficha **Monitoring**.

- En la página de ficha **Monitoring**, elija **Metrics** y seleccione un intervalo de tiempo (como 5 minutos, 15 minutos o 1 hora) para consultar la función.
- Se muestran las siguientes métricas: invocaciones, errores, duración (incluidas las duraciones máxima, media y mínima) y aceleración.

Paso 6: Eliminar una función

1. En la página de detalles de la función, elija **Operation** > **Delete function** en la esquina superior derecha.
2. En el cuadro de diálogo mostrado, haga clic en **OK** para liberar recursos.

4 Despliegue de una función mediante una imagen de contenedor

4.1 Desarrollo de una función HTTP

Introducción

Cuando se desarrolla una función HTTP usando una imagen personalizada, despliegue un servidor HTTP en la imagen y escuche las solicitudes del puerto 8000. Las funciones HTTP solo admiten activadores de APIG.

Paso 1: Preparar el entorno

Para realizar las operaciones descritas en esta sección, asegúrese de que tiene los permisos **FunctionGraph Administrator**, esto es, los permisos completos de FunctionGraph. Para obtener más información, vea la **Gestión de permisos**.

Paso 2: Crear una imagen

Tome el SO Linux x86 de 64 bits como ejemplo.

1. Cree una carpeta.
   ```
   mkdir custom_container_http_example && cd custom_container_http_example
   ```
2. Despliegue un servidor HTTP. Node.js se utiliza como ejemplo. Para obtener más información sobre otros idiomas, consulte la sección **Creación de una función HTTP**.

   Cree el archivo **main.js** para introducir el marco de Express, recibir solicitudes de POST, imprimir el cuerpo de la solicitud como salida estándar y devolver "Hello FunctionGraph, method POST" al cliente.

   ```
   const express = require('express');
   const PORT = 8000;
   const app = express();
   app.use(express.json());
   app.post('/*', (req, res) => {
       console.log('receive', req.body);
       res.send('Hello FunctionGraph, method POST');
   });
   app.listen(PORT, () => {
   ```



```
  console.log(`Listening on http://localhost:${PORT}`);
});
```

3. Cree el archivo **package.json** para npm para que pueda identificar el proyecto y procesar las dependencias del proyecto.

```
{
  "name": "custom-container-http-example",
  "version": "1.0.0",
  "description": "An example of a custom container http function",
  "main": "main.js",
  "scripts": {},
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
      "express": "^4.17.1"
  }
}
```

   - **name**: Nombre del proyecto
   - **version**: Versión del proyecto
   - **main**: Archivo de entrada de aplicación
   - **dependencies**: Todas las dependencias disponibles del proyecto en npm

4. Cree un Dockerfile.

```
FROM node:12.10.0

ENV HOME=/home/custom_container
ENV GROUP_ID=1003
ENV GROUP_NAME=custom_container
ENV USER_ID=1003
ENV USER_NAME=custom_container

RUN mkdir -m 550 ${HOME} && groupadd -g ${GROUP_ID} ${GROUP_NAME} && useradd -
u ${USER_ID} -g ${GROUP_ID} ${USER_NAME}

COPY --chown=${USER_ID}:${GROUP_ID} main.js ${HOME}
COPY --chown=${USER_ID}:${GROUP_ID} package.json ${HOME}

RUN cd ${HOME} && npm install

RUN chown -R ${USER_ID}:${GROUP_ID} ${HOME}

RUN find ${HOME} -type d | xargs chmod 500
RUN find ${HOME} -type f | xargs chmod 500

USER ${USER_NAME}
WORKDIR ${HOME}

EXPOSE 8000
ENTRYPOINT ["node", "main.js"]
```

   - **FROM**: Especificar la imagen básica **node:12.10.0**. La imagen básica es obligatoria y su valor se puede cambiar.
   - **ENV**: Establece las variables de entorno **HOME** (**/home/custom_container**), **GROUP_NAME** y **USER_NAME** (**custom_container**), **USER_ID** y **GROUP_ID** (**1003**). Estas variables de entorno son obligatorias y sus valores pueden cambiarse.
   - **RUN**: Utilice el formato **RUN** <*Command*>. Por ejemplo, **RUN mkdir -m 550 ${HOME}**, que significa crear el directorio **home** para el usuario *${USER_NAME}* durante la construcción de contenedores.
   - **USER**: Cambie al usuario *${USER_NAME}*.
   - **WORKDIR**: Cambie el directorio de trabajo al directorio **home** del usuario *${USER_NAME}*.

