

FunctionGraph

Primeiros passos

Edição 01
Data 2025-08-25



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. Todos os direitos reservados.

Nenhuma parte deste documento pode ser reproduzida ou transmitida em qualquer forma ou por qualquer meio sem consentimento prévio por escrito da Huawei Cloud Computing Technologies Co., Ltd.

Marcas registradas e permissões



HUAWEI e outras marcas registradas da Huawei são marcas registradas da Huawei Technologies Co., Ltd. Todas as outras marcas registradas e os nomes registrados mencionados neste documento são propriedade dos seus respectivos detentores.

Aviso

Os produtos, os serviços e as funcionalidades adquiridos são estipulados pelo contrato estabelecido entre a Huawei Cloud e o cliente. Os produtos, os serviços e as funcionalidades descritos neste documento, no todo ou em parte, podem não estar dentro do âmbito de aquisição ou do âmbito de uso. Salvo especificação em contrário no contrato, todas as declarações, informações e recomendações neste documento são fornecidas "TAL COMO ESTÃO" sem garantias ou representações de qualquer tipo, sejam expressas ou implícitas.

As informações contidas neste documento estão sujeitas a alterações sem aviso prévio. Foram feitos todos os esforços na preparação deste documento para assegurar a exatidão do conteúdo, mas todas as declarações, informações e recomendações contidas neste documento não constituem uma garantia de qualquer tipo, expressa ou implícita.

Índice

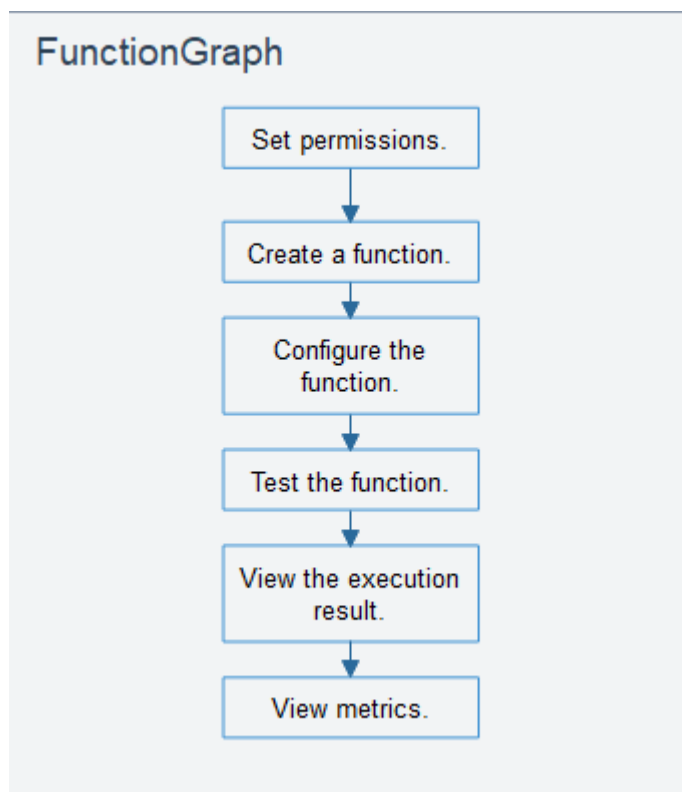
1 Introdução.....	1
2 Criação de uma função a partir do rascunho.....	3
3 Criação de uma função usando um modelo.....	7
4 Implementação de uma função usando uma imagem de contêiner.....	10
4.1 Desenvolvimento de uma função HTTP.....	10
4.2 Desenvolvimento de uma função de evento.....	15

1 Introdução

Procedimento geral

FunctionGraph permite que você execute seu código sem provisionar nem gerenciar servidores e garantindo alta disponibilidade e escalabilidade. Tudo o que você precisa fazer é enviar seu código e definir as condições de execução, e o FunctionGraph cuidará do resto. Você paga apenas pelo que usa e não é cobrado quando o seu código não está em execução.

Para criar rapidamente uma função usando o FunctionGraph faça o seguinte:



1. Definir permissão: verifique se você tem as permissões **FunctionGraph Administrator**.
2. Criar uma função: crie uma função do zero ou usando o código de amostra ou uma imagem de contêiner.

