FunctionGraph

Guia de usuário

Edição 01

Data 2022-11-16

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registadas e permissões

e outras marcas registadas da Huawei são marcas registadas da Huawei Technologies Co., Ltd. Todos as outras marcas registadas e os nomes registados 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 Antes de começar

1.1 Uso do FunctionGraph

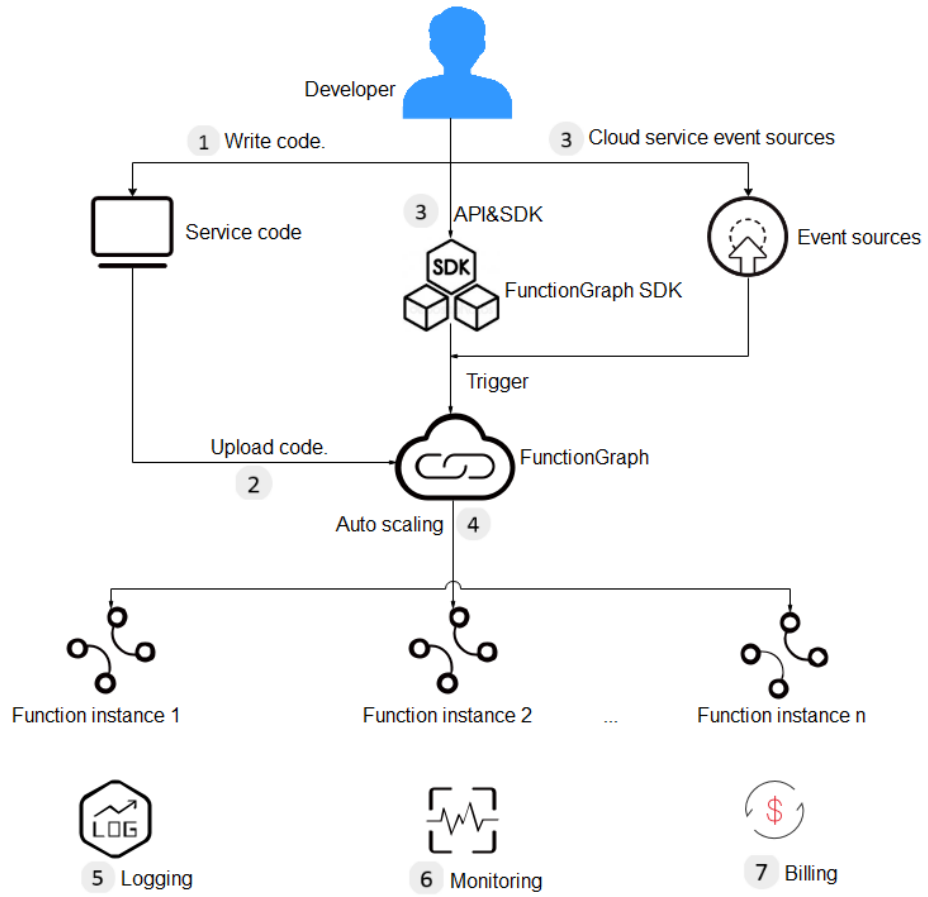
O FunctionGraph permite que você execute seu código sem provisionar ou gerenciar servidores, garantindo alta disponibilidade e escalabilidade. Tudo o que você precisa fazer é enviar seu código e definir as condições de execução, e o FunctionGraph cuidará do resto. Você paga apenas pelo que usa e não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

Processo

Figura 1-1 mostra o processo de utilização de funções.

1. Escreva código, empacote e carregue-o na FunctionGraph e adicione fontes de eventos como Notificação de Mensagem Simples (SMN), Serviço de Armazenamento de Objetos (OBS) e API Gateway (APIG) para criar aplicativos.
2. As funções são acionadas por chamadas de API RESTful ou origens de eventos para atingir os propósitos de serviço esperados. Durante esse processo, o FunctionGraph agenda automaticamente os recursos.
3. Visualize registrações e métricas. Observe que você será cobrado com base na duração da execução do código.

Figura 1-1 Fluxograma



O seguinte mostra os detalhes:

1. Grave o código.

   Escreva código em Node.js, Python, Java, Go, C# ou PHP. Para obter detalhes, consulte o **Guia de desenvolvedor do FunctionGraph**.

2. Carregue código.

   Edite o código em linha, carregue um arquivo ZIP ou JAR local ou carregue um arquivo ZIP do OBS. Para mais detalhes, veja **Criação de um pacote de implantação**.

3. Disparar funções por chamadas de API ou eventos de serviço de nuvem.

   As funções são acionadas por chamadas de API ou eventos de serviço de nuvem. Para mais detalhes, veja **Criação de gatilhos**.

4. Implemente o escalonamento automático.

   O FunctionGraph implementa o escalonamento automático com base no número de solicitações. Para obter detalhes, consulte **Cotas e restrições de uso**.

5. Visualize registrações.

   Visualize registrações de execução de função como FunctionGraph está interconectado com Log Tank Service (LTS). Para mais detalhes, veja **Registrações**.

6. Visualize informações de monitoramento.

   Visualize informações gráficas de monitoramento à medida que o FunctionGraph é interconectado com o Cloud Eye. Para mais detalhes, veja **Métricas**.

7. Modo de cobrança

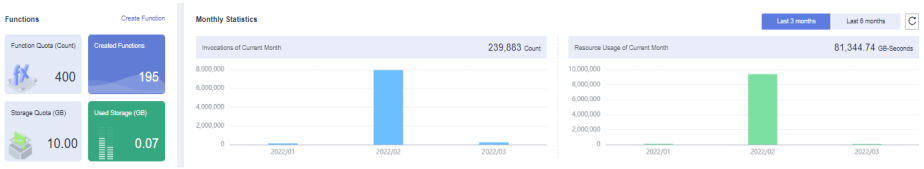
   Depois que uma função é executada, você será cobrado com base no número de solicitações de execução da função e na duração da execução. (v1: faturado a cada 100 ms; v2: faturado a cada 1 ms)

Introdução ao Dashboard

Faça login no console do FunctionGraph e escolha **Dashboard** no painel de navegação à esquerda.

- Visualize as funções/quota de funções criadas, a cota de armazenamento/armazenamento usada e as chamadas mensais e o uso de recursos.

Figura 1-2 Estatísticas mensais



- Você pode exibir informações de monitoramento no nível do locatário, incluindo o número de chamadas, o número de erros, a duração e o número de aceleradores, conforme mostrado em **Figura 1-3**.

Figura 1-3 Métricas

Tabela 1-1 descreve as métricas da função.

Tabela 1-1 Métricas de função

Métrica	Unidade	Descrição
Invocações	Contagem	Número total de solicitações de invocação, incluindo erros de invocação e invocações estranguladas. No caso de invocação assíncrona, a contagem começa somente quando uma função é executada em resposta a uma solicitação.

Métrica	Unidade	Descrição
Duração	ms	**Maximum Duration**: a duração máxima que todas as funções são executadas de cada vez dentro de um período. **Minimum Duration**:Duração mínima: a duração mínima em que todas as funções são executadas de cada vez dentro de um período. **Average Duration**: a duração média que todas as funções são executadas de cada vez dentro de um período.
Erros	Contagem	Número de vezes que suas funções falharam com o código de erro **200**sendo retornado. Erros causados pela sintaxe da função ou execução também estão incluídos.
Aceleradores	Contagem	Número de vezes que o FunctionGraph limita suas funções devido ao limite de recursos.

1.2 Gerenciamento de permissões

1.2.1 Criação de um usuário e concessão de permissões

Esta seção descreve como usar **Gerenciamento de identidade e acesso (IAM)** para implementar o controle de permissões refinado para seus recursos do FunctionGraph. Com o IAM, você pode:

- Crie usuários do IAM para funcionários com base na estrutura organizacional da sua empresa. Cada usuário do IAM tem suas próprias credenciais de segurança para acessar os recursos do FunctionGraph.
- Conceda somente as permissões necessárias para que os usuários executem uma tarefa.
- Confie outras contas ou serviços em nuvem para realizar O&M profissional e eficiente em seus recursos de FunctionGraph.

Se sua conta não precisar de usuários individuais do IAM, você pode pular este capítulo.

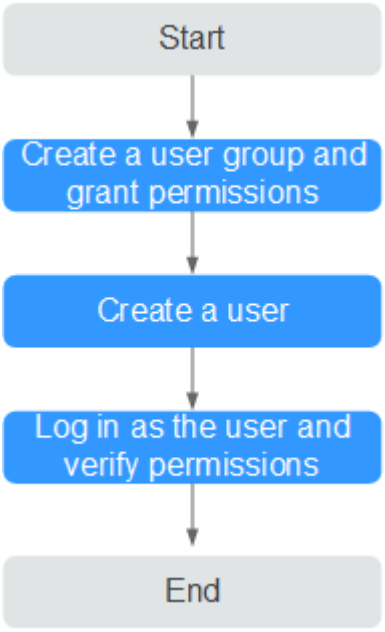
Esta seção descreve o procedimento para conceder permissões. Para mais detalhes, veja **Figura 1-4**.

Pré-requisitos

Antes de atribuir permissões a grupos de usuários, você deve aprender sobre as permissões do sistema listadas em **Descrição de permissões**. Para obter as políticas do sistema de outros serviços, consulte **Permissões do sistema**.

Processo

Figura 1-4 Processo para conceder permissões de FunctionGraph



1. **Criação de um grupo de usuários e atribuição de permissões**

   Crie um grupo de usuários no console do IAM e atribua o papel **FunctionGraph Invoker** ao grupo.

2. **Criação de um usuário do IAM**

   Crie um usuário no console do IAM e adicione o usuário ao grupo criado em **1**.

3. **Execução de login como um usuário do IAM** e verificação de permissões

   Faça login no console de gerenciamento como o usuário criado e verifique se esse usuário só tem permissões de leitura para FunctionGraph:

   - Escolha **Service List** > **FunctionGraph** para acessar o console do FunctionGraph. No painel de navegação, escolha **Functions** > **Function List**. Em seguida, clique em **Create Function**. Se aparecer uma mensagem indicando permissões insuficientes para executar a operação, a função **FunctionGraph Invoker** já entrou em vigor.
   - Escolha qualquer outro serviço na **Service List**. Se uma mensagem for exibida indicando permissões insuficientes para acessar o serviço, a função **FunctionGraph Invoker** já entrou em vigor.

1.2.2 Criação de uma política personalizada

Políticas personalizadas podem ser criadas como um complemento às políticas do sistema do FunctionGraph.

Você pode criar políticas personalizadas de uma das seguintes maneiras:

- Editor visual: Selecione serviços de nuvem, ações, recursos e condições de solicitação. Isso não requer conhecimento de sintaxe política.

- JSON: Edite políticas JSON do zero ou com base em uma política existente.

Para obter detalhes, consulte **Criação de uma política personalizada**. Esta seção apresenta exemplos de políticas personalizadas comuns do FunctionGraph.

Exemplo de Políticas Personalizadas

- Exemplo 1: Autorizando um usuário a consultar código de função e configuração

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "functiongraph:function:list",
               "functiongraph:function:getConfig",
               "funcitongraph:function:getCode"
            ]
         }
    ]
}
```

- Exemplo 2: Negação da exclusão de função

  Uma política com apenas permissões "Negar" deve ser usada em conjunto com outras políticas para entrar em vigor. Se as permissões "Permitir" e "Negar" forem atribuídas a um usuário, as permissões "Negar" terão precedência sobre as permissões "Permitir".

  Se você precisar atribuir permissões da política **FunctionGraph FullAccess** a um usuário, mas impedir que ele exclua funções, crie uma política personalizada para negar exclusão de função e anexe ambas ao grupo ao qual o usuário pertence. Desta forma, o usuário pode executar todas as operações no FunctionGraph, exceto a exclusão de funções. O seguinte é um exemplo de uma política de negação:

```
{
    "Version": "1.1",
    "Statement": [
        "Effect": "Deny",
        "Action": [
            "functiongraph:function:delete"
        ]
    ]
}
```

- Exemplo 3: Configuração das permissões para recursos específicos

  Você pode conceder permissões de usuário do IAM para recursos específicos. Por exemplo, para conceder permissões a um usuário para a função **functionname** no aplicativo **Default**, defina **functionname** para um caminho de recurso especificado, ou seja, **FUNCTIONGRAPH:*:*:function:Default/functionname**.

  NOTA

  Especifique os recursos da função:

  Formato: **FUNCTIONGRAPH:*:*:function:** *application or function name*

  Para recursos de função, o IAM gera automaticamente o prefixo de caminho de recurso **FUNCTIONGRAPH:*:*:function:**. Você pode especificar um caminho de recurso adicionando o nome do aplicativo ou da função ao lado do prefixo do caminho. Wildcards (*) são suportados. Por exemplo, **FUNCTIONGRAPH:*:*:function:Default/*** indica qualquer função na aplicação **Default**.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
```



```
            "Action": [
                "functiongraph:function:list"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "functiongraph:function:listAlias",
                "functiongraph:function:listVersion",
                "functiongraph:function:getConfig",
                "functiongraph:function:getCode",
                "functiongraph:function:updateCode",
                "functiongraph:function:invoke",
                "functiongraph:function:updateConfig",
                "functiongraph:function:createVersion",
                "functiongraph:function:updateAlias",
                "functiongraph:function:createAlias"
            ],
            "Resource": [
                "FUNCTIONGRAPH:*:*:function:Default/*"
            ]
        }
    ]
}
```

1.3 Linguagens de programação suportadas

1.3.1 Node.js

√: Compatível. ×: Incompatível.

Tempo de execução	FunctionGraph V1	FunctionGraph V2	Guia de desenvolvimento
Node.js 6.10	√	√	Para obter detalhes sobre a sintaxe da função, as APIs do SDK e o desenvolvimento de funções, consulte **Desenvolvimento de funções no Node.js**.
Node.js 8.10	√	√	
Node.js 10.16	√	√	
Node.js 12.13	√	√	
Node.js 14.18	×	√	

1.3.2 Python

√: Compatível. ×: Incompatível.

Tempo de execução	FunctionGraph V1	FunctionGraph V2	Guia de desenvolvimento
Python 2.7	√	√	Para obter detalhes sobre sintaxe de funções, as API de SDK e desenvolvimento de funções, consulte **Desenvolvimento de funções em Python**.
Python 3.6	√	√	
Python 3.9	×	√	

1.3.3 Java

√: Compatível. ×: Incompatível.

Tempo de execução	FunctionGraph da V1	FunctionGraph da V2	Guia de desenvolvimento
Java 8	√	√	Para obter detalhes sobre sintaxe de função, as API de SDK e desenvolvimento de funções, consulte **Desenvolvimento de funções em Java**.
Java 11	×	√	

1.3.4 Go

√: Compatível. ×: Incompatível.

Tempo de execução	FunctionGraph da V1	FunctionGraph da V2	Guia de desenvolvimento
Go 1.8	√	×	Para obter detalhes sobre sintaxe de função, as API de SDK e desenvolvimento de funções, consulte Desenvolvimento de funções em Go.
Go 1.x	√	√	

1.3.5 C#

√: Compatível. ×: Incompatível.

Tempo de execução	Function Graph V1	FunctionGraph V2	Guia de desenvolvimento
C# (.NET Core 2.0)	√	×	Para obter detalhes sobre sintaxe de função, as API de SDK e desenvolvimento de funções, consulte **Desenvolvimento de funções em C#**.
C# (.NET Core 2.1)	√	×	
C# (.NET Core 3.1)	√	×	

1.3.6 PHP

√: Compatível. ×: Incompatível.

Tempo de execução	Function Graph V1	FunctionGraph V2	Guia de desenvolvimento
PHP 7.3	√	×	Para obter detalhes sobre sintaxe de funções, as API do SDK, e desenvolvimento de funções, consulte **Desenvolvimento de funções de PHP**.

1.3.7 Tempo de execução personalizado

Cenários

Um tempo de execução executa o código de uma função, lê o nome do manipulador de uma variável de ambiente e lê eventos de invocação das API de tempo de execução do FunctionGraph. O tempo de execução passa os dados do evento para o manipulador da função e retorna a resposta do manipulador para FunctionGraph.

O FunctionGraph oferece suporte a tempos de execução personalizados. Você pode usar um arquivo executável chamado **bootstrap** para incluir um runtime no pacote de implantação da função. O tempo de execução executa o método manipulador da função quando a função é invocada.

Seu tempo de execução é executado no ambiente de execução do FunctionGraph. Pode ser um script shell ou um arquivo executável binário compilado no Linux.

NOTA

Após a programação, basta empacotar seu código em um arquivo ZIP (Java, Node.js, Python e Go) ou arquivo JAR (Java) e fazer o upload do arquivo para FunctionGraph para execução. Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.

Se você editar código em Go, compacte o arquivo compilado e verifique se o nome do arquivo de biblioteca dinâmica é consistente com o nome do plug-in do manipulador. Por exemplo, se o nome do arquivo de biblioteca dinâmica for **testplugin.so**, defina o nome do manipulador como **testplugin.Handler**.

Bootstrap do arquivo de tempo de execução

Se houver um arquivo chamado **bootstrap** no pacote de implantação da função, o FunctionGraph executará esse arquivo. Se o arquivo **bootstrap** não for encontrado ou não for executável, sua função retornará um erro quando for invocada.

O código de tempo de execução é responsável por concluir as tarefas de inicialização. Ele processa eventos de invocação em um loop até que seja encerrado.

As tarefas de inicialização são executadas uma vez para cada instância da função para preparar o ambiente para manipulação de invocações.

As API de tempo de execução

O FunctionGraph fornece as API de tempo de execução HTTP para receber eventos de chamada de função e retornar dados de resposta no ambiente de execução.

- **Próxima invocação**

  Método – Get

  Caminho – http://$RUNTIME_API_ADDR/v1/runtime/invocation/request

  Essa API é usada para recuperar um evento de invocação. O corpo da resposta contém os dados do evento. A tabela a seguir descreve dados adicionais sobre a invocação contida no cabeçalho da resposta.

  Tabela 1-2 Informações de cabeçalho de resposta

Parâmetro	Descrição
X-Cff-Request-Id	Solicitar ID.
X-CFF-Access-Key	AK da conta. Uma agência deve ser configurada para a função se essa variável for usada.
X-CFF-Auth-Token	Token da conta. Uma agência deve ser configurada para a função se essa variável for usada.
X-CFF-Invoke-Type	Tipo de invocação da função.
X-CFF-Secret-Key	SK da conta. Uma agência deve ser configurada para a função se essa variável for usada.

Parâmetro	Descrição
X-CFF-Security-Token	Token de segurança da conta. Uma agência deve ser configurada para a função se essa variável for usada.

- **Resposta de invocação**

  Método – POST

  Caminho – http://$RUNTIME_API_ADDR/v1/runtime/invocation/response/$REQUEST_ID

  Essa API é usada para enviar uma resposta de invocação bem-sucedida para a FunctionGraph. Depois que o tempo de execução invoca o manipulador da função, ele publica a resposta da função no caminho de resposta da invocação.

- **Erro de invocação**

  Método – POST

  Caminho – http://$RUNTIME_API_ADDR/v1/runtime/invocation/error/$REQUEST_ID

  $REQUEST_ID é o valor da variável **X-Cff-Request-Id** no cabeçalho de uma resposta de recuperação de evento. Para obter mais informações, consulte **Tabela 1-2**.

  $RUNTIME_API_ADDR é uma variável de ambiente do sistema. Para obter mais informações, consulte **Tabela 1-3**.

  Essa API é usada para enviar uma resposta de invocação de erro para a FunctionGraph. Depois que o tempo de execução invoca o manipulador da função, ele publica a resposta da função no caminho de resposta da invocação.

Variáveis de ambiente de tempo de execução

Você pode usar variáveis de ambiente personalizadas e de tempo de execução no código de função. A tabela a seguir lista as variáveis de ambiente de tempo de execução usadas no ambiente de execução do FunctionGraph.

Tabela 1-3 Variáveis de ambiente

Chave	Descrição
RUNTIME_PROJECT_ID	ID do projeto
RUNTIME_FUNC_NAME	Nome da função
RUNTIME_FUNC_VERSION	Versão da função
RUNTIME_PACKAGE	Aplicativo ao qual a função pertence
RUNTIME_HANDLER	Manipulador de funções
RUNTIME_TIMEOUT	Duração do timeout da função
RUNTIME_USERDATA	Valor passado por uma variável de ambiente
RUNTIME_CPU	Número de núcleos de CPU alocados

Chave	Descrição
RUNTIME_MEMORY	Memória alocada
RUNTIME_CODE_ROOT	Diretório que armazena o código da função
RUNTIME_API_ADDR	Endereço IP do host e porta de uma API de tempo de execução personalizada

O valor de uma variável de ambiente personalizada pode ser recuperado da mesma forma que o valor de uma variável de ambiente FunctionGraph.

Exemplo

Este exemplo contém um arquivo chamado **bootstrap**. O arquivo é implementado em Bash.

O tempo de execução carrega o script de função do pacote de implantação usando duas variáveis.

O arquivo **bootstrap** é o seguinte:

```
#!/bin/sh

set -o pipefail

#Processing requests loop
while true
do

HEADERS="$(mktemp)"
  # Get an event
  EVENT_DATA=$(curl
-sS -LD "$HEADERS" -X GET
"http://$RUNTIME_API_ADDR/v1/runtime/invocation/request")
  # Get request id
from response header
  REQUEST_ID=$(grep
-Fi x-cff-request-id "$HEADERS" | tr -d '[:space:]' | cut -d: -f2)
  if [ -z
"$REQUEST_ID" ]; then
    continue
  fi
  # Process request
data

RESPONSE="Echoing request: '$EVENT_DATA'"
  # Put response
  curl -X POST
"http://$RUNTIME_API_ADDR/v1/runtime/invocation/response/$REQUEST_ID"
-d "$RESPONSE"
done
```

Após carregar o script, o runtime processa eventos de invocação em um loop até que ele seja encerrado. Ele usa a API para recuperar eventos de invocação do FunctionGraph e envia as respostas para FunctionGraph.

Para obter a ID da solicitação, o runtime salva o cabeçalho da resposta da API em um arquivo temporário e lê a ID da solicitação no campo do cabeçalho **x-cff-request-id**. O tempo de execução processa os dados do evento recuperados e envia uma resposta de volta para FunctionGraph.

A seguir, um exemplo de código-fonte em Go. Ele pode ser executado somente após a compilação.

```
package main

import (
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "io/ioutil"
    "log"
    "net"
    "net/http"
    "os"
    "strings"
    "time"
)

var (
    getRequestUrl           = os.ExpandEnv("http://$
{RUNTIME_API_ADDR}/v1/runtime/invocation/request")
    putResponseUrl          = os.ExpandEnv("http://$
{RUNTIME_API_ADDR}/v1/runtime/invocation/response/{REQUEST_ID}")
    putErrorResponseUrl     = os.ExpandEnv("http://$
{RUNTIME_API_ADDR}/v1/runtime/invocation/error/{REQUEST_ID}")
    requestIdInvalidError   = fmt.Errorf("request id invalid")
    noRequestAvailableError = fmt.Errorf("no request available")
    putResponseFailedError  = fmt.Errorf("put response failed")
    functionPackage         = os.Getenv("RUNTIME_PACKAGE")
    functionName            = os.Getenv("RUNTIME_FUNC_NAME")
    functionVersion         = os.Getenv("RUNTIME_FUNC_VERSION")

    client = http.Client{
        Transport: &http.Transport{
            DialContext: (&net.Dialer{
                Timeout: 3 * time.Second,
            }).DialContext,
        },
    }
)

func main() {
    // main loop for processing requests.
    for {
        requestId, header, payload, err := getRequest()
        if err != nil {
            time.Sleep(50 * time.Millisecond)
            continue
        }

        result, err := processRequestEvent(requestId, header, payload)
        err = putResponse(requestId, result, err)
        if err != nil {
            log.Printf("put response failed, err: %s.", err.Error())
        }
    }
}

// event processing function
func processRequestEvent(requestId string, header http.Header, evtBytes
[]byte) ([]byte, error) {
```



```
    log.Printf("processing request '%s'.", requestId)
    result := fmt.Sprintf("function: %s:%s:%s, request id: %s, headers: %
+v, payload: %s", functionPackage, functionName,
        functionVersion, requestId, header, string(evtBytes))

    var event FunctionEvent
    err := json.Unmarshal(evtBytes, &event)
    if err != nil {
        return (&ErrorMessage{ErrorType: "invalid event", ErrorMessage:
"invalid json formated event"}).toJsonBytes(), err
    }

    return (&APIGFormatResult{StatusCode: 200, Body:
result}).toJsonBytes(), nil
}

func getRequest() (string, http.Header, []byte, error) {
    resp, err := client.Get(getRequestUrl)
    if err != nil {
        log.Printf("get request error, err: %s.", err.Error())
        return "", nil, nil, err
    }
    defer resp.Body.Close()

    // get request id from response header
    requestId := resp.Header.Get("X-CFF-Request-Id")
    if requestId == "" {
        log.Printf("request id not found.")
        return "", nil, nil, requestIdInvalidError
    }

    payload, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        log.Printf("read request body error, err: %s.", err.Error())
        return "", nil, nil, err
    }

    if resp.StatusCode != 200 {
        log.Printf("get request failed, status: %d, message: %s.",
resp.StatusCode, string(payload))
        return "", nil, nil, noRequestAvailableError
    }

    log.Printf("get request ok.")
    return requestId, resp.Header, payload, nil
}

func putResponse(requestId string, payload []byte, err error) error {
    var body io.Reader
    if payload != nil && len(payload) > 0 {
        body = bytes.NewBuffer(payload)
    }

    url := ""
    if err == nil {
        url = strings.Replace(putResponseUrl, "{REQUEST_ID}", requestId,
-1)
    } else {
        url = strings.Replace(putErrorResponseUrl, "{REQUEST_ID}",
requestId, -1)
    }
```



```
    resp, err := client.Post(strings.Replace(url, "{REQUEST_ID}",
requestId, -1), "", body)
    if err != nil {
        log.Printf("put response error, err: %s.", err.Error())
        return err
    }
    defer resp.Body.Close()

    responsePayload, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        log.Printf("read request body error, err: %s.", err.Error())
        return err
    }

    if resp.StatusCode != 200 {
        log.Printf("put response failed, status: %d, message: %s.",
resp.StatusCode, string(responsePayload))
        return putResponseFailedError
    }

    return nil
}

type FunctionEvent struct {
    Type string `json:"type"`
    Name string `json:"name"`
}

type APIGFormatResult struct {
    StatusCode      int               `json:"statusCode"`
    IsBase64Encoded bool              `json:"isBase64Encoded"`
    Headers         map[string]string `json:"headers,omitempty"`
    Body            string            `json:"body,omitempty"`
}

func (result *APIGFormatResult) toJsonBytes() []byte {
    data, err := json.MarshalIndent(result, "", "  ")
    if err != nil {
        return nil
    }

    return data
}

type ErrorMessage struct {
    ErrorType     string `json:"errorType"`
    ErrorMessage  string `json:"errorMessage"`
}

func (errMsg *ErrorMessage) toJsonBytes() []byte {
    data, err := json.MarshalIndent(errMsg, "", "  ")
    if err != nil {
        return nil
    }

    return data
}
```

Tabela 1-4 descreve as variáveis de ambiente usadas no código anterior.

Tabela 1-4 Variáveis de ambiente

Variável de ambiente	Descrição
RUNTIME_FUNC_NAME	Nome da função
RUNTIME_FUNC_VERSION	Versão da função
RUNTIME_PACKAGE	Aplicativo ao qual a função pertence

2 Construção de uma função de FunctionGraph

2.1 Criação de um pacote de implantação

Para criar uma função, você deve criar um pacote de implantação que inclua seu código e todas as dependências. Você pode criar um pacote de implantação localmente ou editar código no console do FunctionGraph. Se você editar código embutido, o FunctionGraph criará e carregará automaticamente um pacote de implantação para sua função. O FunctionGraph permite que você edite o código da função da mesma forma que gerencia um projeto. Você pode criar e editar arquivos e pastas. Depois de carregar um pacote de CEP, você pode exibir e editar o código no console.

NOTA

- Após a programação, basta empacotar seu código em um arquivo ZIP (Java, Node.js, Python e Go) ou arquivo JAR (Java) e fazer o upload do arquivo para FunctionGraph para execução.
- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- Se você editar código em Go, compacte o arquivo compilado e verifique se o nome do arquivo de biblioteca dinâmica é consistente com o nome do plug-in do manipulador. Por exemplo, se o nome do arquivo de biblioteca dinâmica for **testplugin.so**, defina o nome do manipulador como **testplugin.Handler**.