- **COPY**: Copie **main.js** y **package.json** al directorio **home** del usuario *${USER_NAME}* en el contenedor.
- **EXPOSE**: Exponga el puerto 8000 del contenedor. No cambie este parámetro.
- **ENTRYPOINT**: Ejecute el comando **node main.js** para iniciar el contenedor. No cambie este parámetro.

NOTA

1. Puede utilizar cualquier imagen básica.
2. En el entorno de nube, UID 1003 y GID 1003 se usan para iniciar el contenedor por defecto. Los dos ID se pueden modificar seleccionando **Configuration** > **Basic Settings** > **Container Image Override** en la página de detalles de la función. No pueden ser **root** ni un ID reservado.

5. Construya una imagen.

   En el siguiente ejemplo, el nombre de la imagen es **custom_container_http_example** y la etiqueta **latest** y el punto (.) indica el directorio donde se encuentra el Dockerfile. Ejecute el comando de la construcción de image para empaquetar todos los archivos del directorio y envíe el paquete a un motor de contenedores para construir una imagen.

   ```
   docker build -t custom_container_http_example:latest .
   ```

Paso 3: Realizar la verificación local

1. Inicie el contenedor de Docker.
   ```
   docker run -u 1003:1003 -p 8000:8000 custom_container_http_example:latest
   ```
2. Abra una nueva ventana del símbolo del sistema y envíe un mensaje a través del puerto 8000 para acceder al directorio **/invoke** especificado en el código de plantilla.
   ```
   curl -XPOST -H 'Content-Type: application/json' -d '{"message":"HelloWorld"}'
   localhost:8000/helloworld
   ```
   Se devuelve la información siguiente basada en el código del módulo:
   ```
   Hello FunctionGraph, method POST
   ```
3. Compruebe si se muestra la siguiente información:
   ```
   receive {"message":"HelloWorld"}
   ```

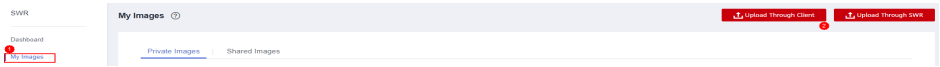
   ```
   [root@ecs-74d7 ~]# docker run -u 1003:1003 -p 8000:8000 custom_container_http_example:latest
   Listening on http://localhost:8000
   receive { message: 'HelloWorld' }
   ```

   También puede ejecutar el comando **docker logs** para obtener logs de contenedores.

   ```
   [root@ecs-74d7 custom_container_http_example]# docker logs 1354c3580638
   Listening on http://localhost:8000
   receive { message: 'HelloWorld' }
   [root@ecs-74d7 custom_container_http_example]#
   ```

Paso 4: Subir la imagen

1. Inicie sesión en la consola de SoftWare Repository for Container (SWR). En el panel de navegación, elija **My Images**.
2. Haga clic en **Upload Through Client** o **Upload Through SWR** en la esquina superior derecha.
3. Cargue la imagen como se le solicite.



4. Vea la imagen en la página **My Images**.

Paso 5: Crear una función

1. Inicie sesión en **la consola de FunctionGraph**. En el panel de navegación, seleccione **Functions** > **Function List**.
2. Haga clic en **Create Function** en la esquina superior derecha y elija **Select template**.
3. Establezca la información básica,
   - **Function Type**: Seleccione **HTTP Function**.
   - **Function Name**: Ingrese **custom_container_http**.
   - **Use container image**: Seleccione la imagen cargada a SWR. Ejemplo: **swr.***{Region ID}***.myhuaweicloud.com/***{Organization name}***/***{Image name}***:***{Image tag}*
   - **Agency**: Seleccione una delegación con el permiso del **SWR Admin**. Si no hay ninguna delegación disponible, cree una haciendo referencia a **Creación de una delegación**.
4. Una vez completada la configuración, haga clic en **Create Function**.