3. Configurar a função: configure a fonte do código ou modifique outros parâmetros.
4. Testar a função: crie um evento de teste para depurar a função.
5. Ver o resultado da execução: na página de detalhes da função, visualize o resultado da execução com base no evento de teste configurado.
6. Exibir métricas: na página de guia **Monitoring** da página de detalhes da função, exiba as métricas da função.

2 Criação de uma função a partir do rascunho

Introdução

Esta seção descreve como criar e testar rapidamente uma função HelloWorld no console do FunctionGraph.

Passo 1: preparar o ambiente

Para executar as operações descritas nesta seção, verifique se você tem as permissões de **FunctionGraph Administrator**, ou seja, as permissões completas para o FunctionGraph. Para obter mais informações, consulte [Gerenciamento de permissões](#).

Passo 2: criar uma função

1. Efetue login no [console do FunctionGraph](#). No painel de navegação, escolha **Functions > Function List**.
2. Clique em **Create Function** no canto superior direito e escolha **Create from scratch**.
3. Na página exibida, defina **Function Name** como **HelloWorld** e mantenha os valores padrão para outros parâmetros e clique em **Create Function**. Para mais detalhes, consulte [Figura 2-1](#).

Figura 2-1 Configurar informações básicas

Basic Information

* Function Type

Event Function HTTP Function

* Region

▼

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

* Function Name

HelloWorld

Enter 1 to 60 characters, starting with a letter and ending with a letter or digit. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Agency ?

Use no agency ▼ [Create Agency](#)

* Enterprise Project ?

default ▼ [View Enterprise Project](#)

Runtime ?

Node.js 10.16 ▼

4. Configure a origem do código, copie o código a seguir para a janela de código e clique em **Deploy**.

O código de exemplo permite obter eventos de teste e imprimir informações de eventos de teste.

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

Passo 3: testar a função

1. Na página de detalhes da função, clique em **Test**. Na caixa de diálogo exibida, crie um evento de teste.
2. Selecione **blank-template**, defina **Event Name** para **test**, modifique o evento de teste da seguinte forma e clique em **Create**.

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

Figura 2-2 Configurar um evento de teste

Configure Test Event

☒ Create new test event ☐ Edit saved test event

Event Template (18)

Search

- blank-template
- apig-event-template
- cts-event-template
- dds-event-template
- gaussmongo-event-template
- dis-event-template
- dms-event-template
- image-tag
- kafka-event-template
- login-security-template

★ Event Name

```

1  {
2    "hello": "function"
3  }
```

Passo 4: ver o resultado da execução.

Clique em **Test** e visualize o resultado da execução à direita.

- **Function Output:** exibe o resultado do retorno da função.
- **Log Output:** exibe os logs de execução da função.
- **Summary:** exibe informações-chave dos logs.

Figura 2-3 Visualização do resultado da execução

test

```

1 exports.handler = function (event, context, callback) {
2   const error = null;
3   const output = 'hello message: ' + JSON.stringify(event);
4   callback(error, output);
5 }
6
```

Execution Result

Execution successful

Function Output

hello message: {"hello":"function"}

Log Output

2022-11-16T01:39:20Z Start invoke request 'C05d2e-9b450538ccaf', version: latest
2022-11-16T01:39:20Z Finish invoke request 'C05d2e-9b450538ccaf', duration: 108.081ms, memory used: 25.339MB, billing memory: 120MB

Summary

Request ID	C05d2e-c7ef-4235-8b7a-9b450538ccaf
Memory Configured	128 MB
Execution Duration	108.081 ms
Memory Used	25.339 MB
Billed Duration	109 ms

NOTA

Um máximo de 2 KB de logs podem ser exibidos. Para obter mais informações de log, consulte [Consulta de logs de função](#).

Passo 5: exibir métricas de monitoramento

Na página de detalhes da função, clique na guia **Monitoring**.

- Na página de guia **Monitoring**, escolha **Metrics** e selecione um intervalo de tempo (como 5 minutos, 15 minutos ou 1 hora) para consultar a função.
- As seguintes métricas são exibidas: chamadas, erros, duração (incluindo as durações máxima, média e mínima) e aceleradores.

Passo 6: excluir uma função

1. Na página Detalhes da função, escolha **Operation** > **Delete function** no canto superior direito.
2. Na caixa de diálogo exibida, clique em **OK** para liberar recursos.