Tabela 2-1 lista os modos de entrada de código suportados pelo FunctionGraph para cada tempo de execução.

Tabela 2-1 Modos de entrada de código

Tempo de execução	Edição de código em linha	Carregamento de um arquivo ZIP	Carregamento de um arquivo JAR	Carregamento de um arquivo ZIP do OBS
Node.js	Compatível	Compatível	Incompatível	Compatível
Python	Compatível	Compatível	Incompatível	Compatível

Tempo de execução	Edição de código em linha	Carregamento de um arquivo ZIP	Carregamento de um arquivo JAR	Carregamento de um arquivo ZIP do OBS
Java	Incompatível	Compatível	Compatível	Compatível
Go	Incompatível	Compatível	Incompatível	Compatível
C#	Incompatível	Compatível	Incompatível	Compatível
PHP	Compatível	Compatível	Incompatível	Compatível
Custom runtime	Compatível	Compatível	Incompatível	Compatível

AVISO

Se o código a ser carregado contiver informações confidenciais (como senhas de conta), criptografe as informações confidenciais para evitar vazamentos.

Node.js

Editing Code Inline

O FunctionGraph fornece um SDK para edição de código em Node.js. Se o código personalizado usar apenas a biblioteca do SDK, você poderá editar o código usando o editor embutido no console do FunctionGraph. Após editar o código embutido e carregá-lo na FunctionGraph o console compacta o código e as configurações relacionadas em um pacote de implantação que o FunctionGraph pode executar.

Uploading a Deployment Package

Se o código usar outros recursos, como uma biblioteca gráfica para processamento de imagens, crie um pacote de implantação e carregue-o no console do FunctionGraph. Você pode carregar um pacote de implantação do Node.js de duas maneiras.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

- Carregamento de um pacote de implantação usando um bucket do OBS

  Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

Python

Editing Code Inline

O FunctionGraph fornece um SDK para edição de código em Python. Se o código personalizado usar apenas a biblioteca do SDK, você poderá editar o código usando o editor embutido no console do FunctionGraph. Após editar o código embutido e carregá-lo na FunctionGraph o console compacta o código e as configurações relacionadas em um pacote de implantação que o FunctionGraph pode executar.

Uploading a Deployment Package

Se o código usar outros recursos, como uma biblioteca gráfica para processamento de imagens, crie um pacote de implantação e carregue-o no console do FunctionGraph. Você pode carregar um pacote de implantação do Python de duas maneiras.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.
- Quando você escreve código em Python, não nomeie seu pacote com o mesmo sufixo que uma biblioteca Python padrão, como **json**, **lib**, e **os**. Caso contrário, um erro indicando uma falha de carregamento do módulo será relatado.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

- Carregamento de um pacote de implantação usando um bucket do OBS

  Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

Java

Java é uma linguagem compilada, que não suporta edição de código em linha. Você só pode carregar um pacote de implantação local, que pode ser um arquivo ZIP ou JAR.

Uploading a JAR File

- Se a função não usar nenhuma dependência, carregue diretamente um arquivo JAR.
- Se a função usar dependências, faça o upload delas em um bucket do OBS, defina-as durante a criação da função e carregue o arquivo JAR.

Uploading a ZIP File

Se a função usar dependências de terceiros, compacte as dependências e o arquivo JAR da função em um arquivo ZIP e, em seguida, carregue o arquivo ZIP.

Você pode carregar um pacote de implantação Java de duas maneiras.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.
- Carregamento de um pacote de implantação usando um bucket do OBS

  Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

Go

Uploading a Deployment Package

Você só pode carregar um pacote de implantação do Go no formato ZIP. Há duas maneiras de fazer o upload.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

- Carregamento de um pacote de implantação usando um bucket do OBS

  Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

C#

Uploading a Deployment Package

Você só pode carregar um pacote de implantação C# no formato ZIP. Há duas maneiras de fazer o upload.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

- Carregamento de um pacote de implantação usando um bucket do OBS

  Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

PHP

Editing Code Inline

O FunctionGraph fornece um SDK para edição de código em PHP. Se o código personalizado usar apenas a biblioteca do SDK, você poderá editar o código usando o editor embutido no console do FunctionGraph. Após editar o código embutido e carregá-lo na FunctionGraph o console compacta o código e as configurações relacionadas em um pacote de implantação que o FunctionGraph pode executar.

Uploading a Deployment Package

Se o código usar outros recursos, como uma biblioteca gráfica para processamento de imagens, crie um pacote de implantação e carregue-o no console do FunctionGraph. Você pode fazer o upload de um pacote de implantação PHP de duas maneiras.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.
- Carregamento de um pacote de implantação usando um bucket do OBS

  Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

Tempo de execução personalizado

Editing Code Inline

Após editar o código embutido e carregá-lo na FunctionGraph o console compacta o código e as configurações relacionadas em um pacote de implantação que o FunctionGraph pode executar.

Uploading a Deployment Package

Se o código usar outros recursos, como uma biblioteca gráfica para processamento de imagens, crie um pacote de implantação e carregue-o no console do FunctionGraph. Você pode carregar um pacote de implantação para um tempo de execução personalizado de duas maneiras.

AVISO

- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- O tamanho do código-fonte descompactado não pode exceder 1,5 GB. Se o código for muito grande, entre em contato com o especialista.

- Carregamento de diretamente um pacote de implantação local

  Depois de criar um pacote de implantação ZIP, carregue-o no console do FunctionGraph. Se o tamanho do pacote exceder 50 MB, carregue o pacote do OBS.

  Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.
- Carregamento de um pacote de implantação usando um bucket do OBS

Depois de criar um pacote de implantação ZIP, carregue-o em um intervalo do OBS na mesma região do FunctionGraph e cole o URL do link do intervalo do OBS na função. O tamanho máximo do arquivo ZIP que pode ser carregado para o OBS é de 300 MB.

Para obter detalhes sobre restrições de recursos de função, consulte **Cotas e restrições de uso**.

2.2 Criação de uma função a partir do zero

2.2.1 Criação de uma função de evento

Visão geral

Uma função é um código personalizado para processar eventos. Você pode criar uma função a partir do zero e configurar a função com base nos requisitos do site.

O FunctionGraph gerencia os recursos de computação necessários para a execução da função. Depois de editar o código da sua função, configure os recursos de computação no console do FunctionGraph.

Você pode criar uma função a partir do zero ou usando **um modelo** ou **imagem de contêiner**.

NOTA

Ao criar uma função a partir do zero, configure as informações básicas e de código com base em **Tabela 2-2**. Os parâmetros marcados com um asterisco (*) são obrigatórios.

Depois de criar uma função, modifique sua configuração com base em **Modificação de configurações de função**, se necessário.

Cada função do FunctionGraph é executada em seu próprio ambiente e tem seus próprios recursos e sistema de arquivos.

Pré-requisitos

1. Você deve estar familiarizado com as linguagens de programação suportadas pelo FunctionGraph. Para mais detalhes, veja **Linguagens de programação suportadas**.
2. Você criou um pacote. Para mais detalhes, veja **Criação de um pacote de implantação**.
3. (Opcional) Você criou uma agência. Para mais detalhes, veja **Configuração das permissões de agência**.

Procedimento

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Na página **Function List**, clique em **Create Function** no canto superior direito.
3. Clique em **Create from scratch** e configure as informações da função consultando **Tabela 2-2**. Os parâmetros marcados com um asterisco (*) são obrigatórios.

Figura 2-1 Criação de uma função a partir do zero

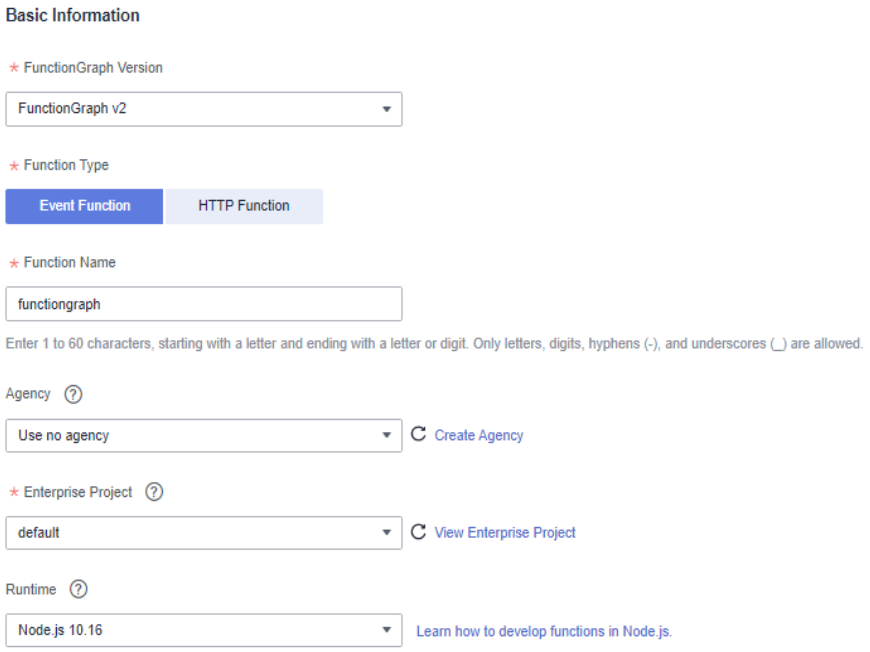


Tabela 2-2 Informações básicas

Parâmetro	Descrição
*FunctionGraph Version	Selecione uma versão do FunctionGraph. ● v1: hospeda funções orientadas a eventos em um contexto sem servidor. ● FunctionGraph v2: um serviço de hospedagem de funções de última geração com a arquitetura YuanRong da Huawei .
* Function Type	Se você definir **FunctionGraph Version** para **FunctionGraph v2**, srá necessário selecionar um tipo de função. ● Funções de evento: acionadas por gatilhos. ● Funções HTTP: acionadas assim que as solicitações de HTTP são enviadas para as URL específicas. **NOTA** ● As funções HTTP não fazem distinção entre linguagens de programação. O manipulador deve ser definido no arquivo **bootstrap**. Você pode escrever diretamente o comando de inicialização e permitir o acesso pela porta 8000. ● As funções HTTP suportam apenas gatilhos de APIG e APIC. ● Para obter detalhes sobre como usar funções de HTTP, consulte **Criação de uma função de HTTP**.
*Function Name	Nome da função, que deve atender aos seguintes requisitos: ● Consiste de 1 a 60 caracteres e pode conter letras, dígitos, hífens (-) e sublinhados (_). ● Começa com uma letra e termina com uma letra ou dígito.

Parâmetro	Descrição
Agency	Uma agência é necessária se o FunctionGraph acessar outros serviços em nuvem. Para obter detalhes sobre como criar uma agência, consulte **Configuração das permissões de agência**. Nenhuma agência é necessária se o FunctionGraph não acessar nenhum serviço em nuvem.
*Enterprise Project	Selecione um projeto empresarial criado e adicione a função a ele. Por padrão, **default** é selecionado. **NOTA** Se o EPS (Enterprise Project Management Service) não estiver ativado, esse parâmetro não estará disponível. Para obter detalhes, consulte **Ativação da função do projeto empresarial**.
Runtime	Selecione um runtime para compilar a função. **AVISO** O CloudIDE é compatível apenas com Node.js, Python e PHP.

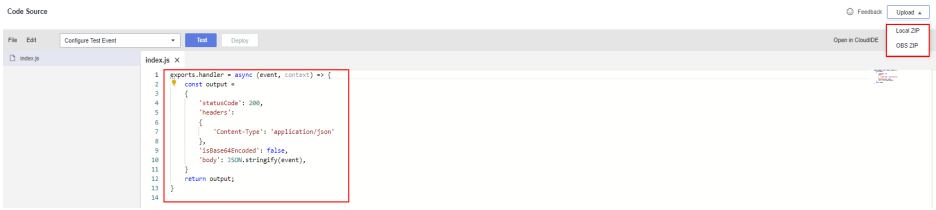
4. Clique em **Create Function**. Na página de guia **Code** exibida, continue a configurar o código.

Configuração do código

1. Você pode implantar o código com base no tempo de execução selecionado. Para mais detalhes, veja **Criação de um pacote de implantação**. Após a conclusão da implantação, clique em **Deploy**.

   Por exemplo, para implantar o código (**Figura 2-2**) no Node.js 10.16, você pode editar o código embutido, carregar um arquivo ZIP local ou carregar um arquivo ZIP do OBS.

   Figura 2-2 Implantação de código

2. Você pode modificar o código e clicar em **Deploy** para implantar o código novamente.

Visualização de Informações do Código

1. Visualize atributos de código.

   Atributos de código mostram o tamanho do código e a hora em que o código foi modificado.

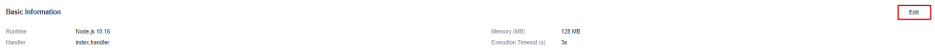
   Figura 2-3 Visualização de atributos de código

2. Visualize informações básicas.

Figura 2-4 mostra a memória padrão e o tempo limite de execução em cada tempo de execução. Você pode clicar em **Edit** para alternar para a página **Basic Settings** e modificar **Handler**, **Memory (MB)**, e **Execution Timeout (s)** conforme necessário. Para mais detalhes, veja **Configuração das configurações básicas**.

Figura 2-4 Edição de informações básicas



AVISO

Uma vez que uma função é criada, o tempo de execução não pode ser alterado.

Tabela 2-3 Informações básicas padrão de cada tempo de execução

Tempo de execução	Informações Básicas Padrão
JAVA	Memória (MB): 512 Manipulador: com.huawei.demo.TriggerTests.apigTest Tempo limite de execução (s): 15
Node.js	Memória (MB): 128 Handler: index.handler Tempo limite de execução (s): 3
Personalizado	Memória (MB): 128 Manipulador: bootstrap Tempo limite de execução (s): 3
PHP	Memória (MB): 128 Handler: index.handler Tempo limite de execução (s): 3
Python	Memória (MB): 128 Handler: index.handler Tempo limite de execução (s): 3

2.2.2 Criação de uma função de HTTP

NOTA

Este recurso é suportado apenas pelo FunctionGraph v2.

Visão geral

As funções de HTTP são projetadas para otimizar os serviços da web. Você pode enviar solicitações de HTTP para os URL para acionar a execução da função. As funções de HTTP suportam apenas gatilhos de APIG e APIC.

NOTA

- As funções de HTTP não fazem distinção entre linguagens de programação. O manipulador deve ser definido no arquivo **bootstrap**. Você pode escrever diretamente o comando de inicialização e permitir o acesso pela porta 8000. O endereço IP vinculado é **127.0.0.1**.
- O arquivo **bootstrap** é o arquivo de inicialização da função de HTTP. A função de HTTP só pode ler **bootstrap** como o nome do arquivo de inicialização. Se o nome do arquivo não for **bootstrap**, o serviço não poderá ser iniciado. Para obter mais informações, consulte **o exemplo do arquivo bootstrap**.
- As funções de HTTP suportam várias linguagens de programação.
- Esta seção usa Java como exemplo. Para usar outro tempo de execução, basta alterar o caminho do tempo de execução. O caminho do pacote de código não precisa ser alterado. Para os caminhos de outros tempos de execução, veja **Tabela 2-4**.

Pré-requisitos

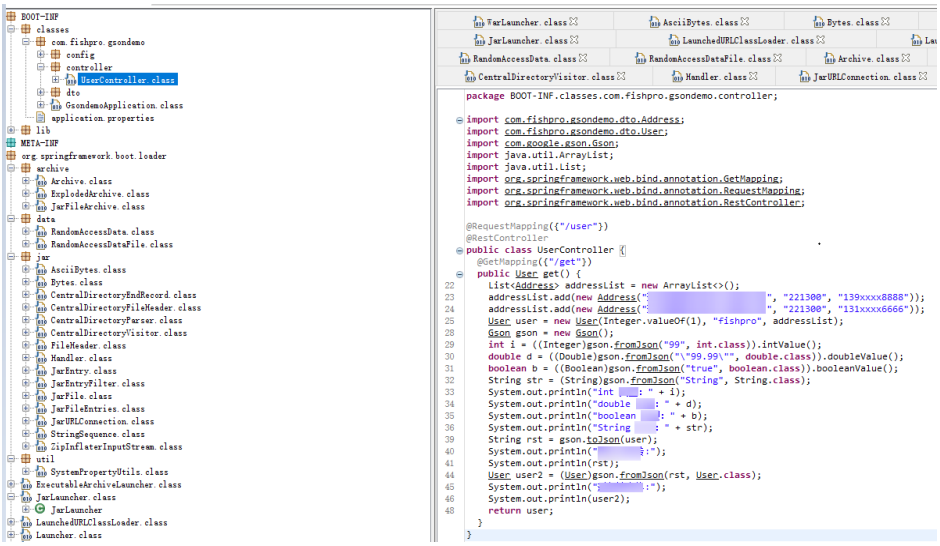
1. Você preparou um pacote Java JAR.
2. Você preparou um arquivo de bootstrap como o arquivo de inicialização da função de HTTP.

   Example

   O conteúdo do arquivo bootstrap é o seguinte:

   ```
   /opt/function/runtime/java8/rtsp/jre/bin/java -jar -Dfile.encoding=utf-8 /opt/function/code/gsondemo-0.0.1-SNAPSHOT.jar
   ```

   - **/opt/function/runtime/java8/rtsp/jre/bin/java**: Caminho Java.
   - **/opt/function/code**: caminho do pacote de código de função.
   - **gsondemo-0.0.1-SNAPSHOT.jar**: exemplo de pacote JAR. O caminho do serviço é **/user/get**.

   Para usar outro tempo de execução, altere o caminho do tempo de execução referindo-se a **Tabela 2-4**. O caminho do pacote de código não precisa ser alterado.

Tabela 2-4 Caminhos para diferentes tempos de execução

Tempo de execução	Caminho
Java 8	/opt/function/runtime/java8/rtsp/jre/bin/java
Java 11	/opt/function/runtime/java11/rtsp/jre/bin/java
Node.js 6	/opt/function/runtime/nodejs6.10/rtsp/nodejs/bin/node
Node.js 8	/opt/function/runtime/nodejs8.10/rtsp/nodejs/bin/node
Node.js 10	/opt/function/runtime/nodejs10.16/rtsp/nodejs/bin/node
Node.js 12	/opt/function/runtime/nodejs12.13/rtsp/nodejs/bin/node
Node.js 14	/opt/function/runtime/nodejs14.18/rtsp/nodejs/bin/node
Python 2.7	/opt/function/runtime/python2.7/rtsp/python/bin/python
Python 3.6	/opt/function/runtime/python3.6/rtsp/python/bin/python3
Python 3.9	/opt/function/runtime/python3.9/rtsp/python/bin/python3
PHP 7.3	/opt/function/runtime/php7.3/rtsp/php/bin/php

Procedimento

1. Crie uma função.
   a. Crie uma função de HTTP. Para mais detalhes, veja **Criação de uma função de evento**. Preste atenção especial aos seguintes parâmetros:
      - **FunctionGraph Version**: FunctionGraph v2
      - **Function Type**: Função de HTTP
   b. Carregue o código. Por exemplo, carregue um arquivo ZIP do OBS. Após a conclusão do upload, clique em **Deploy**.

      Compacte o pacote JAR e o arquivo de bootstrap e escolha **Upload** > **OBS ZIP**.

      Figura 2-5 Carregamento de um arquivo ZIP do OBS

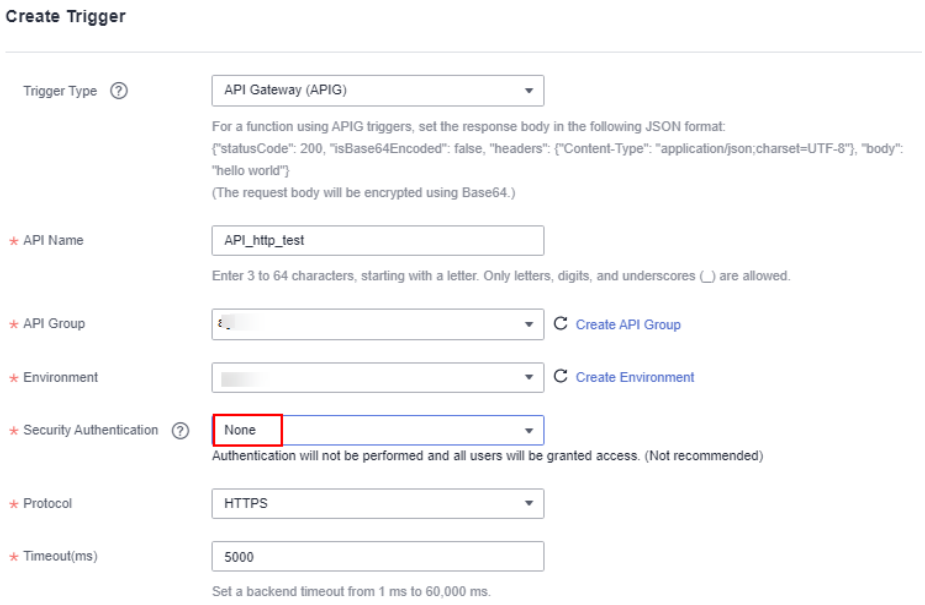
2. Crie um gatilho.

   NOTA

   As funções de HTTP suportam apenas gatilhos de APIG e APIC.

   a. Na página de detalhes da função, escolha **Configuration** > **Trigger** e clique em **Create Trigger**.
   b. Configure as informações do gatilho. A seguir, descrevemos como criar um gatilho de APIG. Para mais detalhes, veja **Uso de um gatilho de APIG (dedicado)**.

Figura 2-6 Criação de um gatilho



NOTA

Neste exemplo, **Security Authentication** é ajustada a **None**. Você precisa selecionar um modo de autenticação com base nos requisitos do site.

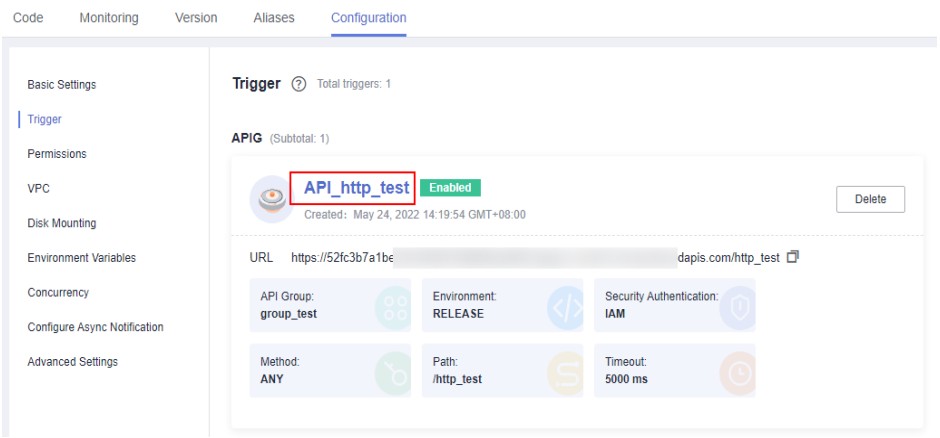
- **App**: Autenticação AppKey e AppSecret. Este modo é de alta segurança e é recomendado.
- **IAM**: Autenticação IAM. Esse modo concede permissões de acesso apenas a usuários do IAM e é de segurança média.
- **None**: Sem autenticação. Este modo concede permissões de acesso a todos os usuários.

c. Quando a configuração estiver concluída, clique em **OK**. Depois que o gatilho for criado, **API_test_http** será gerado no console do API Gateway.

3. Publique a API.

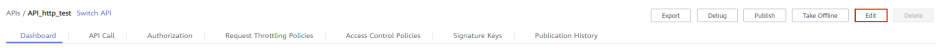
a. Na página de guia **Triggers**, clique em um nome de API para ir para a página de visão geral da API.

Figura 2-7 Gatilho de APIG



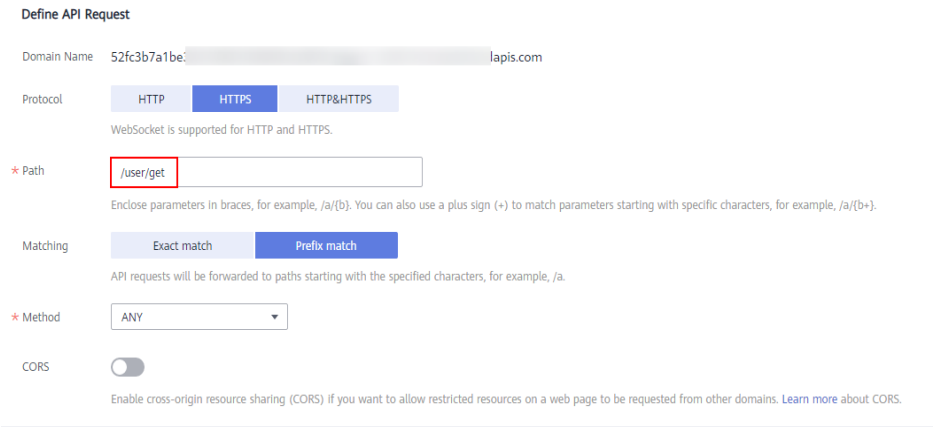
b. Clique em **Edit** no canto superior direito. A página **Basic Information** é exibida.

Figura 2-8 Edição de uma API



c. Clique em **Next**. Na página **Define API Request** exibida, altere **Path** para **/user/get** e clique em **Finish**.

Figura 2-9 Definição de uma solicitação de API



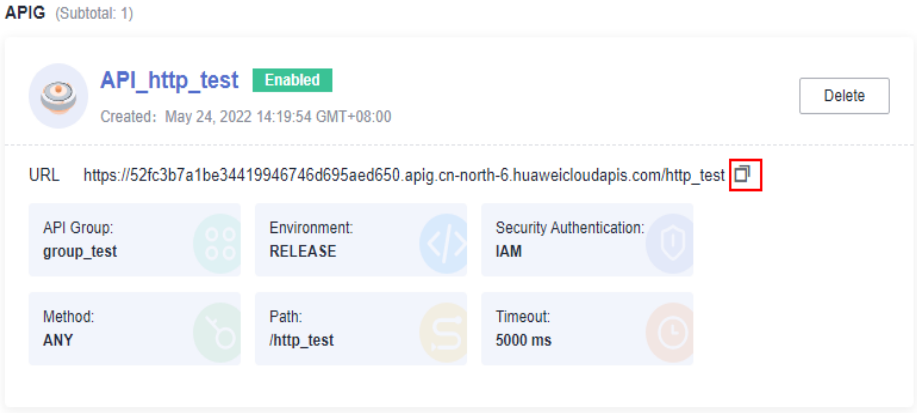
d. Clique em **Publish API**. Na página exibida, clique em **Publish**.

4. Acionar uma função.

a. Acesse o console do FunctionGraph, selecione **Functions** > **Function List** no painel de navegação e clique na função HTTP criada para acessar a página de detalhes.

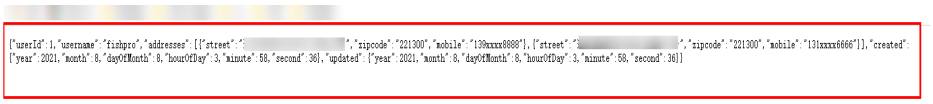
b. Escolha **Configuration** > **Trigger**, copie a URL e acesse-a usando um navegador.

Figura 2-10 Cópia do URL



c. Visualize o resultado da solicitação.

Figura 2-11 Visualização do resultado da solicitação



Cabeçalhos de Solicitação de Função Comum

A tabela a seguir lista os campos de cabeçalho de solicitação padrão de uma função HTTP.

Tabela 2-5 Campos de cabeçalho de solicitação padrão

Campo	Descrição
X-CFF-Request-Id	ID da solicitação atual
X-CFF-Memory	Memória alocada
X-CFF-Timeout	Duração do timeout da função
X-CFF-Func-Version	Versão da função
X-CFF-Func-Name	Nome da função
X-CFF-Project-Id	ProjectId
X-CFF-Package	Aplicativo ao qual a função pertence
X-CFF-Region	Região atual

2.3 Criação de uma função com o uso de um modelo

Visão geral

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

Criação de uma função

1. Efetue login no **Console do FunctionGraph**. No painel de navegação, escolha **Templates**.
2. Na página exibida, selecione o serviço **FunctionGraph**, selecione o modelo de **context-class-introduction** para Python 2.7 e clique em **Configure**.