Paso 6: Probar la función

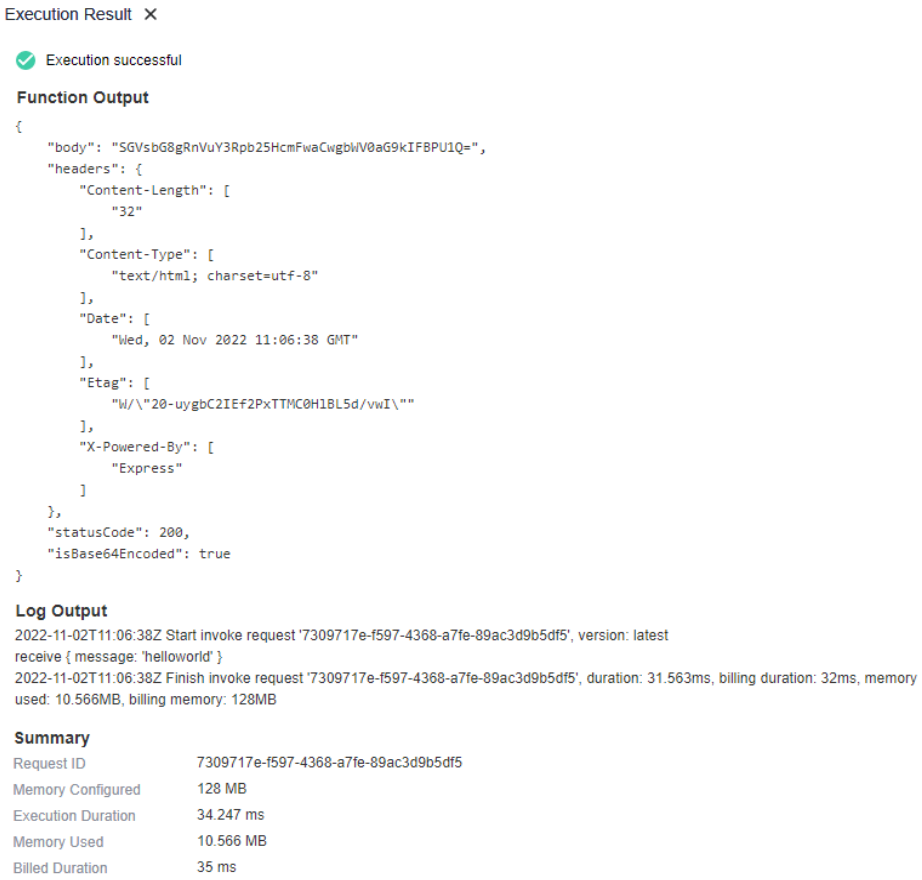
1. En la página de detalles de la función, haga clic en **Test**. En el cuadro de diálogo que se muestra, cree un evento de prueba.
2. Seleccione **apig-event-template**, establezca **Event Name** en **helloworld**, modifique el evento de prueba de la siguiente manera y haga clic en **Create**.

```
{
    "body": "{\"message\": \"helloworld\"}",
    "requestContext": {
        "requestId": "11cdcdcf33949dc6d722640a13091c77",
        "stage": "RELEASE"
    },
    "queryStringParameters": {
        "responseType": "html"
    },
    "httpMethod": "POST",
    "pathParameters": {},
    "headers": {
        "Content-Type": "application/json"
    },
    "path": "/helloworld",
    "isBase64Encoded": false
}
```

Paso 7: Ver el resultado de la ejecución

Haga clic en **Test** y vea el resultado de la ejecución a la derecha.

Figura 4-1 Resultado de la ejecución



- **Function Output**: muestra el resultado de retorno de la función.
- **Log Output**: muestra los logs de ejecución de la función.
- **Summary**: muestra la información clave de los logs.

  NOTA

  Se puede mostrar un máximo de 2 KB de logs. Para obtener más información de log, véase la **Consulta de logs de funciones**.

Paso 8: Ver métricas de monitoreo

En la página de detalles de la función, haga clic en la ficha **Monitoring**.

- En la página de ficha **Monitoring**, elija **Metrics** y seleccione un intervalo de tiempo (como 5 minutos, 15 minutos o 1 hora) para consultar la función.
- Se muestran las siguientes métricas: invocaciones, errores, duración (incluidas las duraciones máxima, media y mínima) y aceleración.

Paso 9: Eliminar la función

1. En la página de detalles de la función, elija **Operation** > **Delete function** en la esquina superior derecha.
2. En el cuadro de diálogo mostrado, haga clic en **OK** para liberar recursos.