3 Criação de uma função usando um modelo

Introdução

O FunctionGraph fornece modelos para completar automaticamente o código e executar configurações de ambiente quando você cria uma função, ajudando a criar aplicações rapidamente.

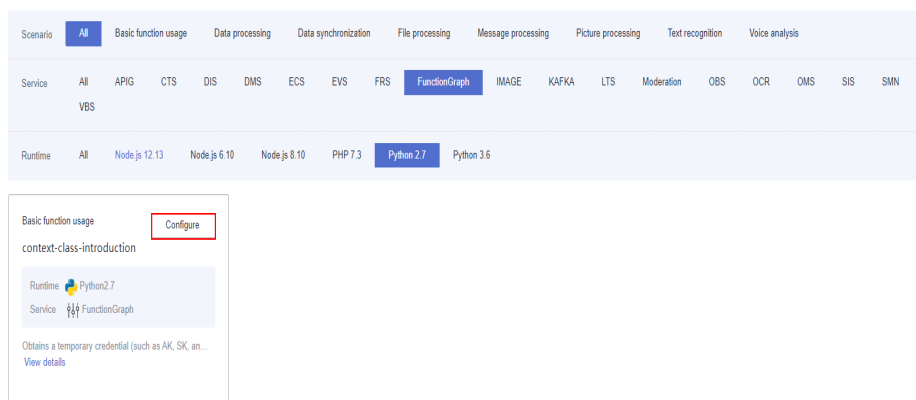
Passo 1: preparar o ambiente

Para executar as operações descritas nesta seção, verifique se você tem as permissões de **FunctionGraph Administrator**, ou seja, as permissões completas para o FunctionGraph. Para obter mais informações, consulte [Gerenciamento de permissões](#).

Passo 2: criar uma função

1. Efetue login no [console do FunctionGraph](#). No painel de navegação, escolha **Functions > Function List**.
2. Clique em **Create Function** no canto superior direito e escolha **Select template**.
3. Selecione o modelo mostrado em [Figura 3-1](#) e clique em **Configure**.

Figura 3-1 Selecionar um modelo



4. Defina o **Function Name** como **context**, selecione qualquer agência na lista suspensa de **Agency**, mantenha os valores padrão para outros parâmetros e clique em **Create Function**.

NOTA

Se nenhuma agência estiver configurada, a seguinte mensagem será exibida quando a função for acionada:

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

Figura 3-2 Configuração de informações básicas

Basic Information

Function Template
context-class-introduction-python [Reselect](#)

* Region

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

* Function Name

Enter 1 to 60 characters, starting with a letter and ending with a letter or digit. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Agency 

[Create Agency](#)

* Enterprise Project 

[View Enterprise Project](#)

Runtime 

Passo 3: testar a função

1. Na página de detalhes da função, clique em **Test**. Na caixa de diálogo exibida, crie um evento de teste.
2. Selecione **blank-template**, defina **Event Name** para **test** e clique em **Create**.

Figura 3-3 Configurar um evento de teste

Configure Test Event

☒ Create new test event ☐ Edit saved test event

Event Template (18)

Search

- blank-template
- apig-event-template
- cts-event-template
- dds-event-template
- gaussmongo-event-template
- dis-event-template
- dms-event-template
- image-tag
- kafka-event-template
- login-security-template

★ Event Name

```
1 {  
2   "key": "value"  
3 }
```

Passo 4: ver o resultado da execução.

Clique em **Test** e visualize o resultado da execução à direita.

- **Function Output**: exibe o resultado do retorno da função.
- **Log Output**: exibe os logs de execução da função.
- **Summary**: exibe informações-chave dos logs.

NOTA

Um máximo de 2 KB de logs podem ser exibidos. Para obter mais informações de log, consulte [Consulta de logs de função](#).

Passo 5: exibir métricas de monitoramento

Na página de detalhes da função, clique na guia **Monitoring**.

- Na página de guia **Monitoring**, escolha **Metrics** e selecione um intervalo de tempo (como 5 minutos, 15 minutos ou 1 hora) para consultar a função.
- As seguintes métricas são exibidas: chamadas, erros, duração (incluindo as durações máxima, média e mínima) e aceleradores.