   NOTA

   O template **context-class-introduction** para Python 2.7 é usado como exemplo. Você também pode selecionar outros modelos.
3. Depois de selecionar um modelo de função, o código interno e as configurações do modelo são carregados automaticamente. A página **Create Function**.
4. Defina o **Function Name** como **context**, selecione uma agência criada, 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:

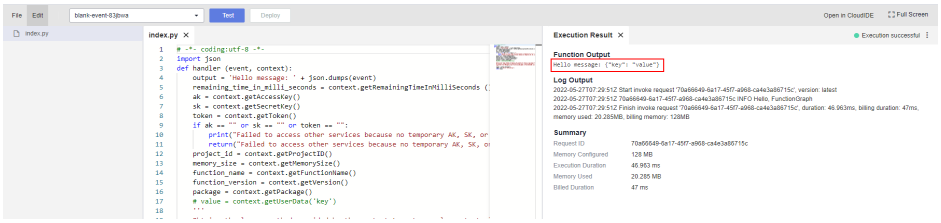
   ```
   Falha ao acessar outros serviços porque nenhum AK, SK, ou token
   temporário foi obtido. Por favor, defina uma agência.
   ```


5. Clique em **Save**.

Gatilho de uma função

1. Na página de guia **Code** da função de **context**, clique em **Test** no canto superior direito.
2. Na caixa de diálogo **Configure Test Event**, selecione **Blank Template** e clique em **Create**.
3. Clique em **Test**. Após a conclusão do teste, visualize o resultado do teste.

Figura 2-12 Resultado de execução bem sucedida



2.4 Implantação de uma função usando uma imagem de contêiner

Visão geral

Você pode empacotar e fazer upload diretamente de imagens de contêiner. As imagens são carregadas e iniciadas pela plataforma e podem ser chamadas de maneira semelhante às funções HTTP. Ao contrário do modo de upload de código anterior, você pode usar um pacote de código personalizado, que é flexível e reduz os custos de migração. Você pode criar imagens personalizadas para funções de evento e HTTP.

Os seguintes recursos são suportados:

- **Downloading images**

  As imagens são armazenadas no SWR e só podem ser baixadas por usuários com a permissão de **SWR Admin**. O FunctionGraph chamará a API do SWR para gerar e definir comandos de login temporários antes de criar instâncias.

- **Setting environment variables**

  Configurações da encriptação e variáveis de ambiente são suportadas. Para mais detalhes, veja **Configuração dos variáveis de ambiente**.

- **Attaching external data disks**

  Discos de dados externos podem ser anexados. Para mais detalhes, veja **Configuração da montagem de disco**.

- **Billing**

  Você não será cobrado quando fizer o download de imagens ou esperar que as imagens estejam prontas.

- **Reserved instances**

  Para obter detalhes, consulte a descrição sobre instâncias reservadas.

NOTA

Os contêineres do usuário serão iniciados usando UID 1003 e GID 1003, que são os mesmos que outros tipos de funções.

Pré-requisitos

Você criou uma agência com as permissões de **SWR Admin** consultando **Configuração das permissões de agência**. As imagens são armazenadas no SWR, e somente os usuários com essa permissão podem chamar e extrair imagens.

Procedimento

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Na página **Function List**, clique em **Create Function** no canto superior direito.
3. Selecione **Use container image**. Para mais detalhes, veja **Tabela 2-6**.

Figura 2-13 Criação de uma função usando uma imagem de contêiner

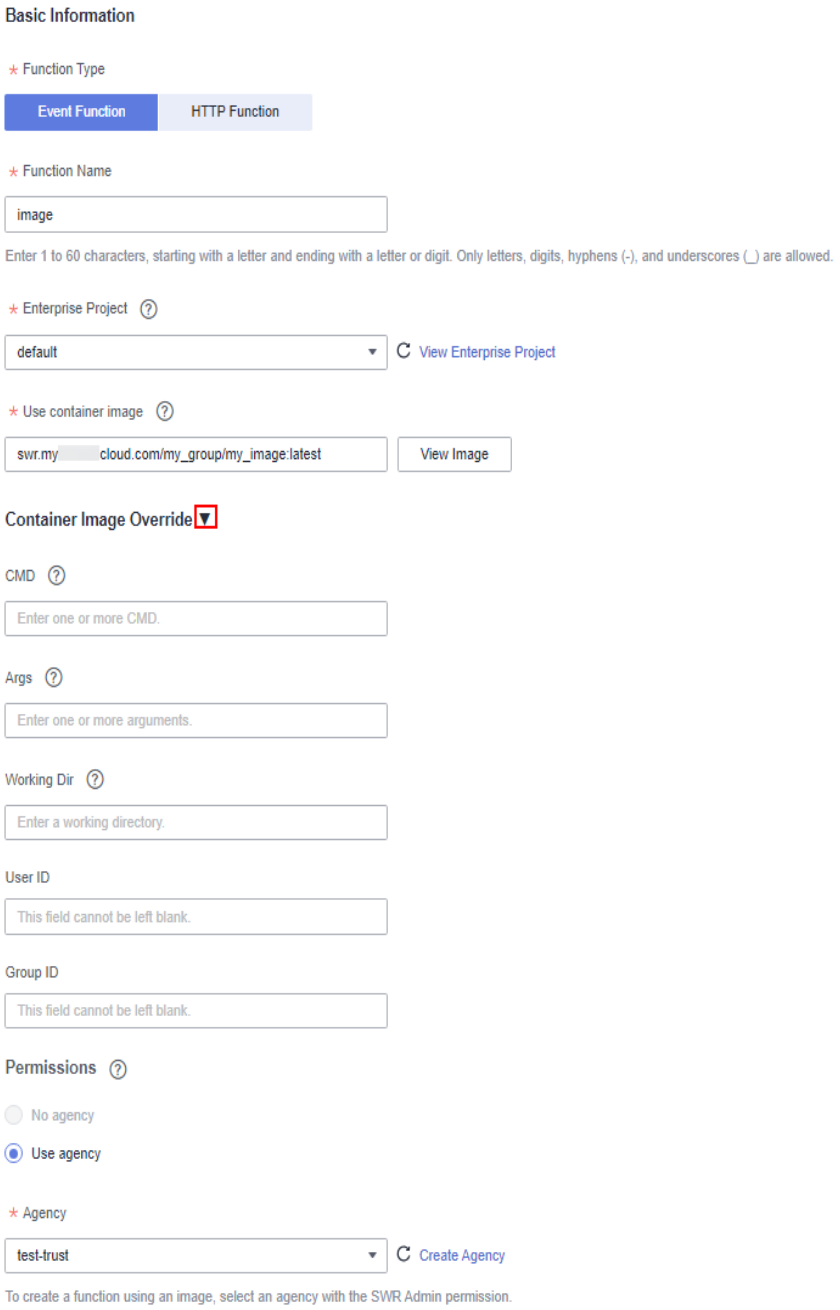


Tabela 2-6 Descrição do parâmetro

Parâmetro	Descrição
*Function Type	Selecione um tipo de função. ● Funções de evento: acionadas por gatilhos. ● Funções HTTP: acionadas assim que as solicitações HTTP são enviadas para os URL específicas. **NOTA** ● As funções HTTP não fazem distinção entre linguagens de programação. O manipulador deve ser definido no arquivo **bootstrap**. Você pode escrever diretamente o comando de inicialização e permitir o acesso pela porta 8000. ● As funções HTTP suportam apenas gatilhos APIG e APIC. ● Para obter detalhes sobre como usar funções HTTP, consulte **Criação de uma função de HTTP**.
*Function Name	Nome da função, que deve atender aos seguintes requisitos: ● Consiste de 1 a 60 caracteres e pode conter letras, dígitos, hífens (-) e sublinhados (_). ● Começa com uma letra e termina com uma letra ou dígito.
*Enterprise Project	Selecione um projeto empresarial criado e adicione a função a ele. Por padrão, **default** é selecionado. **NOTA** Se o EPS não estiver ativado, esse parâmetro não estará disponível. Para obter detalhes, consulte **Ativando a Função Projeto Corporativo**.
Container Image	Insira uma URL de imagem, ou seja, o local da imagem do contêiner. Você pode clicar em **View Image** para exibir imagens privadas e compartilhadas. Imagem no SWR, por exemplo, **swr.myhuaweicloud.com/my_group/my_image:latest**.
Container Image Override	● **CMD**: comando de inicialização do container. Exemplo: **/bin/sh**. Se nenhum comando for especificado, o ponto de entrada ou o CMD na configuração da imagem serão usados. Insira um ou mais comandos separados por vírgulas (,). ● **Args**: parâmetro de inicialização do container. Exemplo: **-args,value1**. Se nenhum argumento for especificado, CMD na configuração da imagem será usado. Insira um ou mais argumentos separados por vírgulas (,). ● **Working Dir**: diretório de trabalho que um contêiner executa. Se nenhum diretório for especificado, o diretório na configuração da imagem será usado. O diretório deve começar com uma barra (/). ● **User ID**: ID do usuário para executar a imagem. Se nenhum ID de usuário for especificado, o valor padrão **1003** será usado. ● **Group ID**: ID do grupo de usuários. Se nenhum ID de grupo de usuários for especificado, o valor padrão **1003** será usado.

Parâmetro	Descrição
Agency	Selecione uma agência com permissões de **SWR Admin**. Para criar uma agência, consulte.

NOTA

- **Command**, **Args**, e **Working dir** podem conter até 5120 caracteres.
- Quando uma função é executada pela primeira vez, a imagem é extraída do SWR e o contêiner é iniciado durante o início a frio da função, o que leva um certo período de tempo. Se não houver nenhuma imagem em um nó durante as partidas a frio subsequentes, uma imagem será extraída do SWR.
- A imagem deve ser pública.
- A porta de uma imagem de contêiner personalizada deve ser 8000.
- O pacote de imagens não pode exceder 2 GB. Para um pacote maior, reduza a capacidade. Por exemplo, monte os dados de uma biblioteca de perguntas em um contêiner onde os dados foram carregados anteriormente por meio de um sistema de arquivos externo.
- O FunctionGraph usa o LTS para coletar todos os logs que o contêiner envia para o console. Esses logs podem ser redirecionados e impressos no console por meio de saída padrão ou uma estrutura de log de código aberto. Os registros devem incluir a hora do sistema, o nome do componente, a linha de código e os principais dados, para facilitar a localização de falhas.

Código de exemplo

A seguir, **NodeJS Express** é usado como exemplo. Durante a inicialização da função, o FunctionGraph usa o método POST para acessar o caminho **/init** (opcional). Sempre que uma função é chamada, o FunctionGraph usa o método POST para acessar o caminho **/invoke**. A função obtém **context** de **req.headers**, obtém **event** de **req.body** e retorna estruturas de resposta HTTP como resultado.

```
const express = require('express');
const app = express();
const PORT = 8000;

app.post('/init', (req, res) => {
  res.send('Hello init\n');
});

app.post('/invoke', (req, res) => {
  res.send('Hello invoke\n');
});

app.listen(PORT, () => {
  console.log(`Listening on http://localhost:${PORT}`);
});
```


Diferenças entre imagens de contêiner, funções de evento e funções HTTP

Tabela 2-7 Diferenças entre imagens de contêiner, funções de evento e funções HTTP

Item de comparação	Função Evento	Função HTTP	Imagem do contentor
Idioma	Qualquer idioma	Qualquer idioma	Qualquer idioma
Modo de inicialização	Inicialização do processo	Inicialização do processo	Inicialização do container
Ambiente	Ambientes especificados	Ambientes especificados	Alta flexibilidade e personalização
Modo de conexão do FunctionGraph	Serve como um cliente HTTP	Serve como um servidor HTTP	Serve como um servidor HTTP
Modo de implantação de dependência	Camada	Camada	Na imagem
Latência de arranque a frio	Curto	Curto	Longitude

3 Configuração da função

3.1 Configuração da inicialização

Visão geral

Depois que uma função é criada, habilite a inicialização. Se a inicialização estiver ativada para uma função, um inicializador especificado será chamado para inicializar a função e, em seguida, um manipulador será chamado para processar solicitações.

Pré-requisitos

Você criou uma função.

Inicializando uma função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Clique na guia **Configuration** e escolha **Advanced Settings**.

Figura 3-1 Ativação da inicialização

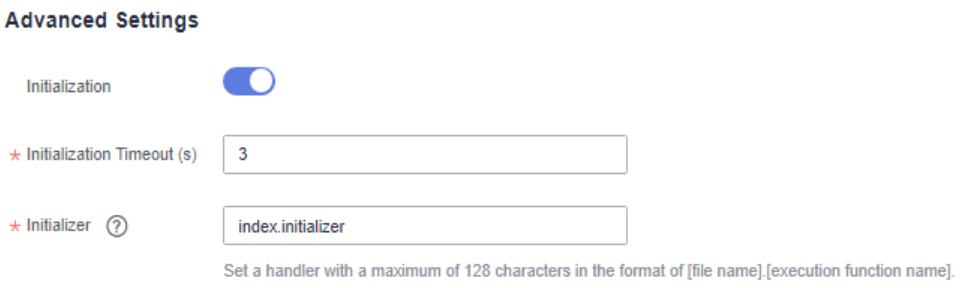


Tabela 3-1 Configuração do parâmetro

Parâmetro	Descrição
Inicialização	Ative a inicialização, se necessário.
Tempo limite (s) de inicialização	Duração máxima que a função pode ser inicializada. Defina este parâmetro se você ativar a inicialização da função. O valor varia de 1s a 300s.
Inicializador	Você pode ativar a inicialização da função na página de guia **Configuration**. O inicializador deve ser nomeado da mesma forma que o manipulador. Por exemplo, para uma função Node.js ou Python, defina um nome de inicializador no formato de *[nome do arquivo].[nome da função de inicialização]*. **NOTA** Este parâmetro não é necessário se a inicialização da função estiver desativada.

NOTA

- Defina o inicializador da mesma forma que o manipulador. Por exemplo, para uma função Node.js ou Python, defina um nome de inicializador no formato de *[nome do arquivo].[nome da função de inicialização]*.
- Para obter detalhes sobre a configuração do código de função, consulte **Criação de um pacote de implantação**.

----Fim

3.2 Configuração das configurações básicas

Visão geral

Depois que uma função é criada, **Memory**, **Handler**, e **Execution Timeout (s)** são definidos automaticamente com base no seu tempo de execução. Se necessário, modifique-os com base nesta seção.

Pré-requisitos

Você criou uma função.

Procedimento

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Escolha **Configuration** > **Basic Settings** e configure os parâmetros com base em **Tabela 3-2**. Os parâmetros marcados com um asterisco (*) são obrigatórios.

Tabela 3-2 Configurações básicas

Parâmetro	Descrição
App	Depois que uma função é criada, ela é automaticamente categorizada no aplicativo **default** e não pode ser alternada para outros aplicativos. **AVISO** Um aplicativo funciona como uma pasta. No futuro, as funções serão gerenciadas pelo rótulo para uma melhor experiência.
*Handler	● Para uma função Node.js, Python ou PHP, o manipulador deve ser nomeado no formato de *[nome do arquivo].[nome da função]*, que deve conter um ponto (.). Exemplo: **myfunction.handler** ● Para uma função Java, o manipulador deve ser nomeado no formato de *[nome do pacote].[nome do arquivo].[nome da função]*. Exemplo: **com.xxxxx.exp.Myfunction.myHandler** ● Para uma função Go, o manipulador deve ser nomeado no formato de *[nome do plug-in].[nome da função]*. O nome da função deve começar com uma letra maiúscula. O nome do manipulador pode conter no máximo 128 caracteres. Exemplo: **function.Handler** ● Para uma função C#, o manipulador deve ser nomeado no formato de *[.NET assembly file name]*::*[namespace e classe da função do manipulador]*::*[handler function name]*. Exemplo: **HelloCsharp::Example.Hello::Handler**
*Enterprise Project	Selecione um projeto empresarial criado e adicione a função a ele. Por padrão, **default** é selecionado. **NOTA** Se o EPS não estiver ativado, esse parâmetro não estará disponível. Para obter detalhes, consulte **Ativando a Função do Projeto Corporativo**.
*Execution Timeout (s)	Duração máxima que a função pode ser executada. Você pode definir esse parâmetro na página de guia **Configuration**. Se a execução demorar mais do que 90s, use a invocação assíncrona. O valor varia de 3s a 900s. **NOTA** Para definir o tempo limite de execução para 900s a 43.200s, entre em contato com o suporte técnico.
Memory (MB)	Memória de uma instância de função. Opções: 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2560, 3072, 3584, e 4096.
Description	Descrição da função, que não pode exceder 512 caracteres.

4. Clique em **Save**.

3.3 Configuração das permissões de agência

Visão geral

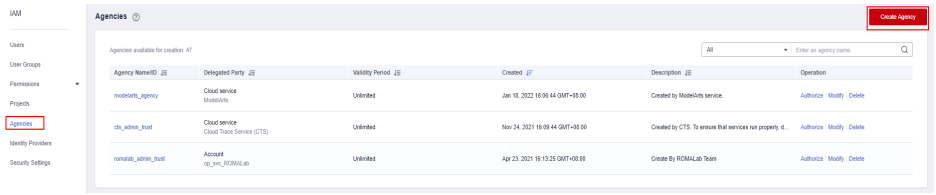
Serviços da HUAWEI CLOUD interagem entre si, e alguns serviços em nuvem dependem de outros serviços. Para delegar um serviço de nuvem para acessar outros serviços e executar O&M de recursos, crie uma agência para o serviço.

Criação de uma agência

Criar uma agência com base em **Criação de uma agência** e definir os seguintes parâmetros:

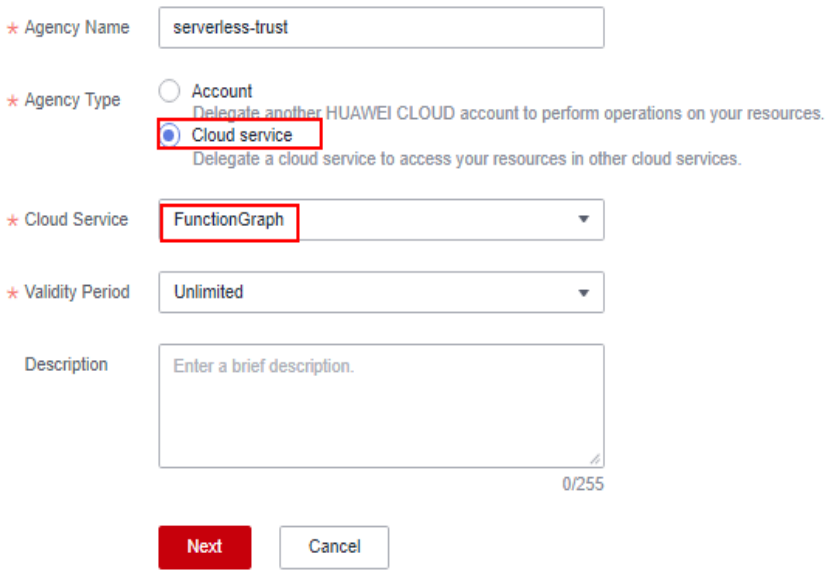
1. Faça login no console do IAM.
2. No console do IAM, escolha **Agencies** no painel de navegação e clique em **Create Agency** no canto superior direito.

Figura 3-2 Criação de uma agência



3. Configure a agência.

Figura 3-3 Configuração de informações básicas



- Para **Agency Name**, insira **serverless-trust**.
- Para **Agency Type**, selecione **Cloud service**.
- Para **Cloud Service**, selecione **FunctionGraph**.

- Para **Validity Period**, selecione **Unlimited**.
- **Description**: Insira uma descrição.

4. Clique em **Next**. Na página exibida, pesquise as permissões a serem adicionadas na caixa de pesquisa à direita e selecione as permissões. A permissão **Tenant Administrator** é usada como um exemplo.

Figura 3-4 Selecionamento de políticas

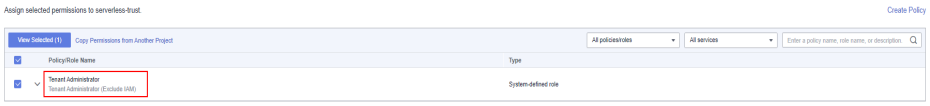


Tabela 3-3 Exemplo de permissões de agência

Nome da política	Cenário
Administrador de locatários	Administrador de todos os serviços de nuvem, exceto o IAM. Este usuário pode executar qualquer operação em todos os recursos de nuvem da empresa.

5. Clique em **Next** e selecione o escopo, por exemplo, **Region-specific project**.

Configuração de uma agência

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Escolha **Configuration** > **Permissions**, click **Create Agency** com base nos requisitos do site consultando **2**–**5**.

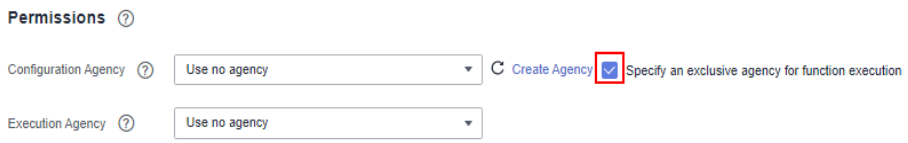
Tabela 3-4 Parâmetros de configuração da agência

Parâmetro	Descrição
Configuration Agency	Seleciona uma função que você criou.
Execution Agency	Obrigatório se você selecionar **Specify an exclusive agency for function execution**.

NOTA

- Para garantir um desempenho ideal, selecione **Specify an exclusive agency for function execution** e defina agências diferentes para configuração e execução de funções. Você também pode usar nenhuma agência ou especificar a mesma agência para ambos os fins. **Figura 3-5** mostra as opções da agência.

  Figura 3-5 Definição de agências

- **Configuration Agency**: Por exemplo, para criar gatilhos do Serviço de Ingestão de Dados (DIS), especifique primeiro uma agência com permissões DIS. Se tal agência não for especificada ou a agência especificada não existir, nenhum gatilho DIS pode ser criado.
- **Execution Agency**: Esse tipo de agência permite que você obtenha um token e AK/SK do contexto no manipulador de funções para acessar outros serviços em nuvem.

4. Clique em **Save**.

Configuração de ações em cenários comuns

Antes de usar o FunctionGraph **crie uma agência** e selecione a ação necessária consultando **Tabela 3-5**.

Tabela 3-5 Ações comuns

Cenário	Ação	Descrição
Uso de uma imagem personalizada	Administrador do SWR	Administrador do SWR: administrador que tem todas as permissões para o serviço Repositório do SoftWare para Container (SWR). Para obter detalhes sobre como criar uma imagem personalizada, consulte **Implantação de uma função usando uma imagem de contêiner**.
Montagem de um sistema de arquivos SFS Turbo	Administrador do SFS ou administrador do Locatário	Administrador SFS: administrador que tem todas as permissões para o serviço SFS. Administrador do locatário: administrador para todos os serviços de nuvem, exceto o IAM. Este usuário pode executar qualquer operação em todos os recursos de nuvem da empresa. Para obter detalhes sobre como montar um sistema de arquivos SFS Turbo, consulte **Montagem de um sistema de arquivos SFS turbo**.

Cenário	Ação	Descrição
Montagem de um diretório compartilhado do ECS	Locatário convidado e administrador VPC	Inquilino convidado: usuário com permissões somente leitura para todos os serviços em nuvem (exceto o IAM) Administrador de VPC: administrador de rede A agência deve ter pelo menos as permissões de locatário convidado e administrador de VPC. Para obter detalhes sobre como montar um diretório compartilhado do ECS, consulte **Montagem de um diretório compartilhado do ECS**.
Uso de um gatilho de DIS	Administrador do DIS	Administrador que tem todas as permissões para o serviço DIS. Para obter detalhes sobre como criar um gatilho DIS, consulte **Uso de um gatilho de DIS**.
Configuração do acesso VPC entre domínios	Administrador da VPC	Os usuários com permissões de **VPC Administrator** podem executar qualquer operação em todos os recursos de nuvem da VPC. Para configurar o acesso entre as VPC, especifique uma agência com permissões de gerenciamento das VPC. Para obter detalhes sobre como configurar o acesso VPC entre domínios, consulte **Configuração da rede**.
Leitura de recursos DNS	DNS ReadOnlyAccess	ReadOnlyAccess DNS: usuário com as permissões apenas para visualizar recursos DNS. Para chamar uma API DNS para resolver nomes de domínio privados, especifique uma agência com permissões para ler recursos DNS.
Criação de um bucket e um gatilho do OBS	Administrador de locatários	Administrador do locatário: administrador para todos os serviços de nuvem, exceto o IAM. Este usuário pode executar qualquer operação em todos os recursos de nuvem da empresa. Para obter detalhes sobre como criar um gatilho do OBS, consulte **Uso de um gatilho de OBS**.

Modificação de uma agência

Modificação de uma agência: Você pode modificar as permissões, o período de validade e a descrição de uma agência no console do IAM.

⚠ CUIDADO

Depois que uma agência é modificada, leva cerca de 10 minutos para que a modificação (por exemplo, **context.getToken**) entre em vigor.

3.4 Configuração da rede

Acesso Público

Por padrão, as funções podem acessar serviços em redes públicas. Se o serviço de rede pública de destino exigir a verificação da lista branca usando um endereço IP fixo, habilite o acesso da VPC, configure um gateway NAT público para a VPC e vincule um Elastic IP (EIP) ao gateway. Para obter detalhes, consulte Configuração de um endereço do IP público fixoAccessing the Internet from a VPC

Configuração do acesso à VPC

As funções podem acessar recursos em uma VPC vinculada a ela. Se uma função precisar de acesso público e VPC, configure um gateway NAT público para a VPC e vincule um EIP ao gateway. Para mais detalhes, consulte Configuração de um endereço do IP público fixoAccessing the Internet from a VPC.

Required Permissions

Configure uma agência fazendo referência a **Configuração das permissões de agência**.

- Permissões para acesso à VPC: uma agência com a permissão de **VPC Administrator** ou com o mínimo de permissões listadas em **Tabela 3-6**

  Tabela 3-6 Mínimo de permissões necessárias

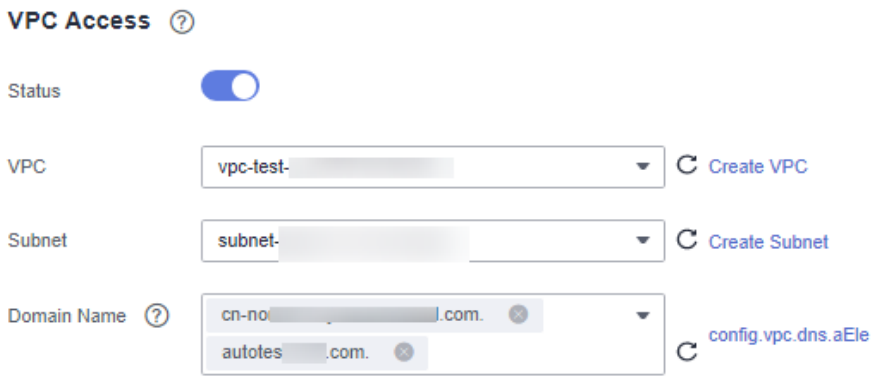
Permissão	Ação
Exclusão de uma porta	vpc:ports:delete
Consulte de uma porta	vpc:ports:get
Criação de um port	vpc:ports:create
Consulte de uma VPC	vpc:vpcs:get
Consulte de uma sub-rede	vpc:subnets:get

- Permissões para resolução de nomes de domínio privado: uma agência com a permissão **DNS ReadOnlyAccess**

Procedure

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Escolha **Configuration** > **VPC**, ative o acesso à VPC e selecione uma VPC e uma sub-rede.

Figura 3-6 Configuração do acesso à VPC



NOTA

1. Para obter detalhes sobre como criar uma VPC e uma sub-rede, consulte Criando uma VPC.
2. Você pode vincular funções em um projeto a até quatro sub-redes diferentes em quaisquer VPC. (Cada projeto tem uma ID de projeto exclusivo de 32 dígitos, que é alocado quando sua conta é criada. Os ID de projeto da sua conta e do usuário do IAM são os mesmos.)

4. (Opcional) Configurar nomes de domínio.

   Insira um ou mais nomes de domínio privado da VPC para que a função possa usá-los para acessar recursos nesta VPC. Veja **Figura 3-6**.