4.2 Desarrollo de una función de evento

Introducción

Cuando se desarrolla una función de evento usando una imagen personalizada, despliegue un servidor HTTP en la imagen y escuche las solicitudes del puerto 8000. De forma predeterminada, la ruta de solicitud **/init** es la entrada de inicialización de función. Despliéguelo según sea necesario. Para obtener más información acerca de la inicialización de funciones, consulte **Iniciador**. La ruta de solicitud **/invoke** es la entrada de ejecución de función donde se procesan los eventos de activador. Para obtener más información acerca de los parámetros de solicitud, consulte **Fuentes de eventos compatibles**.

Paso 1: Preparar el entorno

Para realizar las operaciones descritas en esta sección, asegúrese de que tiene los permisos **FunctionGraph Administrator**, esto es, los permisos completos de FunctionGraph. Para obtener más información, vea la **Gestión de permisos**.

Paso 2: Crear una imagen

Tome el SO Linux x86 de 64 bits como ejemplo.

1. Cree una carpeta.
   ```
   mkdir custom_container_event_example && cd custom_container_evnet_example
   ```
2. Despliega un servidor HTTP para procesar solicitudes **init** e **invoke** y dar una respuesta. Node.js se utiliza como ejemplo.

   Cree el archivo **main.js** para introducir el marco de Express y despliegue un controlador de funciones (método **POST** y path **/invoke**) y un inicializador (método **POST** y path **/init**).

   ```
   const express = require('express');
   const PORT = 8000;
   const app = express();
   app.use(express.json());
   app.post('/init', (req, res) => {
     console.log('receive', req.body);
     res.send('Hello init\n');
   });
   app.post('/invoke', (req, res) => {
     console.log('receive', req.body);
     res.send('Hello invoke\n');
   });
   app.listen(PORT, () => {
     console.log(`Listening on http://localhost:${PORT}`);
   });
   ```
3. Cree el archivo **package.json** para npm para que pueda identificar el proyecto y procesar las dependencias del proyecto.
   ```
   {
     "name": "custom-container-event-example",
     "version": "1.0.0",
     "description": "An example of a custom container event function",
     "main": "main.js",
     "scripts": {},
     "keywords": [],
     "author": "",
     "license": "ISC",
   ```



```
  "dependencies": {
      "express": "^4.17.1"
  }
}
```

- **name**: Nombre del proyecto
- **version**: Versión del proyecto
- **main**: Archivo de entrada de aplicación
- **dependencies**: Todas las dependencias disponibles del proyecto en npm

4. Cree un Dockerfile.

```
FROM node:12.10.0

ENV HOME=/home/custom_container
ENV GROUP_ID=1003
ENV GROUP_NAME=custom_container
ENV USER_ID=1003
ENV USER_NAME=custom_container

RUN mkdir -m 550 ${HOME} && groupadd -g ${GROUP_ID} ${GROUP_NAME} && useradd -
u ${USER_ID} -g ${GROUP_ID} ${USER_NAME}

COPY --chown=${USER_ID}:${GROUP_ID} main.js ${HOME}
COPY --chown=${USER_ID}:${GROUP_ID} package.json ${HOME}

RUN cd ${HOME} && npm install

RUN chown -R ${USER_ID}:${GROUP_ID} ${HOME}

RUN find ${HOME} -type d | xargs chmod 500
RUN find ${HOME} -type f | xargs chmod 500

USER ${USER_NAME}
WORKDIR ${HOME}

EXPOSE 8000
ENTRYPOINT ["node", "main.js"]
```

- **FROM**: Especificar la imagen básica **node:12.10.0**. La imagen básica es obligatoria y su valor se puede cambiar.
- **ENV**: Establece las variables de entorno **HOME** (**/home/custom_container**), **GROUP_NAME** y **USER_NAME** (**custom_container**), **USER_ID** y **GROUP_ID** (**1003**). Estas variables de entorno son obligatorias y sus valores pueden cambiarse.
- **RUN**: Utilice el formato **RUN** <*Command*>. Por ejemplo, **RUN mkdir -m 550 ${HOME}**, que significa crear el directorio **home** para el usuario *${USER_NAME}* durante la construcción de contenedores.
- **USER**: Cambie al usuario *${USER_NAME}*.
- **WORKDIR**: Cambie el directorio de trabajo al directorio **home** del usuario *${USER_NAME}*.
- **COPY**: Copie **main.js** y **package.json** al directorio **home** del usuario *${USER_NAME}* en el contenedor.
- **EXPOSE**: Exponga el puerto 8000 del contenedor. No cambie este parámetro.
- **ENTRYPOINT**: Ejecute el comando **node /home/tester/main.js** para iniciar el contenedor.