Passo 6: excluir uma função

1. Na página Detalhes da função, escolha **Operation** > **Delete function** no canto superior direito.
2. Na caixa de diálogo exibida, clique em **OK** para liberar recursos.

4 Implementação de uma função usando uma imagem de contêiner

4.1 Desenvolvimento de uma função HTTP

Introdução

Ao desenvolver uma função HTTP usando uma imagem personalizada, implemente um servidor HTTP na imagem e ouça a porta 8000 para solicitações. As funções HTTP suportam apenas gatilhos de APIG.

Passo 1: preparar o ambiente

Para executar as operações descritas nesta seção, verifique se você tem as permissões de **FunctionGraph Administrator**, ou seja, as permissões completas para o FunctionGraph. Para obter mais informações, consulte [Gerenciamento de permissões](#).

Passo 2: criar uma imagem

Tome o sistema operacional Linux x86 de 64 bits como exemplo.

1. Crie uma pasta.
`mkdir custom_container_http_example && cd custom_container_http_example`
2. Implemente um servidor HTTP. Node.js é usado como exemplo. Para obter detalhes sobre outras linguagens, consulte [Criação de uma função HTTP](#).

Crie o arquivo **main.js** para introduzir o framework Express, receber solicitações POST, imprimir o corpo da solicitação como saída padrão e retornar "Hello FunctionGraph, method POST" para o 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. Crie o arquivo **package.json** para npm para que ele possa identificar as dependências do projeto e do projeto de processo.

```
{  
  "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:** nome do projeto
- **version:** versão do projeto
- **main:** arquivo de entrada da aplicação
- **dependencies:** todas as dependências disponíveis do projeto em npm

4. Crie um 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 a imagem de base **node:12.10.0**. A imagem de base é obrigatória e seu valor pode ser alterado.
- **ENV:** definir as variáveis de ambiente **HOME** (**/home/custom_container**), **GROUP_NAME** e **USER_NAME** (**custom_container**), **USER_ID** e **GROUP_ID** (**1003**). Essas variáveis de ambiente são obrigatórias e seus valores podem ser alterados.
- **RUN:** usar o formato **RUN <Command>**. Por exemplo, **RUN mkdir -m 550 \${HOME}**, que significa criar o diretório **home** para o usuário **\${USER_NAME}** durante a construção do contêiner.

- **USER**: alternar para o usuário `${USER_NAME}`.
- **WORKDIR**: alternar o diretório de trabalho para o diretório **home** do usuário `${USER_NAME}`.
- **COPY**: copiar **main.js** e **package.json** para o diretório **home** do usuário `${USER_NAME}` no contêiner.
- **EXPOSE**: expor a porta 8000 do contêiner. Não altere este parâmetro.
- **ENTRYPOINT**: executar o comando **node main.js** para iniciar o contêiner. Não altere este parâmetro.

NOTA

1. Você pode usar qualquer imagem de base.
 2. No ambiente de nuvem, o UID 1003 e o GID 1003 são usados para iniciar o contêiner por padrão. Os dois IDs podem ser modificados escolhendo **Configuration > Basic Settings > Container Image Override** na página de detalhes da função. Eles não podem ser **root** ou um ID reservado.
5. Crie uma imagem.

No exemplo a seguir, o nome da imagem é **custom_container_http_example**, a tag é **latest** e o ponto (.) indica o diretório em que o Dockerfile está localizado. Execute o comando de criação de imagem para empacotar todos os arquivos no diretório e enviar o pacote para um mecanismo de contêiner para criar uma imagem.

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

Passo 3: executar verificação local

1. Inicie o contêiner do Docker.

```
docker run -u 1003:1003 -p 8000:8000 custom_container_http_example:latest
```
2. Abra um novo prompt de comando e envie uma mensagem pela porta 8000 para acessar o diretório **/invoke** especificado no código do modelo.