   NOTA

   1. Para obter detalhes sobre como criar um nome de domínio privado, consulte Criação de uma zona privada.
   2. As funções podem resolver somente nomes de domínio do tipo de conjunto de registros A. Para obter detalhes sobre como adicionar um conjunto de registros, consulte Tipos de conjunto de registros e regras de configuração.
   3. **How Does a Function Access Redis?**

5. Clique em **Save**.

Configuração de um endereço do IP público fixo

Se uma função precisar acessar recursos de rede pública em uma VPC ou exigir um endereço IP público fixo, configure um gateway NAT público para a VPC e vincule um EIP ao gateway.

Prerequisites

1. Você criou uma VPC e uma sub-rede de acordo com Criando um VPC
2. Você obteve um EIP de acordo com Atribuição de um EIP.

Procedure

1. Entre ao console do GatewayCreate do NAT, e clique em **Buy Public NAT Gateway**.
2. Na página exibida, insira as informações do gateway, selecione uma VPC (por exemplo, **vpc-01**) e uma sub-rede, confirme e envie as configurações. Para obter detalhes, consulte Compra de um gateway NAT público.
3. Clique no nome do gateway NAT público . Na página de detalhes exibida, clique em Adicionar regra de SNAT, defina a regra e clique em **OK**.

Restrições de rede

A largura de banda de acesso público padrão é compartilhada entre os locatários. Se sua função concluir operações de alta largura de banda, como download de arquivos grandes e rastreamento de recursos, habilite o acesso da VPC para a função, adicione um gateway NAT público e vincule um EIP com uma largura de banda exclusiva a ele.

3.5 Configuração da montagem de disco

Cenários

AVISO

Antes de montar um sistema de arquivos, habilite as portas 111, 445, 2049, 2051, 2052 e 20048 para o grupo de segurança especificado para o sistema de arquivos. Para obter detalhes, consulte **Quais recursos o SFS ocupa?**

Você pode montar sistemas de arquivos em sua função para fornecer armazenamento de arquivos escalável. A função pode então ler e gravar dados nos sistemas de arquivos como faria em sistemas de arquivos locais. Cada sistema de arquivos pode ser compartilhado por diferentes funções e instâncias. Você só precisa especificar informações como sistemas de arquivos e caminhos de acesso a funções.

O FunctionGraph é compatível com os seguintes tipos de sistemas de arquivos:

- SFS Turbo

  O SFS Turbo suporta as seguintes classes de armazenamento: Padrão, Padrão-Aprimorado, Desempenho e Desempenho-Aprimorado. O SFS Turbo é expansível até 320 TB e fornece armazenamento de arquivos compartilhados totalmente hospedado. Ele apresenta alta disponibilidade e durabilidade e suporta grandes quantidades de pequenos arquivos e aplicativos que exigem baixa latência e altas operações de entrada/saída por segundo (IOPS). O SFS Turbo é adequado para sites de alto desempenho, armazenamento de registrações, compactação e descompactação, DevOps e aplicativos em contêiner.

- ECS

  Um diretório em um ECS é especificado como um sistema de arquivos compartilhado (consulte **Montagem de um diretório compartilhado do ECS**) usando o serviço NFS (Network File System). O diretório pode então ser montado em uma função na mesma VPC que o ECS para que a função possa ler e gravar dados no diretório. Os sistemas de arquivos ECS possibilitam a expansão dinâmica dos recursos de computação. Esse tipo de sistema de arquivos é adequado para cenários de baixa demanda de serviço.

Benefícios do uso destes sistemas de arquivos:

- O espaço de execução da função pode ser muito expandido comparando com **/tmp**.
- Um sistema de arquivos pode ser compartilhado por várias funções.
- Os recursos de computação do ECS podem ser expandidos dinamicamente e os recursos de armazenamento existentes do ECS podem ser usados para obter um desempenho de computação mais forte.

> **NOTA**
>
> Você pode escrever arquivos temporários no diretório **/tmp** O tamanho total desses arquivos não pode exceder 512 MB.

Criação de uma agência

Antes de adicionar sistemas de arquivos a uma função, especifique uma agência com permissões para acessar os serviços do sistema de arquivos da função.

Há um limite para o número máximo de agências que você pode criar, e as agências de serviços em nuvem não podem ser modificadas. Portanto, é recomendável criar uma agência com permissões de alto nível, por exemplo, **Tenant Administrator**, para permitir que uma função acesse todos os recursos na região selecionada. Para obter mais informações, consulte **Configuração das permissões de agência**.

Montagem de um sistema de arquivos SFS turbo

Setting an Agency

Antes de montar um sistema de arquivos SFS Turbo em uma função, especifique uma agência que tenha recebido permissões de **SFS Administrator** e **VPC Administrator** para a função. Se nenhuma agência estiver disponível, crie uma no IAM.

Configuring VPC Access

Um sistema de arquivos SFS Turbo é acessível somente na VPC onde foi criado. Antes de montar esse sistema de arquivos em uma função, ative o acesso da VPC para a função.

1. No console do SFS, obtenha as informações sobre a VPC e a sub-rede onde um sistema de arquivos deve ser montado em sua função. Para obter detalhes, consulte **Gerenciamento do sistema de arquivos**.
2. Habilite o acesso à VPC referindo-se à **Configuração da rede** e acesse à VPC e sub-rede obtidas na **1**.

Mounting an SFS Turbo File System

Os sistemas de arquivos SFS Turbo podem ser montados da mesma forma que os sistemas de arquivos SFS. Selecione um sistema de arquivos e defina o caminho de acesso.

Montagem de um diretório compartilhado do ECS

Specifying an Agency

Antes de montar um diretório compartilhado do ECS em uma função, especifique uma agência que tenha recebido permissões de **Tenant Guest** e **VPC Administrator** para a função. Se nenhuma agência estiver disponível, crie uma no IAM.

Configuring VPC Access

Antes de adicionar um diretório compartilhado do ECS, especifique a VPC em que o ECS está implantado. Veja as informações sobre a VPC na página de detalhes do ECS. Clique no nome da VPC para acessar a página de detalhes da VPC e exibir a sub-rede.

Defina a VPC adquirida e a sub-rede para a função.

Mounting an ECS Directory

Insira um diretório compartilhado e um caminho de acesso à função.

Figura 3-7 Configuração do caminho

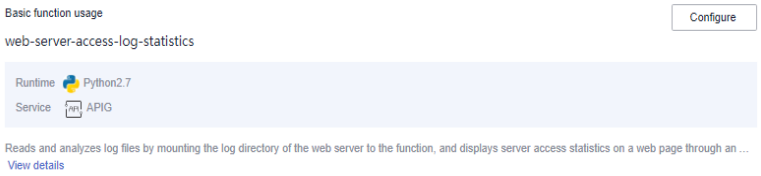
Operações de acompanhamento

Uma função pode ler e gravar dados em um caminho de acesso da mesma forma que no sistema de arquivos montado.

Os registrações de função podem ser persistidos configurando o caminho do registração como um subdiretório no caminho de acesso.

Crie uma função usando o modelo "Web-Server-Access-Log-Statistics" para analisar os registrações do servidor web.

Figura 3-8 Modelo de função



Criação de um diretório compartilhado NFS no ECS

1. **Linux**
   - CentOS, SUSE, EulerOS, Fedora, ou openSUSE
     i. Configurar um repositório YUM.

     1. Crie um arquivo chamado **euleros.repo** no diretório **/etc/yum.repos.d**. Certifique-se de que o nome do arquivo deve terminar com **.repo**.

     2. Execute o seguinte comando para entrar no **euleros.repo** e editar a configuração:

     ```
     vi /etc/yum.repos.d/euleros.repo
     ```

     A configuração do EulerOS 2.0 SP3 YUM é a seguinte:

     ```
     [base]
     name=EulerOS-2.0SP3 base
     baseurl=http://repo.huaweicloud.com/euler/2.3/os/x86_64/
     enabled=1
     gpgcheck=1
     gpgkey=http://repo.huaweicloud.com/euler/2.3/os/RPM-GPG-KEY-EulerOS
     ```

     A configuração do EulerOS 2.0 SP5 YUM é a seguinte:

     ```
     [base]
     name=EulerOS-2.0SP5 base
     baseurl=http://repo.huaweicloud.com/euler/2.5/os/x86_64/
     enabled=1
     gpgcheck=1
     gpgkey=http://repo.huaweicloud.com/euler/2.5/os/RPM-GPG-KEY-EulerOS
     ```


NOTA

Descrição

name: nome do repositório

baseurl: URL do repositório

- Endereço de rede baseado em **http://path/to/repo**
- Endereço do repositório local file: **file:///path/to/local/repo**

gpgcheck: indica se deve habilitar o GNU privacy guard (GPG) para verificar a validade e a segurança dos recursos de pacotes RPM. **0**: A verificação GPG está desabilitada. **1**: A verificação GPG está ativada. Se esta opção não for especificada, a verificação GPG é ativada por padrão.

3. Salve as configurações.

4. Execute o seguinte comando para limpar o cache:

```
yum clean all
```

ii. Execute o seguinte comando para instalar nfs-utils:

```
yum install nfs-utils
```

iii. Crie um diretório compartilhado.

Quando você abrir **/etc/exports** e precisar criar o diretório compartilhado **/sharedata**, adicione a seguinte configuração:

/sharedata 192.168.0.0/24(rw,sync,no_root_squash)

NOTA

A configuração anterior é usada para compartilhar o diretório **/sharedata** com outros servidores na sub-rede **192.168.0.0/24**.

Depois que o comando anterior for executado, execute o comando **exportfs -v** para exibir o diretório compartilhado e verificar se a configuração foi bem-sucedida.

iv. Execute os seguintes comandos para iniciar o serviço NFS:

```
systemctl start rpcbind
início do serviço nfs
```

v. Crie outro diretório compartilhado.

Por exemplo, para criar o diretório **/home/myself/download**, adicione a seguinte configuração ao **/etc/exports**:

/home/myself/download 192.168.0.0/24(rw,sync,no_root_squash)

Reinicie o serviço NFS.

```
service nfs restart
```

Como alternativa, execute o seguinte comando sem reiniciar o serviço NFS:

```
exportfs -rv
```

vi. (Opcional) Habilita a inicialização automática do serviço rpcbind.

Execute o seguinte comando:

```
systemctl habilitar rpcbind
```

- **Ubuntu**

i. Execute os seguintes comandos para instalar o nfs-kernel-server:

```
sudo apt-get update
sudo apt install nfs-kernel-server
```

ii. Crie um diretório compartilhado.

Quando você abrir **/etc/exports** e precisar criar o diretório compartilhado **/sharedata**, adicione a seguinte configuração:

/sharedata 192.168.0.0/24(rw,sync,no_root_squash)

> **NOTA**
>
> A configuração anterior é usada para compartilhar o diretório **/sharedata** com outros servidores na sub-rede **192.168.0.0/24**.
>
> Depois que o comando anterior for executado, execute o comando **exportfs -v** para exibir o diretório compartilhado e verificar se a configuração foi bem-sucedida.

iii. Iniciar o serviço NFS.

```
service nfs-kernel-server restart
```

iv. Crie outro diretório compartilhado.

Por exemplo, para criar o diretório **/home/myself/download**, adicione a seguinte configuração ao **/etc/exports**:

/home/myself/download 192.168.0.0/24(rw,sync,no_root_squash)

Reinicie o serviço NFS.

```
reinício do serviço nfs
```

Como alternativa, execute o seguinte comando sem reiniciar o serviço NFS:

```
exportfs -rv
```

2. **Windows**

1. Instale o servidor NFS.

Software pago: haneWIN. Faça o download do software no **site oficial da haneWIN**.

Software livre: FreeNFS and WinNFSd. Faça o download do software no **site da SourceForge**.

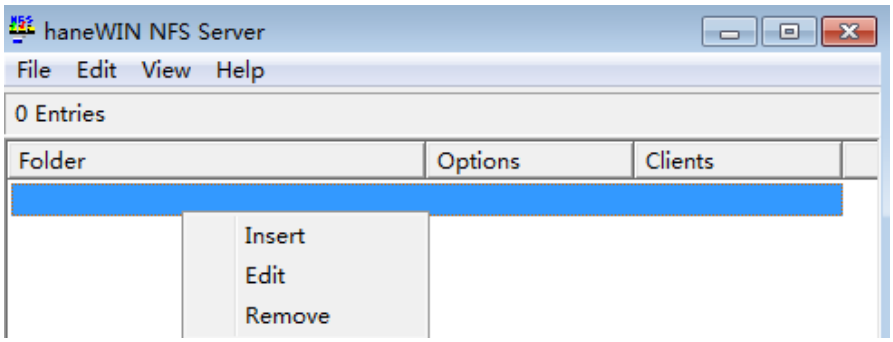
2. Ative a função NFS.

- No caso do WinNFSd, consulte **Configuração do WinNFSd**.
- No caso de haneWIN, execute as seguintes etapas:

i. Execute **o nfsctl.exe** como administrador do Windows.

ii. Clique com o botão direito do mouse na área em branco e escolha **Inserir** no menu de atalho.

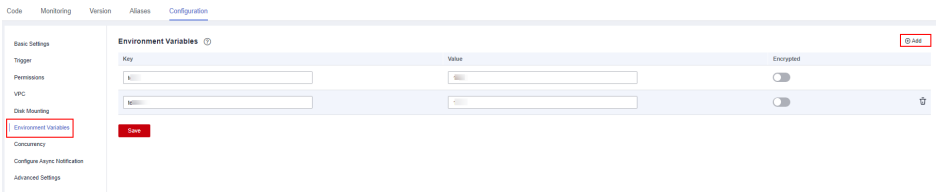
Figura 3-9 Inserir



3.6 Configuração dos variáveis de ambiente

Você pode configurar configurações de criptografia e variáveis de ambiente para passar configurações dinamicamente para o código de função e bibliotecas sem alterar o código.

Figura 3-10 Adicionamento de variáveis de ambiente



Por exemplo, para Node.js, configurações de criptografia e valores de variáveis de ambiente podem ser obtidos de **getUserData(string key)** em **Context**. Para obter detalhes, consulte **Desenvolvimento de funções no Node.js** Para obter detalhes sobre outros tempos de execução, consulte Desenvolvimento de funções.

ATENÇÃO

- Variáveis de ambiente e configurações de criptografia são pares chave-valor definidos pelo usuário que armazenam configurações de função. As teclas podem conter letras, dígitos e sublinhados (_), e devem começar com uma letra.
- O comprimento total da chave e do valor não pode exceder 2048 caracteres.

Parâmetros predefinidos

A seguir, lista os parâmetros predefinidos. Não configure variáveis de ambiente com os mesmos nomes de qualquer um desses parâmetros.

RUNTIME_PROJECT_ID

RUNTIME_FUNC_NAME

RUNTIME_FUNC_VERSION

RUNTIME_PACKAGE

RUNTIME_HANDLER

RUNTIME_TIMEOUT

RUNTIME_USERDATA

RUNTIME_CPU

RUNTIME_MEMORY

RUNTIME_MAX_RESP_BODY_SIZE

RUNTIME_INITIALIZER_HANDLER

RUNTIME_INITIALIZER_TIMEOUT

RUNTIME_ROOT

RUNTIME_CODE_ROOT

RUNTIME_LOG_DIR

RUNTIME_SHARE_DIR

RUNTIME_TMP_DIR

RUNTIME_SOCKET_ADDRESS

RUNTIME_USE_REPLAYABLE

RUNTIME_FSS_REPOSITORY_ROOT

_APP_SHARE_DIR

Exemplo

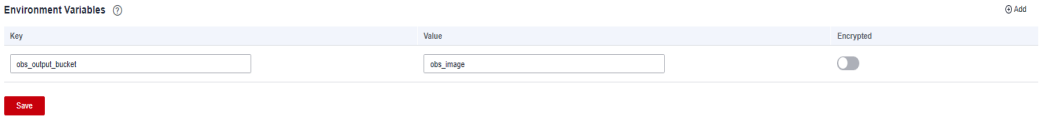
Você pode usar variáveis de ambiente para configurar em qual diretório instalar os arquivos, onde armazenar as saídas e como armazenar as configurações de conexão e registro. Essas configurações são dissociadas da lógica do aplicativo, para que você não precise atualizar seu código de função quando alterar as configurações.

No snippet de código a seguir, **obs_output_bucket** é o intervalo usado para armazenar imagens processadas.

```
def handler(event, context):
    srcBucket, srcObjName = getObsObjInfo4OBSTrigger(event)
    obs_address = context.getUserData('obs_address')
    outputBucket = context.getUserData('obs_output_bucket')
    if obs_address is None:
        obs_address = '{obs_address_ip}'
    if outputBucket is None:
        outputBucket = 'casebucket-out'

    ak = context.getAccessKey()
    sk = context.getSecretKey()

    # download file uploaded by user from obs
    GetObject(obs_address, srcBucket, srcObjName, ak, sk)

    outFile = watermark_image(srcObjName)

    # Upload converted files to a new OBS bucket.
    PostObject (obs_address, outputBucket, outFile, ak, sk)

    return 'OK'
```

Uso do variável de ambiente **obs_output_bucket**, você pode definir de forma flexível o bucket do OBS usado para armazenar imagens de saída.

Figura 3-11 Variáveis de ambiente



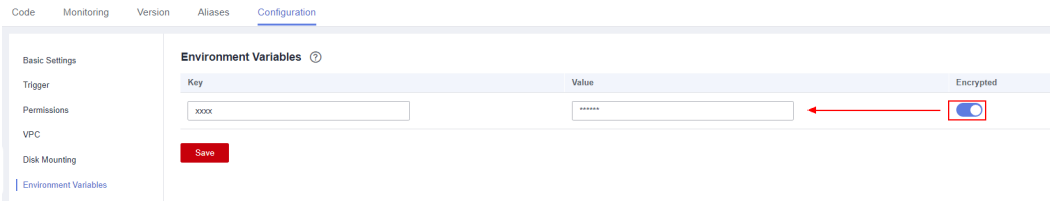
3.7 Configurações da encriptação

Configurações da encriptação referem-se a variáveis de ambiente que são criptografadas, que podem ser definidas na página de detalhes da função. Para obter detalhes sobre como adicionar e excluir variáveis, consulte **Configuração dos variáveis de ambiente**.

AVISO

Depois que a encriptação estiver ativada, os pares de valor-chave serão criptografados no console. Eles também serão criptografados durante a transmissão.

Figura 3-12 Configurações da encriptação



3.8 Configuração da notificação de execução assíncrona

Depois que uma função for criada, configure a notificação de execução assíncrona para enviar os resultados da execução ao serviço de destino.

NOTA

1. Se uma mensagem indicando permissões insuficientes for exibida quando você configurar a notificação de execução assíncrona, adicione a permissão de **FunctionGraph Administrator**. Para obter detalhes, consulte **Criação de um usuário e concessão de permissões**.
2. Defina uma agência que permita que o FunctionGraph acesse o serviço de destino.
3. Para evitar invocação cíclica, não defina duas funções como alvos de execução assíncronas uma da outra.

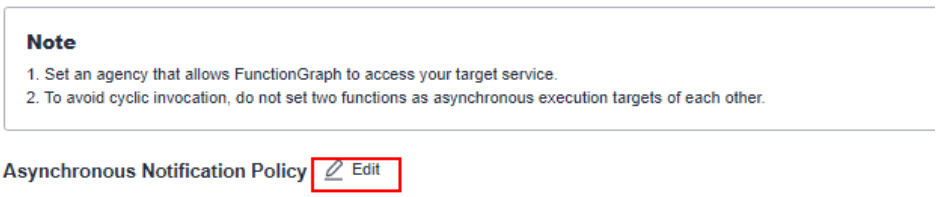
Procedimento

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **Configure Async Notification**. Na página exibida, clique em **Edit** ao lado de **Asynchronous Notification Policy**.

Figura 3-13 Configuração de uma política de notificação assíncrona



Passo 4 Defina parâmetros referindo-se a **Tabela 3-7**. Por exemplo, especifique **FunctionGraph** para **Target Service**.

Figura 3-14 Definição de parâmetros

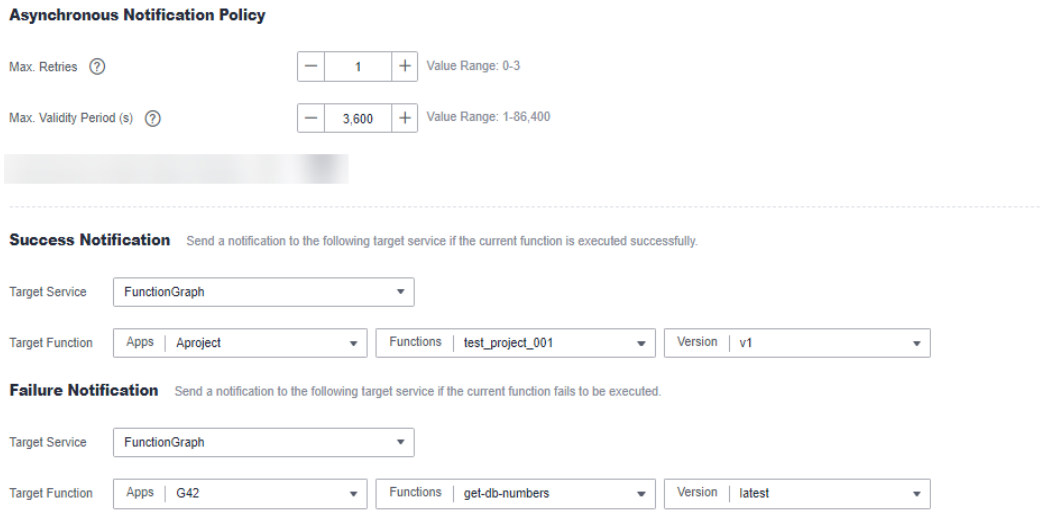


Tabela 3-7 Descrição do parâmetro

Parâmetro	Descrição
Asynchronous Execution Notification Policy	● **Max. Retries**: número máximo de tentativas quando a invocação assíncrona falha. Intervalo do valor: 0 – 3. Valor padrão: **1**. ● **Max. Validity Period (s)**: duração máxima de uma mensagem em segundos. Intervalo do valor: 1 – 86.400.
Success Notification	**Target Service**: para o qual uma notificação será enviada se uma função for executada com sucesso. 1. FunctionGraph 2. OBS 3. DIS 4. SMN
Failure Notification	**Target Service**: para o qual será enviada uma notificação se uma função não for executada. 1. FunctionGraph 2. OBS 3. DIS 4. SMN

Passo 5 Clique em **OK**.

----Fim

Descrição de configuração

Para obter detalhes sobre como definir o alvo para invocação assíncrona, consulte **Tabela 3-8**. A seguir, mostramos um exemplo:

```
{
        "timestamp": "2020-08-20T12:00:00.000Z",
    "request_context": {
            "request_id": "1167bf8c-87b0-43ab-8f5f-26b16c64f252",
            "function_urn": "urn:fss:xx-xxxx-x:xxxxxxx:function:xxxx:xxxx:latest",
            "condition": "",
            "approximate_invoke_count": 0
     },
        "request_payload": "",
        "response_context": {
            "status_code": 200,
            "function_error": ""
      },
     "response_payload": "hello world!"
}
```

Tabela 3-8 Descrição do parâmetro

Parâmetro	Descrição
timestamp	Hora em que a invocação começa.
request_context	Solicite contexto.
request_context.request_id	ID de uma solicitação de invocação assíncrona.
request_context. function_urn	URN da função que deve ser executada de forma assíncrona.
request_context.condition	Tipo de erro de invocação.
request_context. approximate_invoke_count	Número de tempos de chamada assíncrona. Se o valor for maior que 1, a execução da função foi repetida.
request_payload	Carga de solicitação original.
response_context	Contexto de resposta.
response_context.statusCode	Código retornado após a invocação da função. Se o código não for 200, ocorreu um erro de sistema.
response_context.function_error	Informações de erro de invocação.
response_payload	Payload retornado após a execução da função.

3.9 Configuração da multi-corrência de instância única

NOTA

Este recurso é suportado apenas pelo FunctionGraph v2.

Visão geral

A multi-concorrência de instância única permite que uma instância processe um número especificado de solicitações simultaneamente. Esse recurso tem as seguintes vantagens:

- Melhora a eficiência da execução da função e reduz a duração da execução.
- Reduz a probabilidade de partidas a frio.
- Reduz o número total de instâncias e economiza recursos.

Cenários de aplicação

Se as solicitações de função levarem muito tempo para aguardar a resposta dos serviços downstream, você poderá configurar a multi-concorrência de instância única para que as solicitações possam ser processadas simultaneamente.

Comparação

Se uma função demorar 5s para ser executada a cada vez e você definir o número de solicitações que podem ser processadas simultaneamente por uma instância como 1, três solicitações precisam ser processadas em três instâncias, respectivamente. Portanto, a duração total da execução é de 15s.

Quando você definir **Max. Requests per Instance** a **5**, se três solicitações forem enviadas, elas serão processadas simultaneamente por uma instância. O tempo total de execução é de 5s.

NOTA

Quando **Max. Requests per Instance** por Instância for maior que **1**, a concorrência de recursos e a segurança de simultaneidade devem ser consideradas.

Solicitações excessivas precisam esperar um pouco antes de serem processadas. Por exemplo, se o número de solicitações simultâneas de uma única instância for definido como 10, o sistema não retornará um erro imediatamente quando o número de solicitações exceder a capacidade de processamento. Os pedidos excessivos serão processados após um período de tempo.

Tabela 3-9 Comparação

Item de comparação	Concorrência única de instância única	Multi-concorrência de instância única
Impressão de registração	-	Para imprimir registrações, o Runtime do Node.js usa a função **console.info()**, o Runtime do Python usa a função **print()** e o Java Runtime usa a função **System.out.println()**. Nesse modo, os ID de solicitação atuais são incluídos no conteúdo do registração. No entanto, quando várias solicitações são processadas simultaneamente por uma instância, os ID de solicitação estão incorretas se você continuar a usar as funções anteriores para imprimir registrações. Para resolver esse problema, use a função **context.getLogger().registração()**. Esse método retém os ID da solicitação separadamente.
Variáveis compartilhadas	Não envolvido.	Modificar variáveis compartilhadas causará erros. A proteção de exclusão mútua é necessária quando você modifica variáveis non-thread-safe durante a gravação da função.
Métricas de monitoramento	Realizar monitoramento com base na situação real.	Sob a mesma carga, o número de instâncias de função diminui significativamente.
Erro de controle de fluxo	Não envolvido.	Quando há muitas solicitações, o código de erro no corpo é **FSS.0429**, o status no cabeçalho da resposta é **429**, e a mensagem de erro é **Your request has been controlled by overload sdk, please retry later**.

Configurando a Multicorrência de Instância Única

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Escolha **Configuration** > **Concurrency**.

   Defina os parâmetros referindo-se a **Tabela 3-10** e clique em **Save**.

Figura 3-15 Configuração de simultaneidade

Tabela 3-10 Descrição

Parâmetro	Descrição
Max. Requests per Instance	Número de solicitações simultâneas suportadas por uma única instância. Intervalo do valor: 1 – 100. Para mais detalhes, veja **Configuração da multi-corrência de instância única**.
Max. Instances per Function	Número máximo de instâncias nas quais uma função pode ser executada. **–1** (predefinição): A função pode ser executada em qualquer número de instâncias. **0**: A função está desativada. **NOTA** As solicitações que excederem a capacidade de processamento das instâncias serão descartadas. Erros causados por solicitações excessivas não serão exibidos nos registrações de função. Você pode obter detalhes do erro consultando **Configuração da notificação de execução assíncrona**.

Restrições de configuração

- Para funções Java, a multi-concorrência de instância única tem as seguintes restrições:
  - 128 MB de memória: até 25 solicitações simultâneas
  - 256 MB de memória: até 50 solicitações simultâneas
  - 512 MB ou mais de memória: até 100 solicitações simultâneas
- Para funções Python, os threads em uma instância são vinculados a um núcleo devido ao bloqueio GIL (Global Interpreter Lock) do Python. Como resultado, as solicitações simultâneas só podem ser processadas usando o núcleo único, não vários núcleos. O desempenho de processamento de função não pode ser melhorado mesmo que especificações de recursos maiores estejam configuradas.
- Para funções Node.js, o processamento de thread único do mecanismo V8 resulta no processamento de solicitações simultâneas usando apenas um único núcleo, não vários núcleos. O desempenho de processamento de função não pode ser melhorado mesmo que especificações de recursos maiores estejam configuradas.

3.10 Gerenciamento das versões

Visão geral

O FunctionGraph permite publicar uma ou mais versões nos processos de desenvolvimento, teste e produção para gerenciar seu código de função. As variáveis de código e ambiente de cada versão são salvas como um instantâneo. Depois que o código da função for publicado, você poderá modificar as configurações conforme necessário.