NOTA

1. Puede utilizar cualquier imagen básica.
2. En el entorno de nube, UID 1003 y GID 1003 se usan para iniciar el contenedor por defecto. Los dos ID se pueden modificar seleccionando **Configuration** > **Basic Settings** > **Container Image Override** en la página de detalles de la función. No pueden ser **root** ni un ID reservado.

5. Construya una imagen.

   En el siguiente ejemplo, el nombre de la imagen es **custom_container_event_example** y la etiqueta **latest** y el punto (.) indica el directorio donde se encuentra el Dockerfile. Ejecute el comando de la construcción de image para empaquetar todos los archivos del directorio y envíe el paquete a un motor de contenedores para construir una imagen.

   ```
   docker build -t custom_container_event_example:latest .
   ```

Paso 3: Realizar la verificación local

1. Inicie el contenedor de Docker.
   ```
   docker run -u 1003:1003 -p 8000:8000 custom_container_event_example:latest
   ```
2. Abra una nueva ventana del símbolo del sistema y envíe un mensaje a través del puerto 8000 para acceder al directorio **/init** especificado en el código de plantilla.
   ```
   curl -XPOST -H 'Content-Type: application/json' localhost:8000/init
   ```
   Se devuelve la información siguiente basada en el código del módulo:
   ```
   Hello init
   ```
3. Abra una nueva ventana del símbolo del sistema y envíe un mensaje a través del puerto 8000 para acceder al directorio **/invoke** especificado en el código de plantilla.
   ```
   curl -XPOST -H 'Content-Type: application/json' -d '{"message":"HelloWorld"}'
   localhost:8000/invoke
   ```
   Se devuelve la información siguiente basada en el código del módulo:
   ```
   Hello invoke
   ```
4. Compruebe si se muestra la siguiente información:
   ```
   Listening on http://localhost:8000
   receive {}
   receive { message: 'HelloWorld' }
   ```

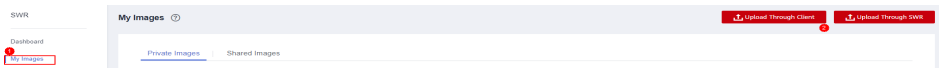
   ```
   [root@ecs-74d7 ~]# docker run -u 1003:1003 -p 8000:8000 custom_container_event_example:latest
   Listening on http://localhost:8000
   receive {}
   receive { message: 'HelloWorld' }
   ```

   También puede ejecutar el comando **docker logs** para obtener logs de contenedores.

   ```
   [root@ecs-74d7 custom_container_event_example]# docker logs 5560e1ec09d3
   Listening on http://localhost:8000
   receive {}
   receive { message: 'HelloWorld' }
   [root@ecs-74d7 custom_container_event_example]#
   ```

Paso 4: Subir la imagen

1. Inicie sesión en la consola de SWR. En el panel de navegación, elija **My Images**.
2. Haga clic en **Upload Through Client** o **Upload Through SWR** en la esquina superior derecha.
3. Cargue la imagen como se le solicite.



4. Vea la imagen en la página **My Images**.

Paso 5: Crear una función

1. Inicie sesión en **la consola de FunctionGraph**. En el panel de navegación, seleccione **Functions** > **Function List**.
2. Haga clic en **Create Function** en la esquina superior derecha y elija **Select template**.
3. Establezca la información básica,
   - **Function Type**: Seleccione **Event Function**.
   - **Function Name**: Ingrese **custom_container_event**.
   - **Use container image**: Seleccione la imagen cargada a SWR. Ejemplo: **swr.***{Region ID}***.myhuaweicloud.com/***{Organization name}***/***{Image name}***:***{Image tag}*
   - **Agency**: Seleccione una delegación con el permiso del **SWR Admin**. Si no hay ninguna delegación disponible, cree una haciendo referencia a **Creación de una delegación**.
4. Una vez completada la configuración, haga clic en **Create Function**.
5. En la página de detalles de la función, seleccione **Configuration** > **Advanced Settings** y habilite **Initialization**. Se invocará a la API **init** para inicializar la función.