```
curl -XPOST -H 'Content-Type: application/json' -d '{"message":"HelloWorld"}' localhost:8000/helloworld
```

As seguintes informações são retornadas com base no código do módulo:

```
Hello FunctionGraph, method POST
```

3. Verifique se as seguintes informações são exibidas:

```
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' }
```

Como alternativa, execute o comando **docker logs** para obter logs de contêiner.

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

Passo 4: carregar a imagem

1. Faça login no console do SoftWare Repository for Container (SWR). No painel de navegação, escolha **My Images**.
2. Clique em **Upload Through Client** ou **Upload Through SWR** no canto superior direito.

3. Carregue a imagem conforme solicitado.



4. Veja a imagem na página **My Images**.

Passo 5: criar uma função

1. Efetue login no [console do FunctionGraph](#). No painel de navegação, escolha **Functions > Function List**.
2. Clique em **Create Function** no canto superior direito e escolha **Select template**.
3. Defina as informações básicas.
 - **Function Type**: selecione **HTTP Function**.
 - **Function Name**: digite **custom_container_http**.
 - **Use container image**: selecione a imagem carregada para SWR. Exemplo: **swr.{ID de região}.myhuaweicloud.com/{nome da organização}/{nome da imagem}:{tag da imagem}**
 - **Agency**: selecione uma agência com a permissão **SWR Admin**. Se nenhuma agência estiver disponível, crie uma consultando [Criação de uma agência](#).
4. Após a conclusão da configuração, clique em **Create Function**.

Passo 6: testar a função

1. Na página de detalhes da função, clique em **Test**. Na caixa de diálogo exibida, crie um evento de teste.
2. Selecione **apig-event-template**, defina o **Event Name** como **helloworld**, modifique o evento de teste da seguinte forma e clique em **Create**.

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

Passo 7: ver o resultado da execução.

Clique em **Test** e visualize o resultado da execução à direita.

Figura 4-1 Resultado da execução

Execution Result ✕

✓ Execution successful

Function Output

```
{
  "body": "SGVsbG8gRnVudY3Rpb25HcmFwaCwgblV0aG9kIFBPU1Q=",
  "headers": {
    "Content-Length": [
      "32"
    ],
    "Content-Type": [
      "text/html; charset=utf-8"
    ],
    "Date": [
      "Wed, 02 Nov 2022 11:06:38 GMT"
    ],
    "Etag": [
      "W/\"20-uygbC2IEF2PxTtMC0H1BL5d/vwI\""
    ],
    "X-Powered-By": [
      "Express"
    ]
  },
  "statusCode": 200,
  "isBase64Encoded": true
}
```

Log Output

2022-11-02T11:06:38Z Start invoke request '7309717e-f597-4368-a7fe-89ac3d9b5df5', version: latest
receive { message: 'helloworld' }
2022-11-02T11:06:38Z Finish invoke request '7309717e-f597-4368-a7fe-89ac3d9b5df5', duration: 31.563ms, billing duration: 32ms, memory
used: 10.566MB, billing memory: 128MB

Summary

Request ID	7309717e-f597-4368-a7fe-89ac3d9b5df5
Memory Configured	128 MB
Execution Duration	34.247 ms
Memory Used	10.566 MB
Billed Duration	35 ms

- **Function Output:** exibe o resultado do retorno da função.
- **Log Output:** exibe os logs de execução da função.
- **Summary:** exibe informações importantes dos logs.

NOTA

Um máximo de 2 KB de logs podem ser exibidos. Para obter mais informações de log, consulte [Consulta de logs de função](#).

Passo 8: exibir métricas de monitoramento

Na página de detalhes da função, clique na guia **Monitoring**.

- Na página de guia **Monitoring**, escolha **Metrics** e selecione um intervalo de tempo (como 5 minutos, 15 minutos ou 1 hora) para consultar a função.
- As seguintes métricas são exibidas: chamadas, erros, duração (incluindo as durações máxima, média e mínima) e aceleradores.

Passo 9: excluir a função

1. Na página Detalhes da função, escolha **Operation** > **Delete function** no canto superior direito.
2. Na caixa de diálogo exibida, clique em **OK** para liberar recursos.

4.2 Desenvolvimento de uma função de evento

Introdução

Ao desenvolver uma função de evento usando uma imagem personalizada, implemente um servidor HTTP na imagem e ouça a porta 8000 para solicitações. Por padrão, o caminho de solicitação **/init** é a entrada de inicialização da função. Implemente-o conforme necessário. Para obter detalhes sobre a inicialização da função, consulte [Inicializador](#). O caminho de solicitação **/invoke** é a entrada de execução da função onde os eventos de gatilho são processados. Para obter detalhes sobre os parâmetros de solicitação, consulte [Fontes de eventos suportadas](#).

Passo 1: preparar o ambiente

Para executar as operações descritas nesta seção, verifique se você tem as permissões de **FunctionGraph Administrator**, ou seja, as permissões completas para o FunctionGraph. Para obter mais informações, consulte [Gerenciamento de permissões](#).

Passo 2: criar uma imagem

Tome o sistema operacional Linux x86 de 64 bits como exemplo.

1. Crie uma pasta.