Depois que uma função é criada, a versão padrão é a mais recente. Cada função tem a versão mais recente. Depois que o código da função for publicado, você poderá modificar a configuração da versão conforme necessário.

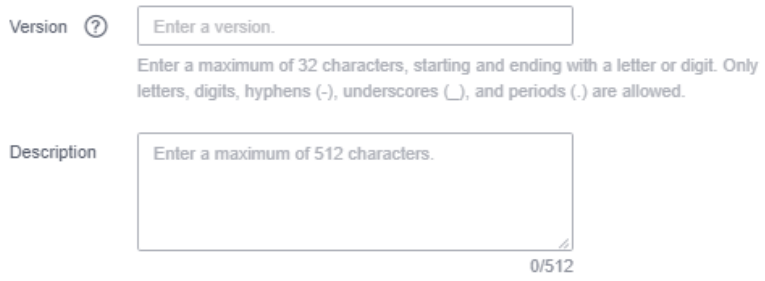
NOTA

Uma versão é um instantâneo de uma função e corresponde a uma tag no código. Cada versão contém a configuração e o código da função. Por padrão, nenhum acionador está vinculado a uma nova versão. Depois que uma versão é publicada, a configuração (como variáveis de ambiente) e o código da versão não podem ser atualizados, para garantir estabilidade e rastreabilidade.

Publicando uma versão

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Na página de guia **Version**, clique em **Publish new version**.

   Figura 3-16 Parâmetros para publicar uma nova versão

   - **Version**: Insira o número da versão. Se nenhum número de versão for especificado, o sistema gerará automaticamente um número de versão com base na data atual, por exemplo, **v20220510-190658**.
   - **Description**: Insira uma descrição para a versão. Este parâmetro é opcional.
4. Clique em **OK**. O sistema publica automaticamente uma versão. Em seguida, você será redirecionado para a nova versão.

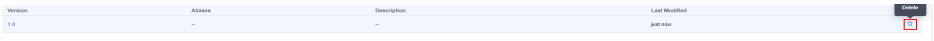
NOTA

- Você pode publicar até 10 versões para uma função.
- Para uma função cuja versão mais recente foi configurada com instâncias reservadas, a configuração da função pode ser modificada. Por padrão, as versões não mais recentes não têm instâncias reservadas.
- Nenhum disco é anexado a uma nova versão criada com base na mais recente. Variáveis de ambiente não podem ser definidas se nenhum gatilho tiver sido vinculado à versão.

Excluindo uma versão

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Na página de guia **Version** da versão mais recente, selecione a versão a ser excluída.

Figura 3-17 Excluindo uma versão



NOTA

- A versão mais recente de uma função não pode ser excluída.
- Se uma versão de função associada a aliases for excluída, os aliases também serão excluídos.

4. Clique em **OK** para excluir a versão.

ATENÇÃO

A exclusão de uma versão excluirá permanentemente o código, a configuração, o alias e o mapeamento da origem do evento associados, mas não excluirá os logs. Versões excluídas não podem ser recuperadas. Tenha cuidado ao realizar esta operação.

3.11 Gerenciamento dos aliases

Visão geral

Um alias representa uma versão de função específica e pode ser usado para chamar essa versão. Para reverter para uma versão anterior, use o alias correspondente para representar a versão em vez de modificar o código da função.

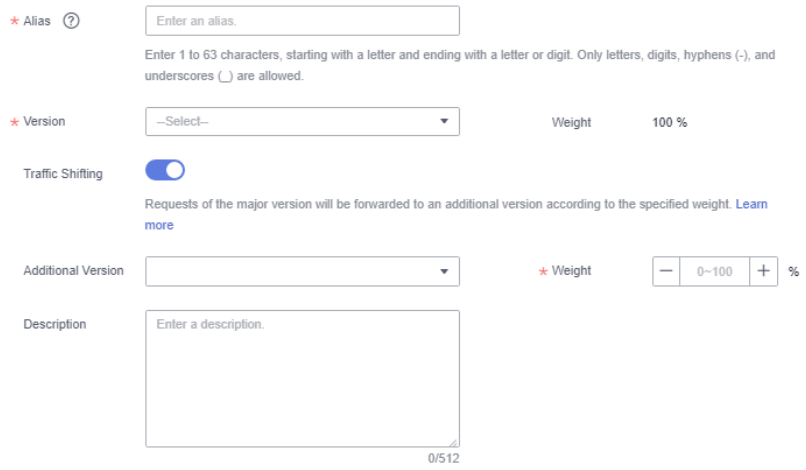
Cada alias de função pode ser vinculado a uma versão principal e a uma versão adicional para deslocamento de tráfego.

Criação de um alias

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.

3. Na página de guia **Aliases**, clique em **Create Alias**.

Figura 3-18 Criação de um alias



- **Alias**: Insira um alias.
- **Version**: Selecione uma versão a ser associada ao alias.
- **Traffic Shifting**: Escolha se deseja ativar o deslocamento de tráfego. Se essa função estiver ativada, você poderá distribuir uma porcentagem específica de tráfego para a versão adicional.
- **Additional Version**: Selecione uma versão adicional a ser associada. A versão mais recente não pode ser usada como uma versão adicional.
- **Weight**: Insira um número inteiro de 0 a 100.
- **Description**: Insira uma descrição para o alias.

4. Clique em **OK**.

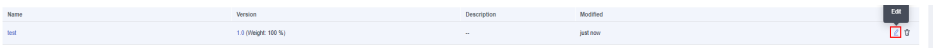
NOTA

Você pode criar até 10 aliases para uma função.

Modificação de um alias

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Na página de guia **Aliases** da versão mais recente, selecione o alias a ser modificado.

Figura 3-19 Modificação de um alias



4. Modifique as informações de alias e clique em **OK**.

Exclusão de um alias

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Na página de guia **Aliases** da versão mais recente, selecione o alias a ser excluído.

Figura 3-20 Exclusão de um alias

4. Clique em **OK** para excluir a versão.

3.12 Configuração da memória dinâmica

NOTA

Este recurso é suportado apenas pelo FunctionGraph v2.

Visão geral

Ao executar uma função, especifique dinamicamente os recursos com base no tamanho dos dados para reduzir custos.

Os recursos de computação necessários para transcodificação variam acentuadamente dependendo do tamanho dos arquivos de vídeo, dos formatos de codificação e das resoluções. Para garantir um desempenho ideal, uma especificação de grande recurso é geralmente configurada. Nesse caso, muitos recursos serão desperdiçados em um cenário de baixa resolução (por exemplo, vídeo curto). Para resolver o problema, o FunctionGraph oferece suporte à alocação dinâmica de memória durante a execução da função. Ele minimiza o consumo de recursos e alcança um controle de recursos refinado a custos mais baixos.

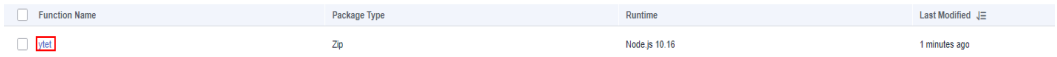
Pré-requisitos

Você criou uma função de acordo com **Criação de uma função a partir do zero**.

Procedimento

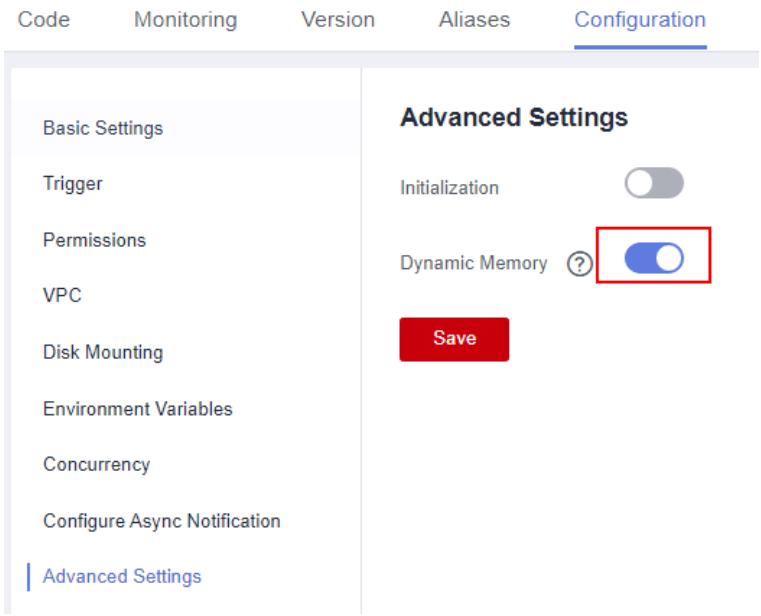
Passo 1 Faça login no console do FunctionGraph, escolha **Functions** > **Function List** no painel de navegação e clique no nome da função criada.

Figura 3-21 Selecção de uma função criada



Passo 2 Na página de detalhes da função, escolha **Configuration** > **Advanced Settings** avançadas e ative **Dynamic Memory**.

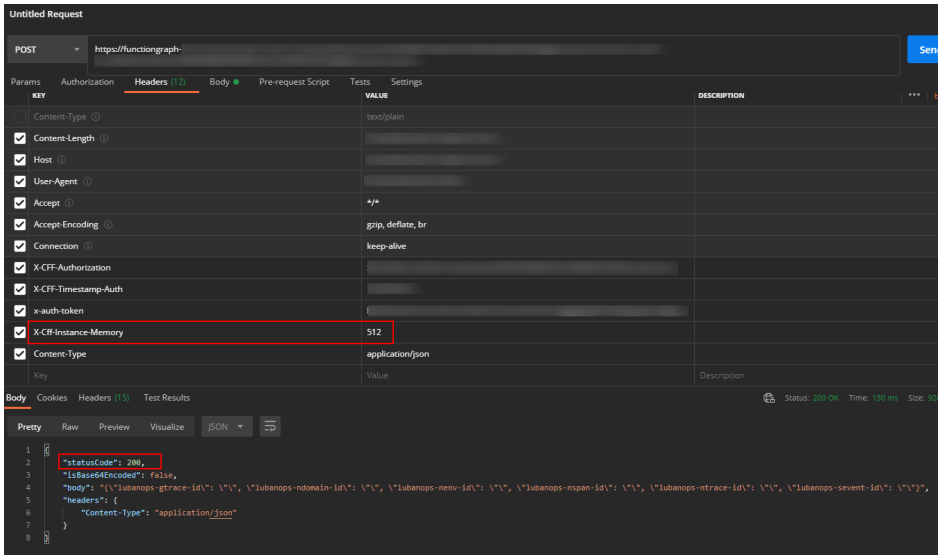
Figura 3-22 Configuração da memória dinâmica



Passo 3 Chame a API de execução de função síncrona ou execução de função assíncrona, adicione **X-Cff-Instance-Memory** ao cabeçalho da solicitação e defina o valor para **128**, **256**, **512**, **768**, **1024**, **1280**, **1536**, **1792**, **2048**, **2560**, **3072**, **3584**, ou **4096**

A seguir, descrevemos como chamar uma API usando o Postman. Adicione **X-Cff-Instance-Memory** aos **Headers** e defina o valor para **512**. Se a API for chamada com sucesso, o código de erro 200 será retornado.

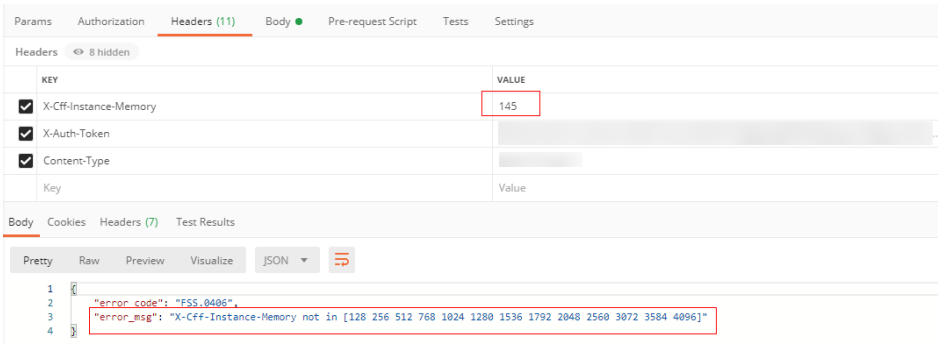
Figura 3-23 Adicionamento de um cabeçalho de requisição e chamando a função



NOTA

- Se **Dynamic Memory** não estiver ativada, o tamanho da memória definido quando a função é criada será usado por padrão.
- Se **Dynamic Memory** estiver ativada, mas o valor da memória não tiver sido definido, o tamanho da memória definido quando a função for criada será usado por padrão. Se a API for chamada com sucesso, o código de erro 200 será retornado.
- Se **Dynamic Memory** estiver ativada, mas o valor da memória não for **128**, **256**, **512**, **768**, **1024**, **1280**, **1536**, **1792**, **2048**, **2560**, **3072**, **3584**, ou **4096**, o código de erro FSS.0406 será retornado quando a API for chamada. Você só precisa redefinir o valor da memória.

Figura 3-24 Falha de invocação



----Fim

4 Criação de gatilhos

4.1 Gerenciamento de gatilhos

Ativação ou desativação de um gatilho

Você pode ativar ou desativar os gatilhos conforme necessário. **Note that SMN, OBS, and APIG triggers cannot be disabled and can only be deleted.**

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique no nome da função desejada.

Passo 3 Escolha **Configuration** > **Trigger**. Na página exibida, localize a linha que contém o gatilho de destino e clique em **Disable** ou **Enable**.

----Fim

Deletando um Trigger

Você pode excluir gatilhos que não serão mais usados.

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique no nome da função desejada.

Passo 3 Escolha **Configuration** > **Trigger**. Na página exibida, localize a linha que contém o gatilho de destino e clique em **Delete**.

----Fim

4.2 Uso de um gatilho de Timer

This section describes how to create a timer trigger to invoke your function based on a fixed rate or cron expression.

Para obter detalhes sobre a origem do evento do temporizador, consulte **Origens de eventos suportadas**.

Pré-requisitos

Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.

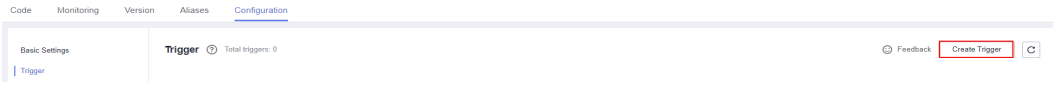
Criação de um gatilho cronômetro

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-1 Criação de um gatilho



Passo 4 Defina os seguintes parâmetros:

- **Trigger Type**: Selecione **Timer**.
- **Timer Name**: Insira um nome de temporizador, por exemplo, **Timer**.
- **Rule**: Defina uma taxa fixa ou uma expressão cron.
  - **Fixed rate**: A função é acionada a uma taxa fixa de minutos, horas ou dias. Você pode definir uma taxa fixa de 1 a 60 minutos, de 1 a 24 horas ou de 1 a 30 dias.
  - **Cron expression**: A função é acionada com base em uma regra complexa. Por exemplo, você pode definir uma função para ser executada às 08:30:00 de segunda a sexta-feira. Para obter mais informações, consulte **Apêndice: expressões Cron para um gatilho de Timer de função**.
- **Enable Trigger**: Escolha se deseja ativar o disparador do temporizador.
- **Additional Information**: As informações adicionais que você configurar serão colocadas no campo **user_event** da origem do evento timer. Para obter detalhes, consulte **Origens de eventos suportadas**.

Passo 5 Clique em **OK**.

----Fim

Visualização do resultado da execução

Depois que o gatilho cronômetro é criado, a função é executada a cada 1 minuto. Para exibir as registrações de execução da função, execute as seguintes etapas:

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique em uma função para ir para a página de detalhes da função.

Passo 3 Escolha **Monitoring** > **Logs** para consultar registrações de execução de funções.

----Fim

4.3 Uso de um gatilho de APIG (dedicado)

This section describes how to create an APIG trigger and call an API to trigger a function.

Para obter detalhes sobre a origem do evento de APIG, consulte **Origens de eventos suportadas**.

Pré-requisitos

Você criou um grupo das API, por exemplo, **APIGroup_test**. Para obter detalhes, consulte **Criação de um grupo das API**.

Criação de um gatilho de APIG

Passo 1 Efetue log-in no console do **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Na página **Function List**, clique em **Create Function** no canto superior direito.

Passo 3 Defina os seguintes parâmetros:

- **Function Name**: Insira um nome de função, por exemplo, **apig**.
- **Agency**: Selecione **Use no agency**.
- **Enterprise Project**: Selecione **default**.
- **Runtime**: Selecione **Python 2.7**.

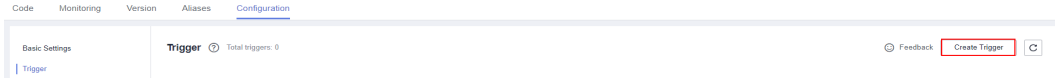
Passo 4 Clique em **Create Function**.

Passo 5 Na página de guia **Code**, copie o código a seguir para a janela de código e clique em **Deploy**.

```
# -*- coding:utf-8 -*-
import json
def handler (event, context):
    body = "<html><title>Functiongraph Demo</title><body><p>Hello, FunctionGraph!
</p></body></html>"
    print(body)
    return {
        "statusCode":200,
        "body":body,
        "headers": {
            "Content-Type": "text/html",
        },
        "isBase64Encoded": False
    }
```

Passo 6 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-2 Criação de um gatilho



Passo 7 Configure as informações do gatilho.

Tabela 4-1 Informações do gatilho

Parâmetro	Descrição
Tipo de gatilho	Selecione **API Gateway (Dedicated Gateway)**.
Instância	Selecione uma instância. Se nenhuma instância estiver disponível, clique em **Create Instance**.
Nome da API	Insira um nome de API, por exemplo, **API_apig**.
Grupo da API	Um grupo das API é uma coleção das API. Você pode gerenciar as API por grupo das API. Selecione **APIGroup_test**.
Ambiente	Uma API pode ser chamada em diferentes ambientes, como ambientes de produção, teste e desenvolvimento. O API Gateway oferece suporte ao gerenciamento de ambiente, o que permite definir diferentes caminhos de solicitação para uma API em diferentes ambientes. Para garantir que a API possa ser chamada, selecione **RELEASE**.
Autenticação de segurança	Existem três modos de autenticação: ● **App**: Autenticação AppKey e AppSecret. Este modo é de alta segurança e é recomendado. Para obter detalhes, consulte **Autenticação de aplicativos**. ● **IAM**: Autenticação do IAM. Esse modo concede permissões de acesso apenas a usuários do IAM e é de segurança média. Para obter detalhes, consulte **Autenticação do IAM**. ● **None**: Sem autenticação. Este modo concede permissões de acesso a todos os usuários. Selecione **None**.
Protocolo	Existem dois tipos de protocolos: ● HTTP ● HTTPS Selecione **HTTPS**.
Tempo limite (ms)	Digite **5000**.

Passo 8 Clique em **OK**.

NOTA

1. URL: endereço do gatilho de APIG.
2. Depois que o gatilho de APIG é criado, uma API chamada **API_apig** é gerada no console do API Gateway. Você pode clicar no nome da API na lista de gatilhos para acessar o console do API Gateway.

----Fim

Invocando a função

Passo 1 Digite o URL do gatilho de APIG na barra de endereços de um navegador e pressione **Enter**.

Passo 2 Após a execução da função, verifique o resultado da execução, conforme mostrado na **Figura 4-3**.

Figura 4-3 Resultado retornado

NOTA

1. A entrada para a invocação do API Gateway vem de um modelo de evento fornecido pela função. Para mais detalhes, veja **Tabela 5-2**.
2. A resposta da função para a invocação do API Gateway é encapsulada e deve conter **body(String)**, **statusCode(int)**, **headers(Map)**, and **isBase64Encoded(boolean)**.

----Fim

Visualização do resultado da execução

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique em uma função para ir para a página de detalhes da função.

Passo 3 Escolha **Monitoring** > **Logs** para consultar logs de execução de funções.

----Fim

4.4 Uso de um gatilho de OBS

This section describes how to create an OBS trigger and upload an image package to a specified OBS bucket to trigger a function.

Para obter detalhes sobre a origem de evento de OBS, consulte **Fontes de evento suportadas**.

Pré-requisitos

Antes de criar um gatilho, certifique-se de ter preparado o seguinte:

- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.
- Você criou um bucket de OBS, por exemplo, **obs_cff**. Para obter detalhes, consulte **Criação de um bucket**.

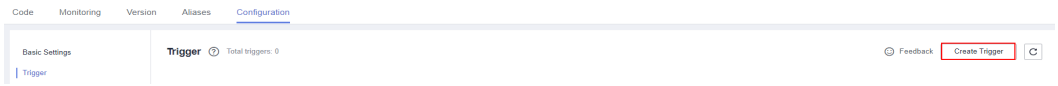
Criação de um gatilho de OBS

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-4 Criação de um gatilho



Passo 4 Defina os seguintes parâmetros:

- **gatilho Type**: Selecione **Object Storage Service (OBS)**.
- **Bucket Name**: Especifique o bucket de OBS a ser usado como uma origem de evento, por exemplo, **obs-cff**.
- **Events**: Selecione os eventos que acionarão a função. Neste exemplo, selecione **Put**, **Post**, e **Delete**. Quando os arquivos no bucket **obs_cff** são atualizados, carregados ou excluídos, a função é acionada.
- **Event Notification Name**: Especifique o nome da notificação de evento a ser enviada pelo SMN quando ocorrer um evento.
- **Prefix**: Insira uma palavra-chave para limitar as notificações àquelas sobre objetos cujos nomes começam com os caracteres correspondentes. Este limite pode ser usado para filtrar os nomes de objetos de OBS.
- **Suffix**: Insira uma palavra-chave para limitar as notificações àquelas sobre objetos cujos nomes terminam com os caracteres correspondentes. Este limite pode ser usado para filtrar os nomes de objetos de OBS.

Passo 5 Clique em **OK**.

----Fim

Disparando uma função

No console de OBS, carregue um pacote ZIP de imagens no bucket **obs-cff**. Para obter detalhes, consulte **Carregando um arquivo**.

NOTA

Depois que o pacote ZIP é carregado no bucket **obs-cff**, a função **HelloWorld** é acionada.

Visualização do resultado da execução

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique em uma função para ir para a página de detalhes da função.

Passo 3 Escolha **Monitoring** > **Logs** para consultar logs de execução de funções.

----Fim

4.5 Uso de um gatilho de Kafka

Esta seção descreve como criar um gatilho de Kafka e configurar um evento de Kafka para acionar uma função.

Depois que um gatilho de Kafka é usado, o FunctionGraph pesquisa periodicamente novas mensagens em um tópico específico em uma instância do Kafka e passa as mensagens como parâmetros de entrada para chamar funções. Para obter detalhes sobre a origem de eventos do DMS for Kafka, consulte **Fontes de eventos suportadas**.

Pré-requisitos

Antes de criar um gatilho, certifique-se de ter preparado o seguinte:

- Você criou uma função. Para mais detalhes, consulte **Criação de uma função a partir do zero**.
- Você ativou o acesso VPC para a função. Para mais detalhes, consulte **Configuração da rede**.
- Você criou uma instância de Kafka. Para obter detalhes, consulte **Compra de uma instância**.
- Você criou um tópico sob uma instância de Kafka. Para obter detalhes, consulte **Criação de um tópico**.

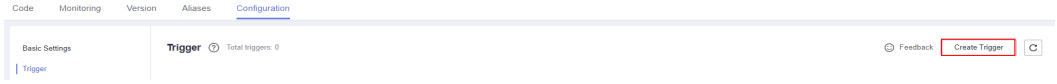
Criação de um gatilho de Kafka

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-5 Criação de um gatilho



Passo 4 Configure os parâmetros a seguir:

- **Trigger Type**: selecione **Distributed Message Service for Kafka (Kafka)**.
- **Instance** selecione uma instância premium de Kafka.
- **Topic**: Selecione um tópico da instância premium de Kafka.
- **Batch Size**: Defina o número de mensagens a serem recuperadas de um tópico a cada vez.
- **Username**: Informe o nome de usuário da instância se o SSL tiver sido ativado para ela.
- **Password**: Informe a senha da instância se o SSL tiver sido ativado para ela.

Passo 5 Clique em **OK**.

NOTA

Depois que o acesso à VPC for ativado, você precisará configurar as permissões de sub-rede correspondentes para o grupo de segurança de Kafka. Para obter detalhes sobre como configurar o acesso à VPC, consulte **Configuração da rede**.

----Fim

Configuração de um evento de Kafka para acionar a função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão.

Passo 4 Na página de guia **Code**, clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 5 Defina os parâmetros descritos em **Tabela 4-2** e clique em **Save**.

Tabela 4-2 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Event Template	Selecione **kafka-event-template**.
Event Name	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **kafka-123test**.
Event data	O sistema carrega automaticamente o modelo de evento Kafka integrado, que é usado neste exemplo sem modificações.

NOTA

O modelo de evento é o seguinte:

```
{
    "event_version": "v1.0",
    "event_time": 1576737962,
    "trigger_type": "KAFKA",
    "region": "xx-xxxx-1",
    "records": [{
        "messages": [
            "kafka message1",
            "kafka message2",
            "kafka message3",
            "kafka message4",
            "kafka message5"
        ],
    "instance_id": "81335d56-b9fe-4679-ba95-7030949cc76b",
    "topic_id": "topic-test"
    }]
}
```

Passo 6 Clique em **Test**. O resultado do teste de função é exibido.

----Fim

4.6 Uso de um gatilho de DIS

This section describes how to create a DIS trigger for a function, and configure a DIS event by using the built-in event template to trigger the function.

Para obter detalhes sobre a origem do evento de DIS, consulte **Origens de eventos suportadas**.

Pré-requisitos

Antes de criar um gatilho, certifique-se de ter preparado o seguinte:

- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.
- Você criou um fluxo DIS, por exemplo, **dis-function**.Para obter detalhes, consulte **Criação de um fluxo de DIS**.

Configuração de uma agência

Antes de criar um gatilho de DIS, defina uma agência para delegar FunctionGraph para acessar o DIS. Para obter detalhes sobre como criar uma agência, consulte **Configuração das permissões de agência**.

Como você não especificou uma agência ao criar a função **HelloWorld**, especifique uma primeiro.

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **Permissions** e altere a agência para **serverless-trust** criada em **Configuração das permissões de agência**.

Passo 4 Clique em **Save**.

----Fim

Criação de um gatilho de DIS

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **Trigger** e clique em **Create Trigger**.

Passo 4 Defina os seguintes parâmetros:

- **Trigger Type**: Selecione **Data Ingestion Service (DIS)**.
- **Stream Name**: Selecione um fluxo DIS, por exemplo, **dis-function**.
- **Max. Fetch Bytes**: Volume máximo de dados que podem ser buscados em cada solicitação. Somente os registros menores que esse valor serão buscados. O valor varia de 1 KB a 4 MB.

- **Starting Position**: Especifique uma posição no fluxo especificado a partir da qual iniciar a leitura de dados.
  - **TRIM_HORIZON**: Os dados são lidos a partir dos registos válidos mais antigos armazenados na partição.
  - **latest**: Os dados são lidos logo após o registro mais recente na partição. Essa configuração garante que você sempre leia os dados mais recentes.
- **Pull Period**: Defina um período para extrair dados do fluxo.
- **Serial Data Processing**: Se essa opção for selecionada, o FunctionGraph extrairá dados do fluxo somente depois que os dados anteriores forem processados. Se essa opção não for selecionada, o FunctionGraph extrairá dados do fluxo desde que o período de baixa termine.

Passo 5 Clique em **OK**.

----Fim

Configuração de um evento de DIS para acionar a função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão.

Passo 4 Na página de guia **Code**, clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 5 Defina os parâmetros descritos em **Tabela 4-3** e clique em **Tabela 4-3**.