Paso 6: Probar la función

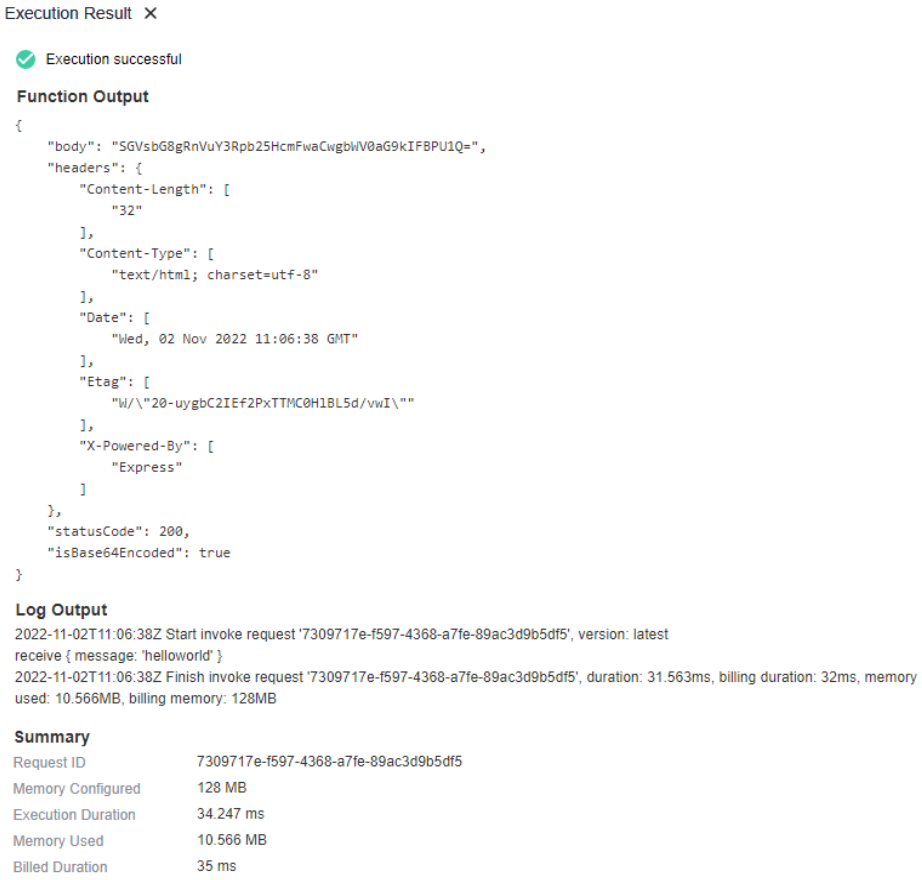
1. En la página de detalles de la función, haga clic en **Test**. En el cuadro de diálogo que se muestra, cree un evento de prueba.
2. Seleccione **blank-template**, establezca **Event Name** en **helloworld**, modifique el evento de prueba de la siguiente manera y haga clic en **Create**.

```
{
    "message": "HelloWorld"
}
```

Paso 7: Ver el resultado de la ejecución

Haga clic en **Test** y vea el resultado de la ejecución a la derecha.

Figura 4-2 Resultado de la ejecución



- **Function Output**: muestra el resultado de retorno de la función.
- **Log Output**: muestra los logs de ejecución de la función.
- **Summary**: muestra la información clave de los logs.

  NOTA

  Se puede mostrar un máximo de 2 KB de logs. Para obtener más información de log, véase la **Consulta de logs de funciones**.

Paso 8: Ver métricas de monitoreo

En la página de detalles de la función, haga clic en la ficha **Monitoring**.

- En la página de ficha **Monitoring**, elija **Metrics** y seleccione un intervalo de tiempo (como 5 minutos, 15 minutos o 1 hora) para consultar la función.
- Se muestran las siguientes métricas: invocaciones, errores, duración (incluidas las duraciones máxima, media y mínima) y aceleración.

Paso 9: Eliminar la función

1. En la página de detalles de la función, elija **Operation** > **Delete function** en la esquina superior derecha.
2. En el cuadro de diálogo mostrado, haga clic en **OK** para liberar recursos.