```
mkdir custom_container_event_example && cd custom_container_event_example
```

2. Implemente um servidor HTTP para processar **init** e **invoke** solicitações e dar uma resposta. Node.js é usado como exemplo.

Crie o arquivo **main.js** para introduzir o framework Express e implementar um manipulador de função (método **POST** e caminho **/invoke** e um inicializador (método **POST** e caminho **/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. Crie o arquivo **package.json** para npm para que ele possa identificar as dependências do projeto e do projeto de processo.

```
{
  "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**: nome do projeto
- **version**: versão do projeto
- **main**: arquivo de entrada da aplicação
- **dependencies**: todas as dependências disponíveis do projeto em npm

4. Crie um 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 a imagem de base **node:12.10.0**. A imagem de base é obrigatória e seu valor pode ser alterado.
- **ENV**: definir as variáveis de ambiente **HOME** (**/home/custom_container**), **GROUP_NAME** e **USER_NAME** (**custom_container**), **USER_ID** e **GROUP_ID** (**1003**). Essas variáveis de ambiente são obrigatórias e seus valores podem ser alterados.
- **RUN**: usar o formato **RUN<Command>**. Por exemplo, **RUN mkdir -m 550 \${HOME}**, que significa criar o diretório **home** para o usuário **\${USER_NAME}** durante a construção do contêiner.
- **USER**: alternar para o usuário **\${USER_NAME}**.
- **WORKDIR**: alternar o diretório de trabalho para o diretório **home** do usuário **\${USER_NAME}**.
- **COPY**: copiar **main.js** e **package.json** para o diretório **home** do usuário **\${USER_NAME}** no contêiner.
- **EXPOSE**: expor a porta 8000 do contêiner. Não altere este parâmetro.
- **ENTRYPOINT**: executar o comando **node /home/tester/main.js** para iniciar o contêiner.

 NOTA

1. Você pode usar qualquer imagem de base.
2. No ambiente de nuvem, o UID 1003 e o GID 1003 são usados para iniciar o contêiner por padrão. Os dois IDs podem ser modificados escolhendo **Configuration > Basic Settings > Container Image Override** na página de detalhes da função. Eles não podem ser **root** ou um ID reservado.
5. Crie uma imagem.

No exemplo a seguir, o nome da imagem é **custom_container_event_example**, a tag é **latest** e o ponto (.) indica o diretório em que o Dockerfile está localizado. Execute o comando de criação de imagem para empacotar todos os arquivos no diretório e enviar o pacote para um mecanismo de contêiner para criar uma imagem.

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

Passo 3: executar verificação local

1. Inicie o contêiner do Docker.
2. Abra um novo prompt de comando e envie uma mensagem pela porta 8000 para acessar o diretório **/init** especificado no código do modelo.

```
curl -XPOST -H 'Content-Type: application/json' localhost:8000/init
```

As seguintes informações são retornadas com base no código do módulo:

```
Hello init
```

3. Abra um novo prompt de comando e envie uma mensagem pela porta 8000 para acessar o diretório **/invoke** especificado no código do modelo.

```
curl -XPOST -H 'Content-Type: application/json' -d '{"message":"HelloWorld"}' localhost:8000/invoke
```

As seguintes informações são retornadas com base no código do módulo:

```
Hello invoke
```

4. Verifique se as seguintes informações são exibidas:

```
Listening on http://localhost:8000
receive {}
receive { message: 'HelloWorld' }
```

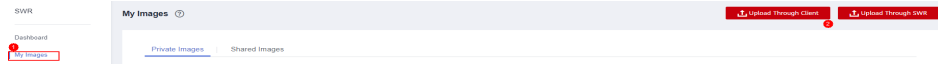
```
[root@ecs-74d7 ~]# docker run -u 1003:1003 -p 8000:8000 custom_container_event_example:latest
Listening on http://localhost:8000
receive {}
receive { message: 'HelloWorld' }
```

Como alternativa, execute o comando **docker logs** para obter logs de contêiner.

```
[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]#
```

Passo 4: carregar a imagem

1. Efetue login no console do SWR. No painel de navegação, escolha **My Images**.
2. Clique em **Upload Through Client** ou **Upload Through SWR** no canto superior direito.
3. Carregue a imagem conforme solicitado.