Tabela 4-3 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Event Template	Selecione **dis-event-template**.
Event Name	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **dis-123test**.
Event data	O sistema carrega automaticamente o modelo de evento DIS embutido, que é usado neste exemplo sem modificações. O código neste modelo é o seguinte:

NOTA

The event template is as follows:

```
{
    "ShardID": "shardId-0000000000",
    "Message": {
        "next_partition_cursor":
"eyJnZXRJdGVyYXRvclBhcmFtIjp7InN0cmVhbS1uYW1lIjoiZGlzLXN3dGVzdCIsInBhc
nRpdGlvbi1pZCI6InNoYXJkSWQtMDAwMDAwMDAwMCIsImN1cnNvci10eXBlIjoiVFJJTV9
IT1JJWk9OIiwic3RhcnRpbmctc2VxdWVuY2UtbnVtYmVyIjoiNCJ9LCJnZW5lcmF0ZVRpb
WVzdGFtcCI6MTUwOTYwNjM5MjE5MX0",
        "records": [
            {
                "partition_key": "shardId_0000000000",
                "data": "d2VsY29tZQ==",
                "sequence_number": "0"
            },
            {
                "partition_key": "shardId_0000000000",
                "data": "dXNpbmc=",
                "sequence_number": "1"
            },
            {
                "partition_key": "shardId_0000000000",
                "data": "RnVuY3Rpb25TdGFnZQ==",
                "sequence_number": "2"
            },
            {
                "partition_key": "shardId_0000000000",
                "data": "c2VydmljZQ==",
                "sequence_number": "3"
            }
        ],
        "millis_behind_latest": ""
    },
    "Tag": "latest",
    "StreamName": "dis-swtest"
}
```

Passo 6 Clique em **Test**. O resultado do teste de função é exibido.

----Fim

4.7 Uso de um gatilho de SMN

Esta seção descreve como criar um gatilho de SMN e publicar uma mensagem para acionar uma função.

Para obter detalhes sobre a origem do evento de SMN, consulte **Fontes de eventos suportadas**.

Pré-requisitos

- Você criou um tópico de SMN, por exemplo, **smn-test**. Para obter detalhes, consulte Criação de um tópico.
- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.

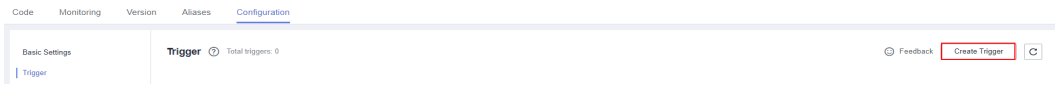
Criação de um gatilho de SMN

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-6 Criação de um gatilho



Passo 4 Configure os parâmetros a seguir:

- **Trigger Type**: Selecione **SMN**.
- **Topic Name**: Selecione um tópico, por exemplo, **smn-test**.

Passo 5 Clique em **OK**.

NOTA

Depois que o gatilho de SMN é criado, uma assinatura está gerada para o tópico correspondente no console de SMN.

----Fim

Pública de uma mensagem para acionar a função

No console de SMN, publique uma mensagem no tópico **smn-test**. Para obter detalhes, consulte Pública de uma mensagem de texto.

Tabela 4-4 descreve os parâmetros necessários para publicar uma mensagem.

Tabela 4-4 Parâmetros necessários para publicar uma mensagem

Parâmetro	Descrição
Subject	Digite **SMN-Test**.
Message Format	Selecione **Text**.
Message	Digite **{"message":"hello"}**.

NOTA

Depois que uma mensagem é publicada, a função é acionada automaticamente. Para obter detalhes sobre eventos de exemplo, consulte **Fontes de eventos suportadas**.

Visualização do resultado da execução

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique em uma função para ir para a página de detalhes da função.

Passo 3 Escolha **Monitoring** > **Logs** para consultar registrações de execução de funções.

----Fim

4.8 Uso de um gatilho de LTS

This section describes how to create an LTS trigger for a function, and invoke the function when log events occur.

Para obter detalhes sobre a origem do evento do timer, consulte **Origens de eventos suportadas**.

Pré-requisitos

- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.
- Você criou uma agência com a permissão de **LTS FullAccess**. Para obter detalhes sobre como criar uma agência, consulte **Configuração das permissões de agência**.
- Você criou um grupo de registrações, por exemplo, LogGroup1. Para obter detalhes, consulte **Criação de um grupo de registrações**.
- Você criou um de streamlog, por exemplo, LogTopic1. cPara obter detalhes, consulte **Criação de um fluxo de registração**.

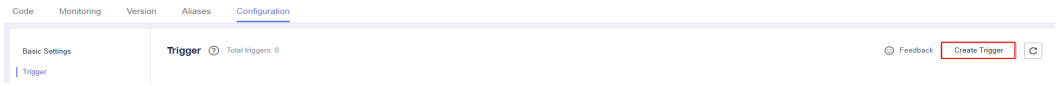
Criação de um gatilho LTS

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-7 Criação de um gatilho



Passo 4 Configure os parâmetros a seguir:

- **Trigger Type**: Selecione **Log Tank Service (LTS)**.
- **Log Group**: Selecione um grupo de registrações, por exemplo, **LogGroup1**.
- **Log Topic**: Selecione um tópico de registração, por exemplo, **LogTopic1**.

Passo 5 Clique em **OK**.

----Fim

Configuração de um evento de LTS para disparar a função

NOTA

Quando o tamanho de uma mensagem de evento de LTS exceder 75 KB, ela será dividida em várias mensagens por 75 KB para acionar a função.

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão.

Passo 4 Na página de guia **Code**, clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 5 Defina os parâmetros descritos em **Tabela 4-5** e clique em **Save**.

Tabela 4-5 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Event Template	Selecione **lts-event-template**.
Event Name	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **lts-123test**.
Event data	O sistema carrega automaticamente o modelo de evento LTS integrado, que é usado neste exemplo sem modificações.

NOTA

O modelo de evento é o seguinte:

```
{
    "lts": {
        "data":
"ewogICAgICAgICJsb2dzIjpbewogICAgICAgICAgICAgIm1lc3NhZ2UiOiIyMDE4L
TA2LTI2LzE4OjQwOjUzIFtJTkZdIFtjb25maWcuZ286NzJdIFN1Y2Nlc3NmdWxseSBsb2F
kZWQgZ2VuZXJhbCBjb25maWd1cmF0aW9uIGZpbGVcXHJcXG4iLAogICAgICAgICAgICAgI
CAgInRpbWUiOjE1MzAwMDk2NTMwNTksCiAgICAgICAgICAgICAgICAiaG9zdF9uYW1lIjo
iZWNzLXRlc3RhZ2VudC5ub3ZhbG9jYWwiLAogICAgICAgICAgICAgICAgImlwIjoiMTkyL
jE2OC4xLjk4IiwKICAgICAgICAgICAgICAgICJwYXRoIjoidXNyL2xvY2FsL3RlbGVzY29
wZS9sb2cvY29tbW9uLmxvZyIsCiAgICAgICAgICAgICAgICAibG9nX3VpZCI6IjY2M2Q2O
TMwLTc5MmQtMTFlOC04YjA4LTI4NmVkNDg4Y2U3MCIsCiAgICAgICAgICAgICAgICAibGl
uZV9ubyI6NjE1CiAgICAgICAgICAgICAgICAgIH0sCiAgICAgICAgICAgICAgICAgICJtZXNzYWdlIjoiMjAxOC0wNi0yNi8xODo0MDo1MyBbV1JOXSBbY29uZmlnLmdvOjg
yXSBUaGUgcHJvamVjdElkIG9yIGluc3RhbmNlSWQgb2YgY29uZmlnLmpzb24gaXMgbm90I
GNvbnNpc3RlbnQgd2l0aCBtZXRhZGF0YSwgdXNlIG1ldGFkYXRhLlxcbiIsCiAgICAgICA
gICAgICAgICAidGltZSI6MTUzMDAwOTY1MzA1OSwKICAgICAgICAgICAgICAgICJob3N0X
25hbWUiOiJlY3MtdGVzdGFnZW50Lm5vdmFsb2NhbCIsCiAgICAgICAgICAgICAgICAiaXA
iOiIxOTIuMTY4LjEuOTgiLAogICAgICAgICAgICAgICAgInBhdGgiOiIvdXNyL2xvY2FsL
3RlbGVzY29wZS9sb2cvY29tbW9uLmxvZyIsCiAgICAgICAgICAgICAgICAibG9nX3VpZCI
6IjY2M2Q2OTMwLTc5MmQtMTFlOC04YjA5LTI4NmVkNDg4Y2U3MCIsCiAgICAgICAgICAgI
CAgICAibGluZV9ubyI6NjE2CiAgICAgICAgICAgICAgICAgIH0sCiAgICAgICAgICAgICAgI
CAgICAgICJtZXNzYWdlIjoiIEluIGNvbmYuanNvbiwgcHJvamVjdElkIGlzIFtdLCBpbnN0YW5jZUlkIGlzIFtdLiBNZXRhRGF0YSBpcyB7NDU0MzI5M2EtNWIyYy00NGM0LWI
3YTAtZGUyMThmN2YyZmE2IDYyODBlMTcwYmQ5MzRmNjBhNGQ4NTFjZjVjYTA1MTI5ICB9X
FxyXFxuIiwKICAgICAgICAgICAgICAgICJ0aW1lIjoxNTMwMDA5NjUzMDU5LAogICAgICA
gICAgICAgICAgImhvc3RfbmFtZSI6ImVjcy10ZXN0YWdlbnQubm92YWxvY2FsIiwKICAgI
CAgICAgICAgICAgICJpcCI6IjE5Mi4xNjguMS45OCIsCiAgICAgICAgICAgICAgICAicGF
0aCI6Ii91c3IvbG9jYWwvdGVsZXNjb3BlL2xvZy9jb21tb24ubG9nIiwKICAgICAgICAgI
CAgICAgICJsb2dfdWlkIjoiNjYzZDY5MzAtNzkyZC0xMWU4LThiMGEtMjg2ZWQ0ODhjZTc
wIiwKICAgICAgICAgICAgICAgICJsaW5lX25vIjo2MTcKICAgICAgICAgICAgICAgICAgfQogICAgI
CAgICAgICBdLAogICAgICAgICJvd25lciI6ICI2MjgwZTE3MGJkOTM0ZjYwYTRkODUxY2Y
1Y2EwNTEyOSIsCiAgICAgICAgICAgImxvZ19ncm91cF9pZCI6ICI5N2E5ZDI4NC00NDQ4LTExZ
TgtOGZhNC0yODZlZDQ4OGNlNzAiLAogICAgICAgICAgICAgICJob3dfdG9waWNfaWQiOiAiMWE5Njc
1YTctNzg0ZC0xMWU4LTlmNzAtMjg2ZWQ0ODhjZTcwIgogICAgICAgICAgICAgICAgIH0="
    }
}
```

Passo 6 Clique em **Test**. O resultado do teste de função é exibido.

----Fim

4.9 Uso de um gatilho de CTS

This section describes how to create a CTS trigger for a function, and invoke the function in response to cloud resource operations recorded by CTS.

Para obter detalhes sobre a origem de eventos de CTS, consulte **Supported Event Sources**.

Pré-requisitos

Você criou uma agência no IAM. Para mais detalhes, veja **Configuração das permissões de agência**.

Criação de um gatilho de CTS

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Na página **Function List**, clique em **Create Function** no canto superior direito.

Passo 3 Configure os parâmetros a seguir:

- **Function Name**: Insira um nome de função, por exemplo, **HelloWorld**.
- **Agency**: Selecione **Use no agency**.
- **Enterprise Project**: Selecione **default**.
- **Runtime**: Selecione **Python 2.7**.

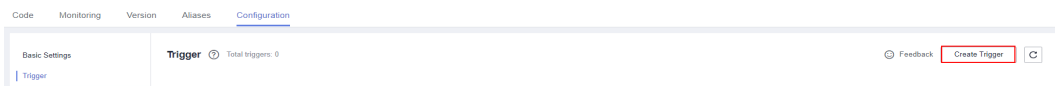
Passo 4 Clique em **Create Function**.

Passo 5 Na página de guia **Code**, copie o código a seguir para a janela de código e clique em **Deploy**.

```
# -*- coding:utf-8 -*-
'''
CTS trigger event:
{
  "cts":  {
        "time": "",
        "user": {
            "name": "userName",
            "id": "",
            "domain": {
                "name": "domainName",
                "id": ""
            }
        },
        "request": {},
        "response": {},
        "code": 204,
        "service_type": "FunctionGraph",
        "resource_type": "",
        "resource_name": "",
        "resource_id": {},
        "trace_name": "",
        "trace_type": "ConsoleAction",
        "record_time": "",
        "trace_id": "",
        "trace_status": "normal"
    }
}
'''
def handler (event, context):
    trace_name = event["cts"]["resource_name"]
    timeinfo = event["cts"]["time"]
    print(timeinfo+' '+trace_name)
```

Passo 6 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-8 Criação de um gatilho



Passo 7 Configure as informações do gatilho.

Tabela 4-6 Informações do gatilho

Parâmetro	Descrição
Trigger Type	Selecione **Cloud Trace Service (CTS)**.
Notification Name	Insira um nome de notificação, por exemplo, **Test**.
Service Type	Selecione **FunctionGraph**.
Resource Type	Tipos de recursos suportados pelo serviço selecionado, como gatilhos, instâncias e funções.
Trace Name	Operações que podem ser executadas no tipo de recurso selecionado, como criar ou excluir um gatilho.

Passo 8 Clique em **OK**.

----Fim

Configuração de um evento CTS para disparar a função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão e clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 4 Defina os parâmetros descritos em **Tabela 4-7** e clique em **Save**.

Tabela 4-7 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Event Template	Selecione **cts-event-template**.
Event Name	Insira um nome de evento, por exemplo, **cts-test**.
Event data	O sistema carrega automaticamente os dados do evento no modelo de evento de CTS. Você pode modificar os dados do evento conforme necessário.

Passo 5 Clique em **Test**. O resultado do teste de função é exibido.

----Fim

4.10 Uso de um gatilho de DDS

Esta seção descreve como criar um gatilho de DDS para uma função e como chamar a função quando uma tabela de banco de dados for alterada.

Uma função usando um gatilho de DDS será acionada toda vez que uma tabela de banco de dados for atualizada. Para obter detalhes sobre a origem do evento de DDS, consulte **Supported Event Sources**.

Pré-requisitos

Antes de criar um gatilho, certifique-se de ter preparado o seguinte:

- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.
- Você ativou o acesso VPC para a função. Para mais detalhes, veja **Configuração da rede**.
- Você criou uma instância de banco de dados de DDS.
- Você criou um banco de dados de DDS.

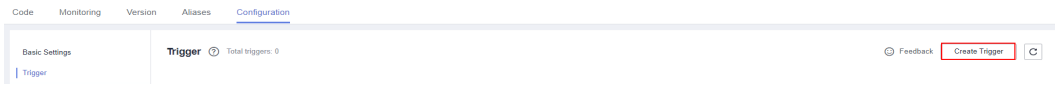
Criação de um gatilho de DDS

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-9 Criação de um gatilho



Passo 4 Configure os parâmetros a seguir:

- **Trigger Type**: Selecione **Document Database Service (DDS)**.
- **DB Instance**: Selecione uma instância de banco de dados de DDS.
- **Password**: Digite a senha do administrador da instância de banco de dados de DDS **rwuser**.
- **Database**: Insira o nome de um banco de dados. Observe que **admin**, **local**, e **config** são nomes de banco de dados reservados e não podem ser usados aqui.
- **Collection**: Insira o nome de uma coleção de banco de dados.
- **Batch Size**: Defina o número de registros a serem lidos do banco de dados por vez.

Passo 5 Clique em **OK**.

> **NOTA**
>
> Depois que o acesso à VPC for ativado, você precisará configurar as permissões de sub-rede correspondentes para o grupo de segurança de DDS. Para obter detalhes sobre como configurar o acesso à VPC, consulte **Configuração da rede**.

----Fim

Configuração de um evento de DDS para acionar a função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão e clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 4 Defina os parâmetros descritos em **Tabela 4-8** e clique em **Save**.

Tabela 4-8 Informações sobre o evento de teste

Parâmetro	Descrição
Configurar evento de teste	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Modelo de evento	Selecione **dds-event-template**.
Nome do evento	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **dds-123test**.
Dados do evento	O sistema carrega automaticamente o modelo de evento DDS integrado, que é usado neste exemplo sem modificações.

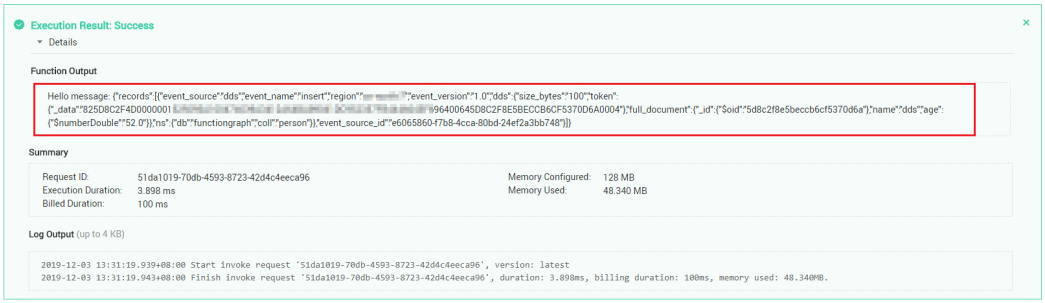
NOTA

O modelo de evento é o seguinte:

```
{
    "records": [
        {
            "event_source": "dds",
            "event_name": "insert",
            "region": "xx-xxxxx-1",
            "event_version": "1.0",
            "dds": {
                "size_bytes": "100",
                "token": {
                    "_data":
"825D8C2F4D0000001529295A100474039A3412A64BA89041DC952357FB4446645F696
400645D8C2F8E5BECCB6CF5370D6A0004"
                },
                "full_document": {
                    "_id": {
                        "$oid": "5d8c2f8e5beccb6cf5370d6a"
                    },
                    "name": "dds",
                    "age": {
                        "$numberDouble": "52.0"
                    }
                },
                "ns": {
                   "db": "functiongraph",
                   "coll": "person"
                }
            },
            "event_source_id": "e6065860-f7b8-4cca-80bd-24ef2a3bb748"
        }
    ]
}
```

Passo 5 Clique em **Test**. O resultado do teste de função é exibido, como mostrado em **Figura 4-10**.

Figura 4-10 Resultado do teste



----Fim

4.11 Uso de um gatilho de GaussDB(para Mongo)

Esta seção descreve como criar um gatilho de GaussDB(para Mongo) para uma função.

Uma função usando um gatilho de GaussDB(para Mongo) será acionada toda vez que uma tabela de banco de dados for atualizada. Para obter detalhes sobre a origem do evento de GaussDB(para Mongo), consulte **Fontes de evento suportadas**.

Pré-requisitos

Antes de criar um gatilho, certifique-se de ter preparado o seguinte:

- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.
- Você ativou o acesso VPC para a função. Para mais detalhes, veja **Configuração da rede**.
- Você criou uma instância de GaussDB(para Mongo). Para obter detalhes, consulte **Compra de uma instância de conjunto de réplicas**.

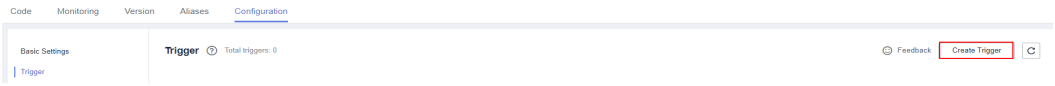
Uso de um gatilho de GaussDB(para Mongo)

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-11 Criação de um gatilho



Passo 4 Configure os parâmetros a seguir:

- **Trigger Type**: Selecione **GaussDB(for Mongo)**.
- **GaussDB(for Mongo) Instance**: Selecione uma instância de GaussDB(para Mongo).
- **Password**: Digite a senha do **rwuser** do administrador da instância de GaussDB(para Mongo).
- **Database**: Digite o nome de um banco de dados de GaussDB(para Mongo). Observe que **admin**, **local**, e **config** são nomes de banco de dados reservados e não podem ser usados aqui.
- **Collection**: Insira o nome de uma coleção de banco de dados.
- **Batch Size**: Defina o número de registros a serem lidos do banco de dados por vez.

Passo 5 Clique em **OK**.

NOTA

Depois que o acesso à VPC for ativado, você precisará configurar as permissões de sub-rede correspondentes para o grupo de segurança de GaussDB(para Mongo). Para obter detalhes sobre como configurar o acesso à VPC, consulte **Configuração da rede**.

----Fim

Configuração de um evento de GaussDB(para Mongo) para acionar a função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão.

Passo 4 Na página de guia **Code**, clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 5 Defina os parâmetros descritos em **Tabela 4-9** e clique em **Save**.

Tabela 4-9 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Event Template	Selecione **gaussmongo-event-template**.
Event Name	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **gaussmongo-123test**.
Event data	O sistema carrega automaticamente o modelo de evento de GaussDB(para Mongo), que é usado neste exemplo sem modificações.

Passo 6 Clique em **Test**. O resultado do teste de função é exibido.

----Fim

4.12 Uso de um gatilho de APIG (compartilhado)

Esta seção descreve como criar um gatilho de APIG e chamar uma API para acionar uma função.

Para obter detalhes sobre a origem do evento de APIG, consulte **Origens de eventos suportadas**.

NOTA

O API Gateway não fornece mais gateways compartilhados. Somente os clientes que se registraram antes que esse recurso fosse removido podem continuar usando-o.

Pré-requisitos

Você criou um grupo das API, por exemplo, **APIGroup_test**. Para obter detalhes, consulte **Criação de um grupo das API**.

Criação de um gatilho de APIG

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em **Create Function**.

Passo 3 Defina os seguintes parâmetros:

- **Function Name**: Insira um nome de função, por exemplo, **apig**.
- **Agency**: Selecione **Use no agency**.
- **Enterprise Project**: Selecione **default**.
- **Runtime**: Selecione **Python 2.7**.

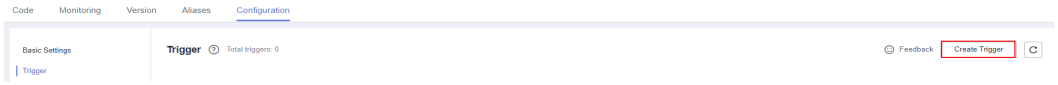
Passo 4 Clique em **Create Function**.

Passo 5 Na página de guia **Code**, copie o código a seguir para a janela de código e clique em **Deploy**.

```
# -*- coding:utf-8 -*-
import json
def handler (event, context):
    body = "<html><title>Functiongraph Demo</title><body><p>Hello, FunctionGraph!
</p></body></html>"
    print(body)
    return {
        "statusCode":200,
        "body":body,
        "headers": {
            "Content-Type": "text/html",
        },
        "isBase64Encoded": False
    }
```

Passo 6 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-12 Criação de um gatilho



Passo 7 Configure as informações do gatilho.

Tabela 4-10 Informações do gatilho

Parâmetro	Descrição
Trigger Type	Selecione **API Gateway (APIG)**.
API Name	Insira um nome de API, por exemplo, **API_apig**.
API Group	Um grupo das API é uma coleção das API. Você pode gerenciar as API por grupo das API. Selecione **APIGroup_test**.
Environment	Uma API pode ser chamada em diferentes ambientes, como ambientes de produção, teste e desenvolvimento. O API Gateway oferece suporte ao gerenciamento de ambiente, o que permite definir diferentes caminhos de solicitação para uma API em diferentes ambientes. Para garantir que a API possa ser chamada, selecione **RELEASE**.

Parâmetro	Descrição
Security Authentication	Existem três modos de autenticação: ● **App**: Autenticação AppKey e AppSecret. Este modo é de alta segurança e é recomendado. Para obter detalhes, consulte **Autenticação de aplicativos**. ● **IAM**: Autenticação do IAM. Esse modo concede permissões de acesso apenas a usuários do IAM e é de segurança média. Para obter detalhes, consulte **Autenticação do IAM**. ● **None**: Sem autenticação. Este modo concede permissões de acesso a todos os usuários. Selecione **None**.
Protocol	Existem dois tipos de protocolos: ● HTTP ● HTTPS Selecione **HTTPS**.
Timeout (ms)	Digite **5000**.

Passo 8 Clique em **OK**.

NOTA

1. O URL do gatilho APIG é **https://0ed9f61512d34982917a4f3cfe8ddd5d.apig.xxx.xxx.com/apig**.
2. Depois que o gatilho APIG é criado, uma API chamada **API_apig** é gerada no console do API Gateway. Você pode clicar no nome da API na lista de gatilhos para acessar o console do API Gateway.

----Fim

Invocação da função

Passo 1 Digite o URL do gatilho APIG na barra de endereços de um navegador e pressione **Enter**.

Passo 2 Depois que a função é executada, um resultado é retornado.

----Fim

4.13 Uso de um gatilho de APIC

Esta seção descreve como criar um gatilho de APIC e chamar uma API para acionar uma função.

Para obter detalhes sobre a origem de evento APIC, consulte **Origens de evento suportadas**.

Pré-requisitos

Você criou um grupo das API, por exemplo, **APIConnect_test**. Para obter detalhes, consulte **Criação de um grupo das API**.

Criação de um gatilho de APIC

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em **Create Function**.

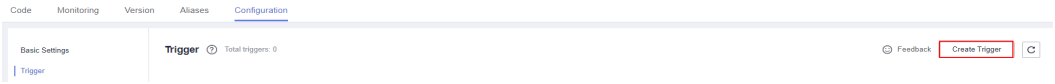
Passo 3 Configure os parâmetros a seguir:

- **Function Name**: Insira um nome de função, por exemplo, **apig**.
- **Agency**: Selecione **Use no agency**.
- **Enterprise Project**: Selecione **default**.
- Para **Runtime**, selecione **Node.js 10.16**.

Passo 4 Clique em **Create Function**.

Passo 5 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-13 Criação de um gatilho



Passo 6 Configure as informações do gatilho.

Tabela 4-11 Informações do gatilho

Parâmetro	Descrição
Trigger Type	Selecione **API Connect (APIC)**.
Instance	Selecione uma instância. Se nenhuma instância estiver disponível, clique em **Create Instance**.
API Name	Insira um nome de API, por exemplo, **API_apic**.
API Group	Um grupo das API é uma coleção dãs API. Você pode gerenciar as API por grupo das API. Exemplo: **DEFAULT**.
Environment	Uma API pode ser chamada em diferentes ambientes, como ambientes de produção, teste e desenvolvimento. O API Gateway oferece suporte ao gerenciamento de ambiente, o que permite definir diferentes caminhos de solicitação para uma API em diferentes ambientes. Para garantir que a API possa ser chamada, selecione **RELEASE**.

Parâmetro	Descrição
Security Authentication	Existem três modos de autenticação: ● **App**: Autenticação AppKey e AppSecret. Este modo é de alta segurança e é recomendado. Para obter detalhes, consulte **Autenticação de aplicativos**. ● **IAM**: Autenticação do IAM. Esse modo concede permissões de acesso apenas a usuários do IAM e é de segurança média. Para obter detalhes, consulte **Autenticação do IAM**. ● **None**: Sem autenticação. Este modo concede permissões de acesso a todos os usuários. Selecione **None**.
Protocol	Existem dois tipos de protocolos: ● HTTP ● HTTPS Selecione **HTTPS**.
Timeout (ms)	Digite **5000**.

Passo 7 Clique em **OK**.

NOTA

Depois que o gatilho de é criado, uma API chamada **API_apic** é gerada no console do API Gateway. Você pode clicar no nome da API na lista de gatilhos para acessar o console do API Gateway.

----Fim

Invocação da função

Passo 1 Efetue login no ROMA Connect, localize a instância selecionada (por exemplo, **Ac6-instance-NoDelete**), e visualize o endereço IP público.

Passo 2 Digite o endereço IP público na caixa de endereço do navegador.

Passo 3 Depois que a função é executada, um resultado é retornado.

----Fim

Visualização do resultado da execução

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função **nodejs-test**.

Passo 3 Na página de detalhes da função exibida, clique na guia **Logs** para consultar os logs de execução da função.

Passo 4 Clique em **View Context** na mesma linha de um log para exibir os detalhes do log.

----Fim

4.14 Uso de um gatilho de DMS (para RabbitMQ)

Esta seção descreve como criar um gatilho de DMS (para RabbitMQ) para uma função. Atualmente, apenas o modo de troca fanout é suportado. Depois que um gatilho de DMS é usado, o FunctionGraph pesquisa periodicamente novas mensagens em um tópico específico vinculado à troca de uma instância de RabbitMQ e passa as mensagens como parâmetros de entrada para invocar funções. Para obter detalhes sobre a origem do evento de DMS (para RabbitMQ), consulte **Fontes de eventos suportadas**.

Pré-requisitos

- Você criou uma função. Para mais detalhes, veja **Criação de uma função a partir do zero**.
- Você ativou o acesso à VPC. Para mais detalhes, veja **Configuração da rede**.
- Uma instância de RabbitMQ foi criada. Para obter detalhes, consulte **Como comprar uma instância**.
- As regras do grupo de segurança da instância foram configuradas corretamente.
  a. Na seção **Network** na página de guia **Basic Information**, clique no nome do grupo de segurança.
  b. Clique na guia **Inbound Rules** para exibir as regras de entrada do grupo de segurança.
     i. SSL desativado

        Para o acesso intra-VPC, o acesso de entrada através da porta 5672 deve ser permitido.

        Para acesso público, o acesso de entrada através da porta 15672 deve ser permitido.
     ii. SSL ativado

        Para o acesso intra-VPC, o acesso de entrada através da porta 5671 deve ser permitido.

        Para acesso público, o acesso de entrada através da porta 15671 deve ser permitido.

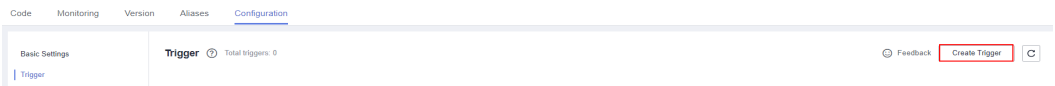
Criação de um gatilho de RabbitMQ

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Escolha **Configuration** > **gatilho** e clique em **Create gatilho**.

Figura 4-14 Criação de um gatilho



Passo 4 Defina os seguintes parâmetros:

- **Trigger Type**: Selecione DMS (para RabbitMQ).

- ***Instance**: Selecione uma instância de RabbitMQ.
- ***Password**: Digite a senha da instância de RabbitMQ.
- ***Exchange**: Digite o nome da troca a ser usada.
- **VM**: Insira um nome de VM personalizado.
- ***Batch Size**: Defina o número de mensagens a serem recuperadas de um tópico a cada vez.

Passo 5 Clique em **OK**.

----Fim

NOTA

Depois que o acesso à VPC for ativado, você precisará configurar as permissões de sub-rede correspondentes para o grupo de segurança de RabbitMQ. Para obter detalhes sobre como configurar o acesso à VPC, consulte **Configuração da rede**.

Configurando um evento de DMS (para RabbitMQ) para acionar a função

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique na função a ser configurada para ir para a página de detalhes da função.

Passo 3 Na página de detalhes da função, selecione uma versão.

Passo 4 Na página de guia **Code**, clique em **Test**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 5 Defina os parâmetros descritos em **Tabela 4-12** e clique em **Save**.

Tabela 4-12 Informações sobre o evento de teste

Parâmetro	Descrição
Configurar evento de teste	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Modelo de evento	Selecione **rabbitmq-event-template**.
Nome do evento	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **kafka-123test**.
Dados do evento	O sistema carrega automaticamente o modelo de evento de RabbitMQ integrado, que é usado neste exemplo sem modificações.

NOTA

O modelo de evento é o seguinte:

```
{
    "event_version": "v1.0",
    "event_time": 1576737962,
    "trigger_type": "RABBITMQ",
    "region": "xx-xxxx-1",
    "records": [{
           "messages": [
                   "rabbitmq message1",
                   "rabbitmq message2",
                   "rabbitmq message3",
                   "rabbitmq message4",
                   "rabbitmq message5"
            ],
    "instance_id": "81335d56-b9fe-4679-ba95-7030949cc76b",
    "exchange": "exchange-test"
    }]
}
```

Passo 6 Clique em **Test**. O resultado do teste de função é exibido.

----Fim

4.15 Apêndice: expressões Cron para um gatilho de Timer de função

Você pode configurar uma expressão cron nos seguintes formatos para um gatilho de timer de função:

- @todos os formatos

  O formato é "@cada unidade *N*". *N* é um inteiro positivo. **unit** pode ser ns, µs, ms, s, m ou h. Uma expressão @every significa invocar uma função a cada *N* unidades de tempo, como mostrado em **Tabela 4-13**.

  Tabela 4-13 Exemplos de expressões

Expressão	Significado
@every 30m	Aciona uma função a cada 30 minutos.
@every 1.5h	Aciona uma função a cada 1,5 horas.
@every 2h30m	Aciona uma função a cada 2,5 horas.

- Formato padrão

  O formato é "*segundos minutos horas dia-do-mês mês dia-da-semana*". *dia-da-semana* é opcional. Os campos devem ser separados uns dos outros usando um espaço. **Tabela 4-14** descreve os campos em uma expressão cron padrão.

Tabela 4-14 Descrição do parâmetro

Parâmetro	Descrição	Intervalo de valores	Caracteres especiais permitidos
Seconds	Sim	0-59	, - * /
Minutes	Sim	0-59	, - * /
Hours	Sim	0-23	, - * /
Day-of-month	Sim	1-31	, - * ? /
Month	Sim	1 – 12 ou jan–dez. O valor não diferencia maiúsculas de minúsculas, como mostrado na **Tabela 4-15**.	, - * /
Day-of-week	Não	0 – 6 ou Dom–Sáb. O valor não diferencia maiúsculas de minúsculas, como mostrado na **Tabela 4-16**. **0** significa domingo.	Comentários – *? /

Tabela 4-15 Descrição do valor do campo do mês

Mês	Dígito	Abreviação
Janeiro	1	Jan
Fevereiro	2	Fev
Março	3	Mar
Abril	4	Abr
Maio	5	Mai
Junho	6	Jun
Julho	7	Jul
Agosto	8	Ago
Setembro	9	Set
Outubro	10	Out
Novembro	11	Nov

Mês	Dígito	Abreviação
Dezembro	12	Dez

Tabela 4-16 Descrição do valor do campo dia da semana

Dia da semana	Dígito	Abreviação
Segunda-feira	1	Seg
Terça-feira	2	Ter
Quarta-feira	3	Qua
Quinta-feira	4	Qui
Sexta-feira	5	Sex
Sábado	6	Sáb
Domingo	0	Dom

Tabela 4-17 descreve os caracteres especiais que podem ser usados em uma expressão cron.

Tabela 4-17 Descrição de caracteres especiais

Caractere especial	Significado	Descrição
*	Usado para especificar todos os valores dentro de um campo.	* no campo minutos significa cada minuto.
,	Usado para especificar vários valores, que podem ser descontínuos.	Por exemplo, "Jan,Abr,Jul,Outubro" ou "1,4,7,10" no campo mês e "Sat,Sun" ou "6,0" no campo dia da semana.
-	Usado para especificar um intervalo.	Por exemplo, "0-3" no campo minutos.
?	Usado para especificar algo em um dos dois campos em que o caractere é permitido, mas não o outro.	Você pode especificar algo apenas no campo dia do mês ou dia da semana. Por exemplo, se você quiser que sua função seja executada em um dia específico (como o dia 10) do mês, mas não se importar com qual dia da semana é, em seguida, coloque "10" no campo dia-do-mês e "?" no campo dia-da-semana.

Caractere especial	Significado	Descrição
/	Usado para especificar incrementos. O caractere antes da barra indica quando iniciar e o caractere após a barra representa o incremento.	Por exemplo, "1/3" no campo minutos significa acionar a função a cada 3 minutos a partir das 00:01:00 da hora.

Tabela 4-18 descreve vários exemplos de expressões cron.

Tabela 4-18 Exemplo de expressões cron

Expressão Cron	Exemplo
0 15 2 * * ?	Executa uma função às 02:15:00 todos os dias.
0 30 8? * Seg-Sex	Executa uma função às 08:30:00 todas as segundas a sextas-feiras.
0 45 7 1-3 * ?	Executa uma função às 07:45:00 nos primeiros três dias de cada mês.
*? 0 0/3 * Seg, quarta, sexta-feira, domingo	Executa uma função a cada 3 minutos em todas as segundas, quartas, sextas e domingos.
0 0/3 9-18? * Seg-Sex	Executa uma função a cada 3 minutos entre as 09:00 e as 18:00 de segunda a sexta-feira.
0 0/30 * * * ?	Executa uma função a cada 30 minutos.

5 Depuração da função

5.1 Depuração online

Precauções

Os dados de eventos são passados para o manipulador de sua função como uma entrada. Após a configuração, os dados do evento são persistidos para uso posterior. Cada função pode ter um máximo de 10 eventos de teste.

Criação de um evento de teste

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função desejada.

Passo 3 Na página de detalhes da função, selecione uma versão e clique em **Test**.

Passo 4 Na caixa de diálogo **Configure Test Event**, configure as informações do evento de teste de acordo com **Tabela 5-1**. O parâmetro marcado com um asterisco (*) é obrigatório.

Tabela 5-1 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Você pode optar por criar um evento de teste ou editar um existente. Use a opção padrão **Create new test event**.
Event Template	Se você selecionar **blank-template**, poderá criar um evento de teste do zero. Se você selecionar um modelo, o evento de teste correspondente no modelo será carregado automaticamente. Para obter detalhes sobre modelos de evento, consulte **Tabela 5-2**.

Parâmetro	Descrição
*Event Name	O nome do evento pode conter de 1 a 25 caracteres e deve começar com uma letra e terminar com uma letra ou dígito. Apenas letras, dígitos, sublinhados (_), e hífens (-) são permitidos. Por exemplo, **even-123test.**
Event data	Digite um evento de teste.

Tabela 5-2 Descrição do modelo de evento

Nome do modelo	Descrição
modelo em branco	O evento de modelo é **{"key": "value"}**, que pode ser alterado com base nos requisitos.
apig-event-template	Simula um evento do API Gateway para acionar sua função.
dis-event-template	Simula um evento DIS para acionar sua função.
smn-event-template	Simula um evento SMN para acionar sua função.
obs-evento-modelo	Simula um evento do OBS para acionar sua função.
timer-event-template	Simula um evento de temporizador para acionar sua função.
lts-evento-modelo	Simula um evento LTS para disparar sua função.
cts-event-template	Simula um evento CTS para acionar sua função.
dds-event-template	Simula um evento DDS para acionar sua função.
kafka-evento-template	Simula um evento Kafka para acionar sua função.
coelhomq-evento-modelo	Simula um evento RabbitMQ para acionar sua função.
gaussmongo-evento-modelo	Simula um evento GaussDB(para Mongo) para acionar sua função.
login-security-template	Serve como uma entrada para o template de função **loginSecurity-realtime-analysis-python**.

Nome do modelo	Descrição
porn-imagem-analyse	Serve como uma entrada para o modelo de função **porn-image-analysis**.
análise de voz	Serve como uma entrada para o modelo de função de **voice-analysis**.
imagem-tag	Serve como uma entrada para os modelos de função **image-tag** e **porn-image-analysis**.

Passo 5 Clique em **Save**.

----Fim

Teste de uma função

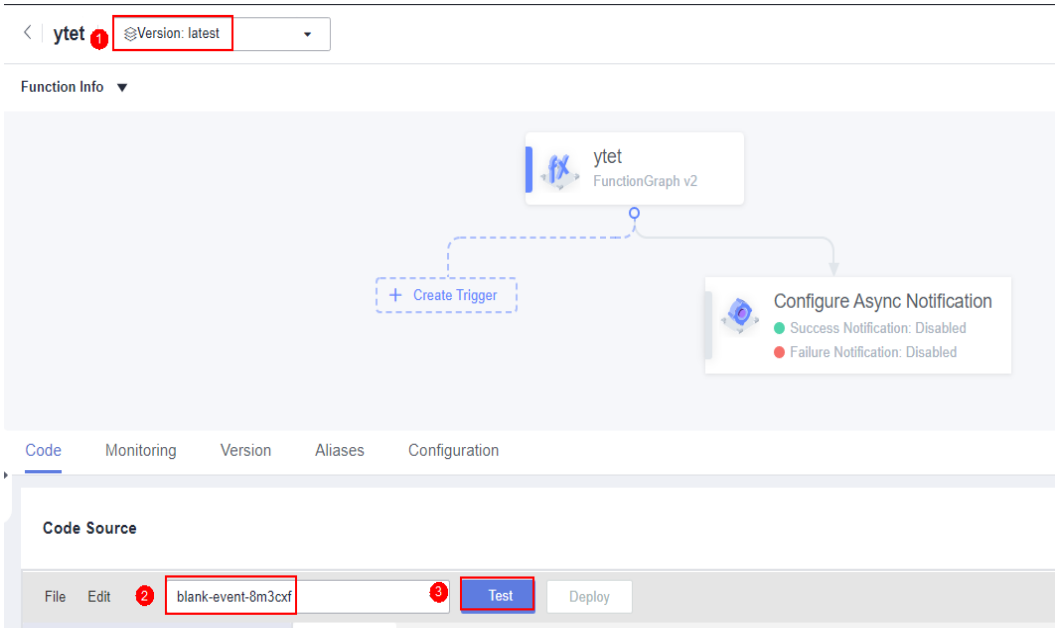
Depois de criar uma função, você pode testá-la on-line para verificar se ela pode ser executada corretamente como esperado.

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique no nome da função desejada.

Passo 3 Na página de detalhes da função exibida, selecione uma versão e um evento de teste e clique em **Test**.

Figura 5-1 Selecionamento de um evento de teste



Passo 4 Clique em **Test**. O resultado do teste de função é exibido.

NOTA

A área **Log Output** exibe um máximo de 2 KB de logs. Para ver mais registos, consulte **Gerenciamento de registrações de função**.

----Fim

Modificação de um evento de teste

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em um nome de função.

Passo 3 Na página de detalhes da função exibida, selecione uma versão e clique em **Configure Test Event**. A caixa de diálogo **Configure Test Event** é exibida.

Passo 4 Na caixa de diálogo **Configure Test Event**, modifique as informações do evento de teste de acordo com **Tabela 5-3**.

Tabela 5-3 Informações sobre o evento de teste

Parâmetro	Descrição
Create new test event	Criar um evento de teste.
Edit saved test event	Modificar um evento de teste existente.
Event data	Modificar o código do evento de teste.

Passo 5 Clique em **Save**.

----Fim

Exclusão de um evento de teste

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em um nome de função.

Passo 3 Na página de detalhes da função exibida, selecione uma versão e escolha **Select test event** > **Configure test event**.

Passo 4 Na caixa de diálogo **Configure Test Event**, selecione o evento de teste que você deseja excluir de acordo com **Tabela 5-4**.

Tabela 5-4 Informações sobre o evento de teste

Parâmetro	Descrição
Configure Test Event	Selecione **Edit saved test event**.
Saved Test Event	Selecione o evento de teste que deseja excluir.

Passo 5 Clique em **Delete**.

----Fim

5.2 Depuração local com VSCode

Visão geral

O FunctionGraph é um plug-in do Visual Studio Code (VSCode) dos produtos sem servidor do Huawei Cloud. Com este plug-in, você pode:

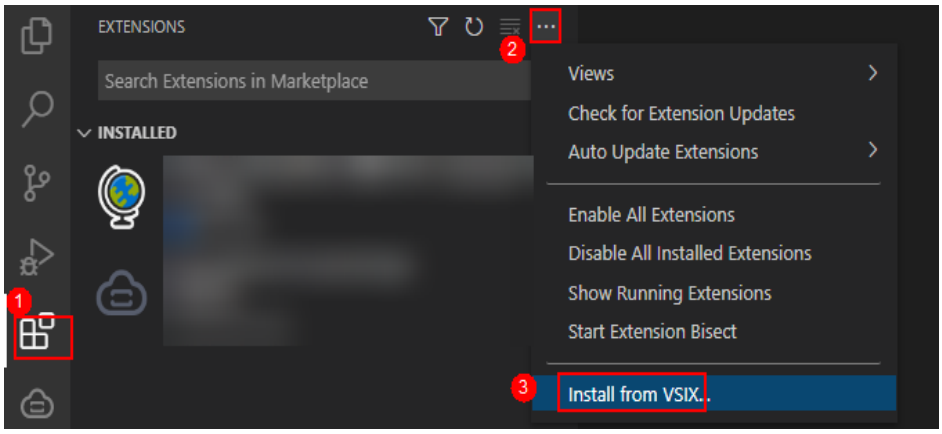
- Crie rapidamente uma função local.
- Execute e depure uma função local e implante-a na nuvem.
- Puxe a lista de funções da nuvem, chame funções de nuvem e faça upload de pacotes ZIP para a nuvem.

Pré-requisitos

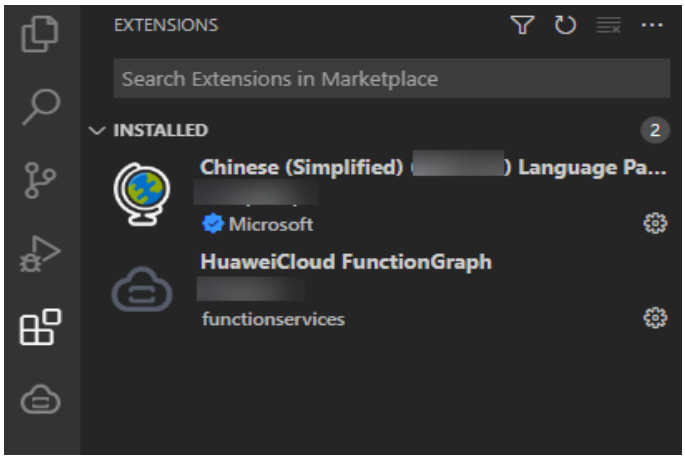
Você baixou a **ferramenta VSCode (posterior a 1.54.0)** e instalou-a.

Você baixou o plug-in **FunctionGraph da Huawei Cloud**.

Instalando o plug-in

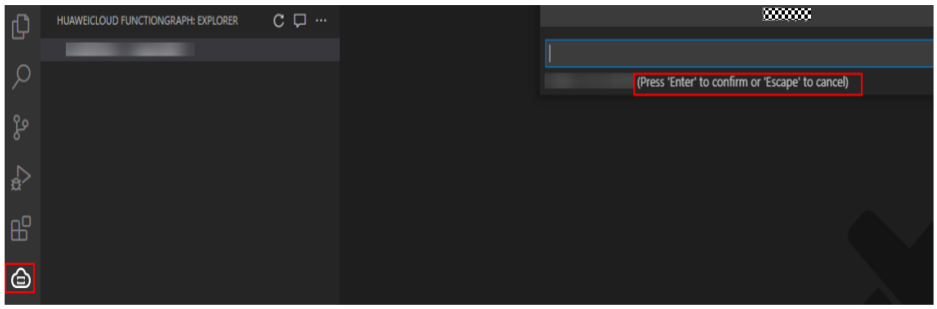
1. Abra a ferramenta VSCode, escolha **Extensions** na árvore de navegação à esquerda e escolha **...** > **Install from VSIX** à direita do painel de extensão.
2. 

3. Na página exibida, importe o plug-in FunctionGraph do Huawei Cloud.
4. Após a instalação ser bem-sucedida, o plug-in será exibido.

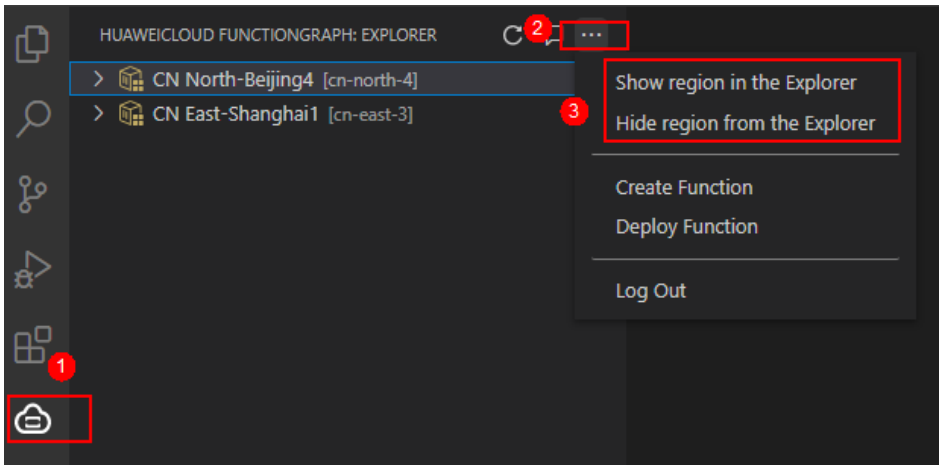


Execução de login no plug-in FunctionGraph

1. Abra o plug-in FunctionGraph do Huawei Cloud à esquerda, use sua conta do Huawei Cloud IAM para fazer login e insira as informações da conta conforme solicitado.



2. Selecione uma região para exibir as informações da função.
   - **Show region in the Explorer**: Selecione a região de destino.
   - **Hide region from the Explorer**: Ocultar regiões que não lhe dizem respeito.

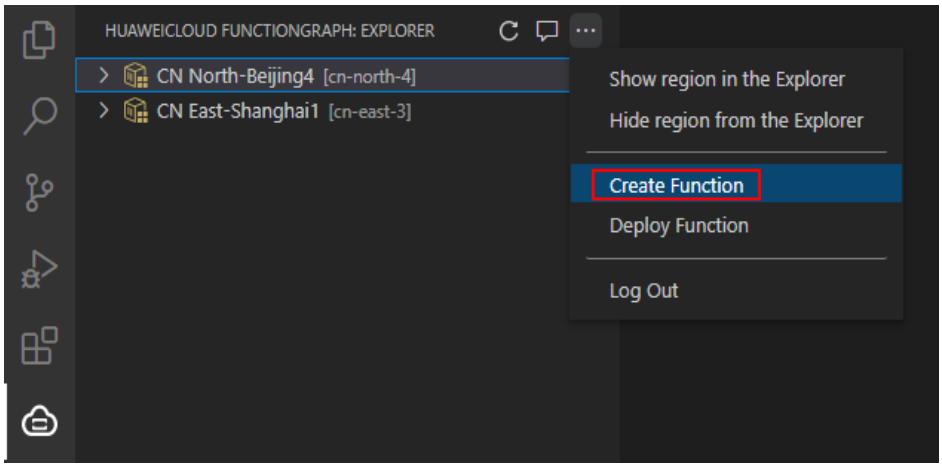


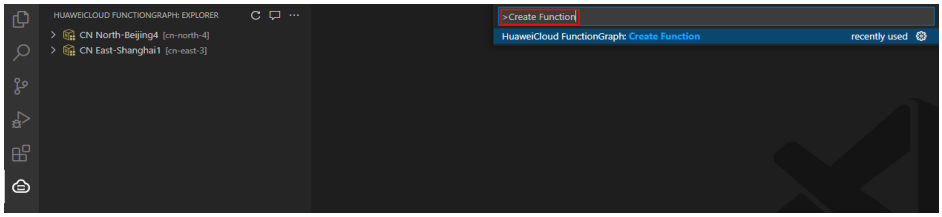
Criação de uma função

Crie uma função em um caminho local.

1. No painel de plug-in, selecione **Create Function** ou pressione **Ctrl+Shift+p** para procurar o comando **Create Function**. Em seguida, selecione ou insira o tempo de

execução, o modelo, o nome da função e a pasta local conforme solicitado. Uma função é criada na pasta especificada.





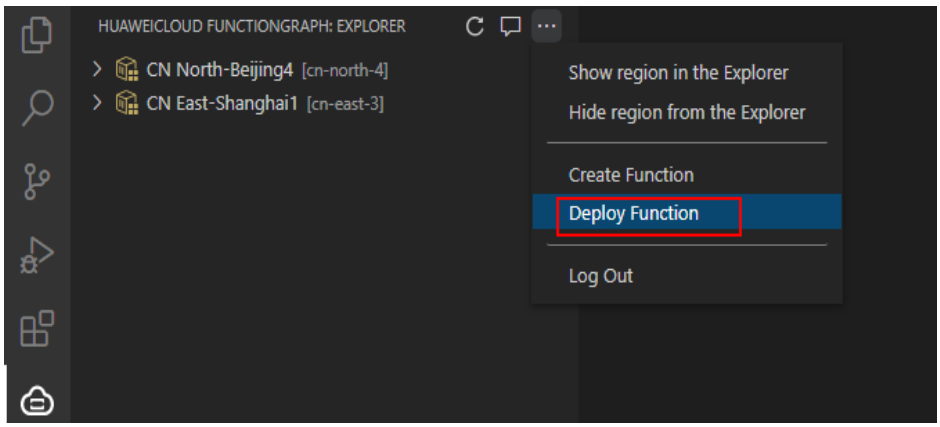
2. Depois que uma função local é criada, o arquivo do manipulador é aberto automaticamente.

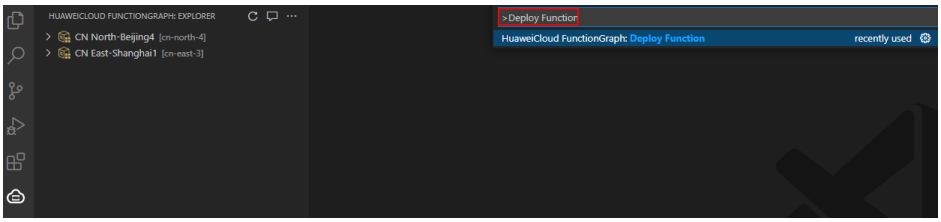
Implantação de uma função

Implante uma função local na plataforma.

No painel de plug-in, selecione **Deploy Function** ou pressione **Ctrl+Shift+p** para procurar o comando **Deploy Function** e selecione a função a ser implantada e a região conforme solicitado.

- Se a implantação for bem-sucedida, uma mensagem de sucesso será exibida no canto inferior direito da página. Alterne para a região de destino para exibir o resultado da implantação.
- Se a implantação falhar, visualize o log de erros na área **Output** e corrija a falha.





Depuração Local

Node.js

Crie o arquivo **main.js** na pasta de funções e copie o seguinte conteúdo para o arquivo. Clique em **Run** e **Debug** à esquerda. Em seguida, clique em **Add Config**, selecione Node.js e pressione **F5** para depuração. **(Prerequisite: Node.js has been installed.)**

```
const handler = require('./index'); //Path of the function handler file. Change
it based on site requirements.
const event = { 'hello': 'world' }; //Test event. Modify it based on site
requirements.
const context = {}; // Context Class
console.log(handler.handler(event, context));
```

Python

Crie o arquivo **main.py** na pasta de funções e copie o seguinte conteúdo para o arquivo. Clique em **Run** e **Debug** à esquerda. Em seguida, clique em **Add Config**, selecione Python e pressione **F5** para depuração. **(Prerequisite: Python has been installed.)**

```
import sys
import index #Path of the function handler file. Change it based on site
requirements.
# The main method is used for debugging, and event is the selected debugging
event.
if __name__ == '__main__':
....event = { 'hello': 'world' } # Test event. Modify it based on site
requirements.
    context = ''
    content = index.handler(event, context)
....print('Returned value:')
    print(content)
```

Outras funções

- **Opening in Portal**

  Clique com o botão direito do mouse em uma função e escolha **Open in Portal** no menu de atalho. A página de detalhes da função é exibida.

- **Executing a Cloud Function**

  a. Clique com o botão direito do mouse na função de destino e escolha **Invoke Function...** a partir do menu de atalho.

  b. No painel **Invoke Function**, selecione o evento a ser transferido e clique em **Invoke**. O log de função e o resultado são exibidos na área **Output**.

- **Downloading a Cloud Function**

  Clique com o botão direito do mouse na função que deseja baixar e escolha **Download...** a partir do menu de atalho. O código de função é baixado da nuvem para o caminho local especificado e o arquivo do manipulador é aberto automaticamente.

- **Updating a Cloud Function**

  Clique com o botão direito do mouse na função alvo, escolha **Upload Function...** no menu de atalho e selecione um pacote ZIP para carregar.

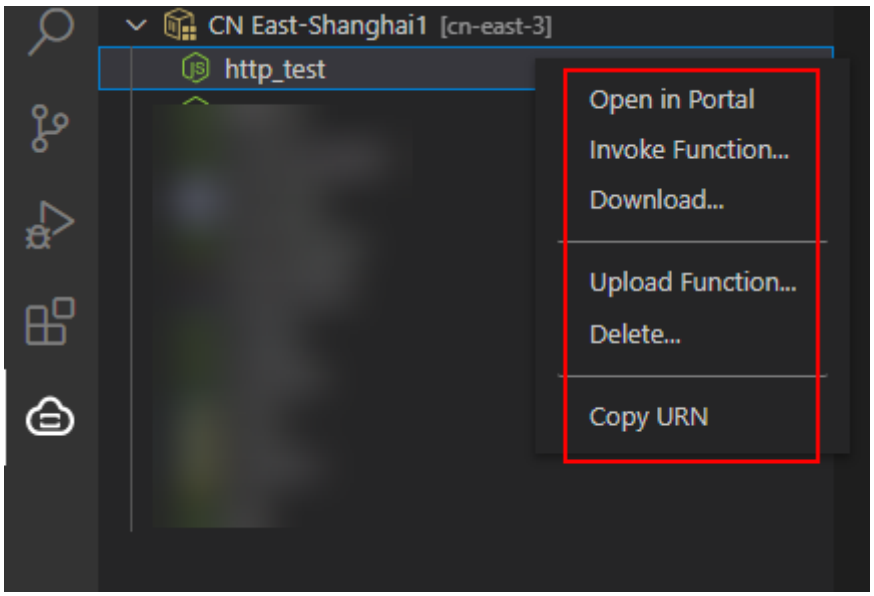
- **Deleting a Cloud Function**

  a. Clique com o botão direito do mouse na função a ser excluída e escolha **Delete...** a partir do menu de atalho.

  b. Na caixa de diálogo de confirmação, clique em **Delete** para excluir a função.