4. Veja a imagem na página **My Images**.

Passo 5: criar uma função

1. Efetue login no [console do FunctionGraph](#). No painel de navegação, escolha **Functions > Function List**.
2. Clique em **Create Function** no canto superior direito e escolha **Select template**.
3. Defina as informações básicas.
 - **Function Type**: selecione **Event Function**.
 - **Function Name**: digite `custom_container_event`.
 - **Use container image**: selecione a imagem carregada para SWR. Exemplo: `swr.{ID de região}.myhuaweicloud.com/{nome da organização}/{nome da imagem}:{tag da imagem}`
 - **Agency**: selecione uma agência com a permissão **SWR Admin**. Se nenhuma agência estiver disponível, crie uma consultando [Criação de uma agência](#).
4. Após a conclusão da configuração, clique em **Create Function**.
5. Na página de detalhes da função, escolha **Configuration > Advanced Settings** e ative **Initialization**. A API `init` será chamada para inicializar a função.

Passo 6: testar a função

1. Na página de detalhes da função, clique em **Test**. Na caixa de diálogo exibida, crie um evento de teste.
2. Selecione **blank-template**, defina o **Event Name** como **helloworld**, modifique o evento de teste da seguinte forma e clique em **Create**.

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

Passo 7: ver o resultado da execução.

Clique em **Test** e visualize o resultado da execução à direita.

Figura 4-2 Resultado da execução

Execution Result ✕

✓ Execution successful

Function Output

```
{
  "body": "SGVsbG8gRnVuY3Rpb25HcmFwaCwgblV0aG9kIFBPU1Q=",
  "headers": {
    "Content-Length": [
      "32"
    ],
    "Content-Type": [
      "text/html; charset=utf-8"
    ],
    "Date": [
      "Wed, 02 Nov 2022 11:06:38 GMT"
    ],
    "Etag": [
      "W/\"20-uygbC2IEf2PxTtMC0H1BL5d/vwI\""
    ],
    "X-Powered-By": [
      "Express"
    ]
  },
  "statusCode": 200,
  "isBase64Encoded": true
}
```

Log Output

```
2022-11-02T11:06:38Z Start invoke request '7309717e-f597-4368-a7fe-89ac3d9b5df5', version: latest
receive { message: 'helloworld' }
2022-11-02T11:06:38Z Finish invoke request '7309717e-f597-4368-a7fe-89ac3d9b5df5', duration: 31.563ms, billing duration: 32ms, memory
used: 10.566MB, billing memory: 128MB
```

Summary

Request ID	7309717e-f597-4368-a7fe-89ac3d9b5df5
Memory Configured	128 MB
Execution Duration	34.247 ms
Memory Used	10.566 MB
Billed Duration	35 ms

- **Function Output:** exibe o resultado do retorno da função.
- **Log Output:** exibe os logs de execução da função.
- **Summary:** exibe informações-chave dos logs.

NOTA

Um máximo de 2 KB de logs podem ser exibidos. Para obter mais informações de log, consulte [Consulta de logs de função](#).

Passo 8: exibir métricas de monitoramento

Na página de detalhes da função, clique na guia **Monitoring**.

- Na página de guia **Monitoring**, escolha **Metrics** e selecione um intervalo de tempo (como 5 minutos, 15 minutos ou 1 hora) para consultar a função.
- As seguintes métricas são exibidas: chamadas, erros, duração (incluindo as durações máxima, média e mínima) e aceleradores.

Passo 9: excluir a função

1. Na página Detalhes da função, escolha **Operation** > **Delete function** no canto superior direito.
2. Na caixa de diálogo exibida, clique em **OK** para liberar recursos.