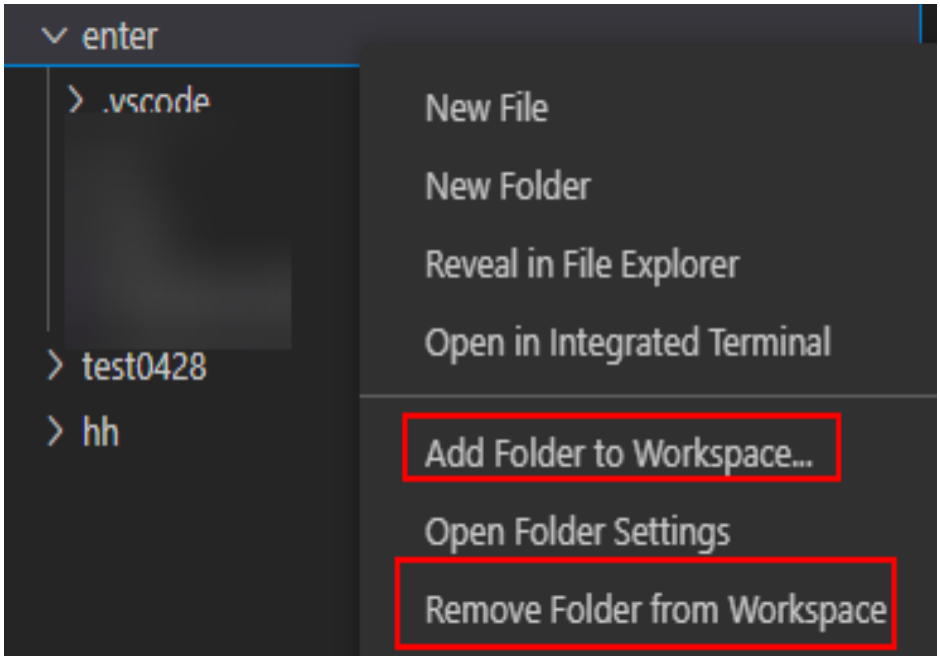
- **Copy URN**

  Clique com o botão direito do mouse na função cujo URN precisa ser copiado e escolha **Copy URN** no menu de atalho.



- **Viewing a Local Function**

  Vá para o gerenciador de recursos ou configure a exibição por meio de **Remove Folder from Workspace** e **Add Folder to Workspace**.



6 Invocação da função

6.1 Invocação síncrona

Os clientes esperam por respostas explícitas às suas solicitações de uma função. As respostas são retornadas somente depois que a função é chamada. Para mais detalhes, veja **Depuração online**.

6.2 Invocação assíncrona

Os clientes não se importam com os resultados de invocação de função de suas solicitações. Depois de receber uma solicitação, o FunctionGraph a coloca em uma fila, retorna uma resposta e processa outras solicitações quando há recursos ociosos.

Se uma função for chamada de forma assíncrona, o FunctionGraph repetirá automaticamente o evento, com um intervalo entre as tentativas. Eventos assíncronos são enfileirados antes de serem usados para invocar uma função.

Tabela 6-1 lista as origens de eventos e os modos de invocação suportados. Para obter detalhes sobre como gerenciar gatilhos, consulte **Criação de gatilhos**.

Tabela 6-1 Modos de invocação de função

Origem do evento	Modo de invocação
SMN	Invocação assíncrona
DMS	Invocação assíncrona
APIG	Invocação síncrona
OBS	Invocação assíncrona
DIS	Invocação assíncrona
Timer	Invocação assíncrona
DMS para Kafka	Invocação assíncrona

Origem do evento	Modo de invocação
GaussDB(para Mongo)	Invocação assíncrona

NOTA

Se a latência de execução da função E2E exceder 90s, a invocação assíncrona é recomendada. Se a invocação síncrona for usada, nenhuma resposta poderá ser recebida após 90s devido a restrições de gateway.

6.3 Mecanismo de repetição

Se a invocação síncrona ou assíncrona falhar, faça o seguinte:

- Invocação síncrona

  Tente novamente.

- Invocação assíncrona

  Os métodos de repetição variam dependendo dos tipos de erro. Para mais detalhes, veja **Tabela 6-2**.

Tabela 6-2 Repetir métodos para diferentes tipos de erro

Tipo de erro	Mecanismo de repetição
O código do usuário é anormal.	Tentar novamente não é suportado.
Ocorre um erro interno no serviço de função.	O FunctionGraph oferece suporte a novas tentativas de invocação assíncrona. Você pode configurar o número máximo de tentativas e o período máximo de validade da mensagem referindo-se a **Configuração da notificação de execução assíncrona**. As tentativas serão realizadas com base no número máximo de tentativas predefinido e no período máximo de validade da mensagem (até 24 horas).

7 Monitoramento

7.1 Métricas

7.1.1 Monitoramento da função

O FunctionGraph é interconectado com o Cloud Eye, permitindo visualizar métricas de função sem a necessidade de configurações.

Exibindo Métricas de Função

O FunctionGraph coleta métricas de função e exibe resultados agregados. Alterne para a versão da função de destino antes de visualizar as métricas.

1. Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.
2. Clique na função a ser configurada para ir para a página de detalhes da função.
3. Escolha **Monitoring** > **Metrics**, selecione um intervalo (5 minutos, 15 minutos ou 1 hora) e verifique o status de execução da função.

   NOTA

   As seguintes métricas são exibidas: chamadas, erros, duração (durações máxima, média e mínima), throttles e estatísticas de instância.

Descrição métrica

Tabela 7-1 descreve as métricas da função.

Tabela 7-1 Métricas de função

Métrica	Unidade	Descrição
Invocações	Contagem	Número total de solicitações de invocação, incluindo erros de invocação e invocações estranguladas. No caso de invocação assíncrona, a contagem começa somente quando uma função é executada em resposta a uma solicitação.
Duração	ms	**Maximum Duration**: a duração máxima de execução de uma função dentro de um período. **Minimum Duration**: a duração mínima que uma função é executada dentro de um período. **Average Duration**: a duração média da execução de uma função dentro de um período.
Erros	Contagem	Número de vezes que suas funções falharam com o código de erro **200** sendo retornado. Erros causados pela sintaxe da função ou execução também estão incluídos.
Aceleradores	Contagem	Número de vezes que o FunctionGraph limita suas funções devido ao limite de recursos.
Estatísticas da Instância	Contagem	Números de solicitações concorrentes e instâncias reservadas.

7.1.2 Métricas da função

Introdução

Esta seção descreve os namespaces, as métricas de função, e as dimensões do FunctionGraph relatadas ao Cloud Eye. Você pode visualizar métricas de função e alarmes usando o console do Cloud Eye ou chamando as API.

Namespaces

SYS.FunctionGraph

Métricas de Função

Tabela 7-2 Métricas de função

ID da métrica	Nome da métrica	Descrição	Intervalo de valores	Objeto Monitorado	Período de monitoramento de dados brutos (minuto)
count	Invocations	Número de chamadas de função Unidade: Contagem	≥ 0 contagens	Características	1
failcount	Errors	Número de erros de invocação Estão incluídos os seguintes erros: ● Erro de solicitação de função (causando uma falha de execução e retornando o código de erro 200) ● Sintaxe da função ou erro de execução Unidade: Contagem	≥ 0 contagens	Características	1
rejectcount	Throttles	Número de aceleradores de função Ou seja, o número de vezes que o FunctionGraph acelera suas funções devido ao limite de recursos. Unidade: Contagem	≥ 0 contagens	Características	1
concurrency	Number of concurrent requests	Número de solicitações concorrentes durante a invocação da função. Unidade: Contagem	≥ 0 contagens	Características	1

ID da métrica	Nome da métrica	Descrição	Intervalo de valores	Objeto Monitorado	Período de monitoramento de dados brutos (minuto)
reservedinstancenum	Number of reserved instances	Número de instâncias reservadas Unidade: Contagem	≥ 0 contagens	Características	1
duration	Average duration	Duração média da invocação da função Unidade: ms	≥ 0 ms	Características	1
maxDuration	Maximum duration	Duração máxima da invocação da função Unidade: ms	≥ 0 ms	Características	1
minDuration	Minimum duration	Duração mínima da invocação da função Unidade: ms	≥ 0 ms	Características	1

Dimensões

Chave	Valor
package-functionname	*Nome do aplicativo-Nome da função* Exemplo: default-myfunction_Python

7.1.3 Criação de uma regra de alarme

Depois de criar uma função e um gatilho, você pode monitorar a invocação e os status de execução da função em tempo real.

Visualização de métricas de função

O FunctionGraph diferencia as métricas de uma função por versão, permitindo consultar as métricas de uma versão específica da função.

Procedimento

Crie uma regra de alarme para uma função relatar métricas ao Cloud Eye para que você possa visualizar gráficos de monitoramento e mensagens de alarme no console do Cloud Eye.

Passo 1 Efetue log-in no **Console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

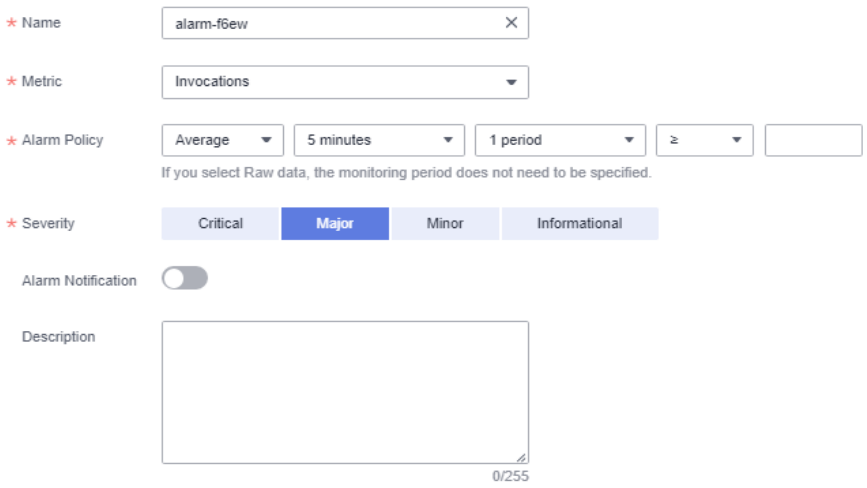
Passo 2 Clique no nome da função desejada.

Passo 3 Na página de detalhes da função exibida, selecione uma versão ou alias de função e escolha **Monitoring** > **Metrics**.

Passo 4 Clique em **Create Alarm Rule**.

Passo 5 Na caixa de diálogo **Create Alarm Rule**, defina os parâmetros de alarme e clique em **Next**, conforme mostrado na **Figura 7-1**.

Figura 7-1 Criação de uma regra de alarme



Passo 6 Insira um nome de regra e clique em **OK**.

----Fim

Métricas de Função

Tabela 7-3 lista as métricas de função que podem ser monitoradas pelo Cloud Eye.

Tabela 7-3 Métricas de função

Métrica	Nome de exibição	Descrição	Unidade	Limite máximo	Limite mínimo	Limite recomendado	Tipo do valor	Dimensão
contagem	Invocations	Número de chamadas de função	contagem	-	0	-	int	nome-função-do-pacote

Métrica	Nome de exibição	Descrição	Unidade	Limite máximo	Limite mínimo	Limite recomendado	Tipo do valor	Dimensão
contagem de falhas	Errors	Número de erros de invocação	contagem	-	0	-	int	nome-função-do-pacote
contagem de rejeição	Throttles	Número de aceleradores de função	contagem	-	0	-	int	nome-função-do-pacote
Duração	Average Duration	Duração média da invocação da função	ms	-	0	-	int	nome-função-do-pacote
maxDuração	Maximum Duration	Duração máxima da invocação da função	ms	-	0	-	int	nome-função-do-pacote
minDuração	Minimum Duration	Duração mínima da invocação da função	ms	-	0	-	int	nome-função-do-pacote

7.2 Registrações

7.2.1 Consulta de registrações de função

O FunctionGraph é interconectado com o LTS, permitindo que você visualize registrações de funções sem a necessidade de configurações.

Visualização de registrações de função

No console do FunctionGraph, os registrações de função de exibição das seguintes maneiras:

- Visualização de registrações na página de resultado da execução

  Depois de criar uma função, teste-a e visualize os registrações de teste na página de resultados da execução. Para mais detalhes, veja **Depuração online**.

  A página de resultados da execução exibe registrações de no máximo 2 KB. Para ver mais registrações da função, vá para a página da guia **Logs**.

- Visualização de registrações na página de guia **Logs**

Na página de detalhes da função, escolha **Monitoring** > **Logs** para consultar informações de registro. Para mais detalhes, veja **Gerenciamento de registrações de função**.

Download de registrações

Depois de consultar os registrações de uma versão de função dentro de um intervalo de datas especificado, você pode baixar os registrações para análise posterior.

NOTA

- Um máximo de 5000 registrações podem ser baixados de cada vez. Ao consultar registrações, selecione um intervalo de tempo adequado para evitar a perda de registração.
- O carimbo de data/hora dos registrações está em UTC.

7.2.2 Gerenciamento de registrações de função

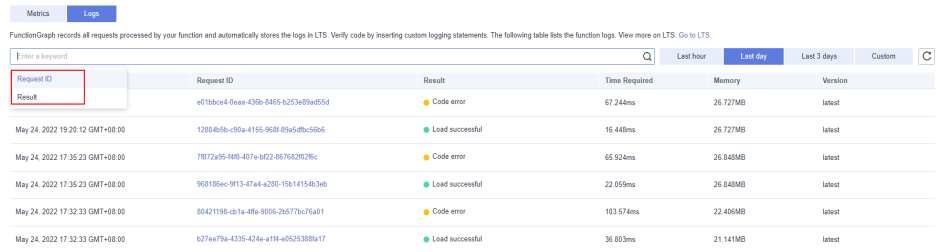
AVISO

O FunctionGraph suporta a gestão de registrações utilizando AOM ou LTS.

O FunctionGraph suporta a gestão de registrações utilizando apenas LTS.

- Uso de AOM para gerenciar registrações de função

- Uso de LTS para gerenciar registrações de função



Uso de AOM para gerenciar registrações de funções

Passo 1 Efetue login no **console do FunctionGraph**. No painel de navegação, escolha **Functions** > **Function List**.

Passo 2 Clique no nome da função desejada.

Passo 3 Escolha **Monitoring** > **Logs**. Na página exibida, informe os critérios de pesquisa.

NOTA

- Você pode:

  1. Informe uma palavra-chave para a pesquisa exata. Uma palavra-chave é uma palavra entre dois delimitadores adjacentes.

  2. Digite uma palavra-chave com curingas para correspondência difusa. Exemplo: ***ROR***, **ERR***, ou **ER*OR**.

  3. Informe uma frase para a pesquisa exata. Exemplo: **Start to refresh alm Statistic**.

  4. Use E (&&) ou OU (||) para combinar palavras-chave para pesquisar. Exemplo: **query&&logs** ou **query||logs**.

- Você pode selecionar **Last 30 minutes**, **Last hour**, **Last day**, e **Custom** (até um mês, por exemplo, 2022/04/01 16:34:48–2022/05/01 16:34:48).
- Você pode consultar registrações por versão.

Passo 4 Clique em  para procurar registrações.

Figura 7-2 Consulta de registrações

NOTA

O resultado da consulta de registração contém hora, ID da solicitação, resultado da chamada, duração, memória, e versão.

Passo 5 Execute as seguintes operações, se necessário:

1. Pesquise registrações por palavra-chave.

2. Filtre registrações por status: **Error**, **Info**, **Error & Warning**, e **Error & Warning & Info**.

3. Visualize registrações em tela cheia.

4. **Baixar registrações**.

Figura 7-3 Outras operações

----Fim

Uso de LTS para gerenciar registrações de funções

Você pode ativar o LTS para gerenciar melhor as registrações de função. Depois de ativar o LTS, o FunctionGraph cria automaticamente um grupo de registrações e 20 fluxos de registrações. Os registrações de uma função serão armazenados consistentemente em um fluxo de registração aleatório. Por exemplo, se os registrações da função A forem

armazenados no fluxo de registração A durante sua primeira execução, seus registrações subsequentes sempre serão armazenados no fluxo de registração A. Um fluxo de registração pode conter os registrações de múltiplas funções.

NOTA

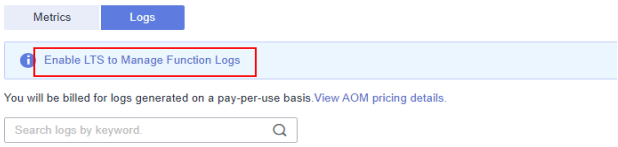
- Por padrão, são criados 20 fluxos de registração, que não podem ser personalizados. Na página de guia **Logs** da função, pressione **F12** para descobrir o ID do fluxo de registração da API **query**, e, em seguida, localize o ID do fluxo de registração correspondente no LTS.

- A exclusão de um grupo de registração de função por engano no console LTS não será detectada pelo FunctionGraph e os dados de registração histórico não poderão mais ser recuperados. Para usar um grupo de registrações, modifique a descrição da função e salve as alterações. Um novo grupo de registrações será criado.

Passo 1 Ative o LTS.

Ativação do LTS no FunctionGraph v1: Na página de guias **Logs**, clique em **Enable LTS to Manage Function Logs**.

Figura 7-4 Ativação do LTS para gerenciar registrações de funções

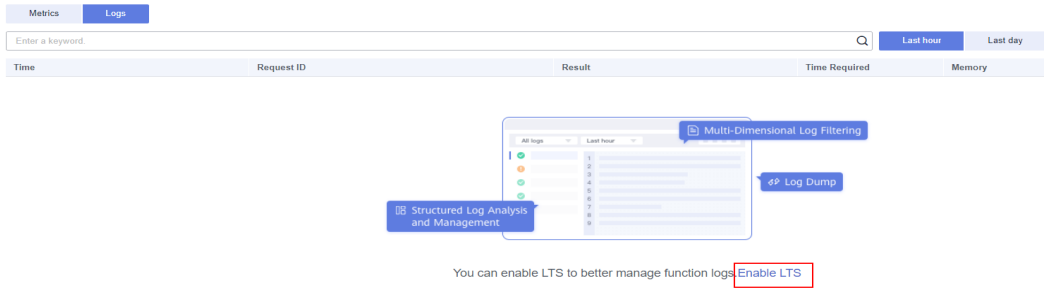


NOTA

No FunctionGraph, você pode alternar de volta para o AOM. Nesse caso, o AOM assumirá o LTS para gerenciar as registrações de função. Você será cobrado pelos registros gerados em uma base de pagamento por uso.

Enabling LTS in FunctionGraph v2: Na página **Logs**, clique em **Enable LTS**. Clique em **OK**. LTS ativado com sucesso.

Figura 7-5 Ativação o LTS para gerenciar registrações de funções



NOTA

No FunctionGraph, somente o LTS pode ser usado para gerenciar registrações de função.

Passo 2 Defina os critérios de filtragem.

Figura 7-6 Configuração de critérios de filtro

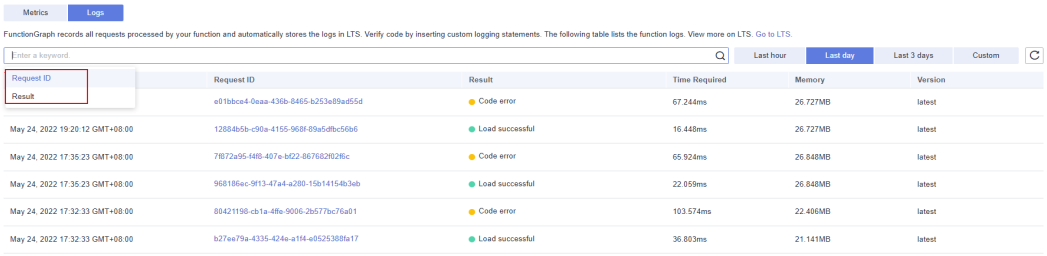
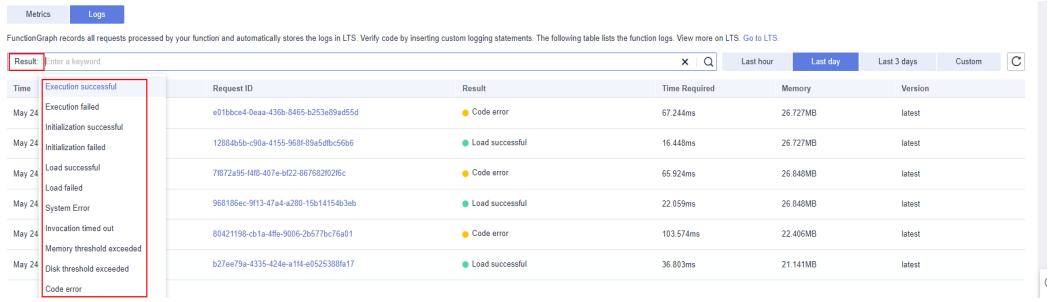


Figura 7-7 Consulta de resultados de invocação



Resultado da invocação	Descrição
Execução bem sucedida	Registração impressa quando uma função é executada com sucesso.
Falha na execução	Registração impressa quando uma função falha ao ser executada devido ao tempo limite de invocação, limite de memória ou disco excedido ou erros de código. Para exibir os registrações sobre o tempo limite de chamada, selecione **Invocation timed out** de chamada na lista suspensa. Os métodos para visualizar os outros três tipos de registrações são os mesmos.
Inicialização bem sucedida	Registração impressa quando uma função é inicializada com sucesso.
Falha na inicialização	Registração impressa quando a inicialização da função falha.
Carregado com sucesso	Registração gerada quando o tempo de execução carrega com sucesso o arquivo de função.
Falha ao carregar	Registração gerada quando o runtime falha ao carregar seu arquivo de função.

Resultado da invocação	Descrição
Erro do sistema	Erro interno.
Tempo limite de invocação	Registração impressa quando o período de invocação da função é maior que o limite predefinido.
Limite de memória excedido	Registração impressa quando o tamanho da memória da função excede o limite predefinido.
Limite de disco excedido	Registração impressa quando o tamanho do disco excede o limite predefinido.
Erro de código	Registração impressa quando ocorre um erro de código.

NOTA

- Você pode pesquisar registrações por ID de solicitação e resultado de invocação.
- Você pode selecionar **Last hour**, **Last day**, **Last 3 days**, e **Custom** (até um mês, por exemplo, 2022/04/01 16:34:48–2022/05/01 16:34:48).
- Você pode ir para o console LTS para executar mais operações.

----Fim

Download de registrações

AVISO

- Atualmente, os registrações podem ser baixados somente quando você usa o AOM para o gerenciamento de registrações.
- O FunctionGraph v1 permite que você gerencie registrações de funções usando o AOM.
- O FunctionGraph v2 permite que você gerencie registrações de função usando LTS, mas não suporta o download de registrações.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função desejada.

Passo 3 Clique em **Logs** para ir para a página de registração.

Passo 4 Selecione uma versão e um intervalo de tempo e clique em **Download Log**.

NOTA

Um máximo de 5000 registrações podem ser baixados de cada vez. Ao consultar registrações, selecione um intervalo de tempo adequado para evitar a perda de registração.

----Fim

8 Gerenciamento de funções

Visão geral

Função é uma combinação de código, tempo de execução, recursos e configurações necessárias para atingir um propósito específico. É a unidade mínima que pode funcionar de forma independente. Uma função pode ser acionada por gatilhos e agendar automaticamente os recursos e ambientes necessários para alcançar os resultados esperados.

Editação de código em linha

O FunctionGraph permite que você edite o código da função da mesma forma que gerencia um projeto. Você pode criar e editar arquivos e pastas. Depois de carregar um pacote de CEP, você pode exibir e editar o código no console. **Tabela 8-1** descreve os menus do editor inline.

Tabela 8-1 Menus do editor inline

Uso	Descrição
File	Cria arquivos e pastas. Você pode criar arquivos, criar pastas com base em modelos e fechar todos os arquivos.
Edit	Edita o código da função. Você pode executar a operação desfazer, editar ou comentar.
Find	Localiza e substitui o código.
Go to	Vai para um determinado local no código. Você pode ir para uma linha especificada, colchetes ou a próxima linha problemática.
View	Fornece funções comuns. É possível visualizar a paleta de comandos e alterar o tema de exibição.
Test	Testa uma função online e exibe o resultado do teste, o resumo e as registrações.

NOTA

- Java é uma linguagem compilada, que não suporta edição de código em linha. Se sua função não usar nenhuma dependência de terceiros, você poderá fazer upload de um arquivo JAR de função. Se a função usar dependências de terceiros, compacte as dependências e o arquivo JAR da função em um arquivo ZIP e, em seguida, carregue o arquivo ZIP.
- Se você editar o código em Go, compacte o arquivo compilado no formato ZIP e verifique se o nome do arquivo de biblioteca dinâmica é consistente com o nome do plug-in do manipulador. Por exemplo, se o nome do arquivo de biblioteca dinâmica for **function.so**, defina o manipulador como **function.Handler** de acordo com **Tabela 8-2**.
- Ao criar um arquivo ZIP, coloque o arquivo do manipulador sob o diretório **root** para garantir que seu código possa ser executado normalmente após ser descompactado.
- Diretório de trabalho: Uma função pode ler arquivos no diretório de código. O diretório de trabalho de uma função é o diretório de nível superior do arquivo do manipulador. Para ler o arquivo **test.conf** no mesmo nível de diretório que o arquivo do manipulador, use o caminho relativo **code/test.conf** ou um caminho absoluto (valor da variável de ambiente **RUNTIME_CODE_ROOT**). Para gravar um arquivo (por exemplo, criar ou baixar um arquivo), acesse o diretório **/tmp** ou use a função de montagem do sistema de arquivos fornecida pelo FunctionGraph.

Modificação do código da função

Depois que uma função é criada, a versão padrão é a mais recente. Cada função tem a versão mais recente. Você pode modificar uma função com base apenas em sua versão mais recente.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em um nome de função.

Passo 3 (Opcional) Na página de detalhes da função, selecione a versão mais recente e clique na guia **Code**.

Passo 4 Na página de guia **Code**, modifique as informações de código de acordo com **Tabela 8-2** e **Tabela 8-3**.

Tabela 8-2 Informação do código

Parâmetro	Descrição
Handler	• Para uma função Node.js ou Python, o manipulador pode ser nomeado no formato de *[nome do arquivo].[nome da função]*, que deve conter um ponto (.). ". Digite **myfunction.handler**, que indica que o nome do arquivo é **myfunction.js** (**myfunction.py** para uma função Python e **myfunction.java** para uma função Java) e o nome da função é **handler**. • Para uma função Java, o manipulador deve ser nomeado no formato de *[nome do pacote].[nome do arquivo].[nome da função]*, por exemplo, **com.*xxxxx*.exp.Myfunction.myHandler**. • Para uma função Go, o manipulador deve ser nomeado no formato de *[nome do plug-in].[nome da função]*, por exemplo, **function.Handler**. O nome da função deve começar com uma letra maiúscula e o nome do manipulador pode conter no máximo 128 caracteres. • Para uma função C#, o manipulador deve ser nomeado no formato de *[.NET assembly file name]*::*[namespace e classe da função do manipulador]*::*[handler function name]*, por exemplo, **HelloCsharp::Example.Hello::Handler**.
Initializer	Você pode ativar a inicialização da função na página de guia **Configuration**. O inicializador deve ser nomeado da mesma forma que o manipulador. Por exemplo, para uma função Node.js ou Python, defina um nome de inicializador no formato de *[nome do arquivo].[nome da função de inicialização]*. **NOTA** Este parâmetro não é necessário se a inicialização da função estiver desativada. Para obter detalhes sobre como criar uma função de inicialização, consulte **Inicializando uma função**.
Dependencies	Pacotes de software de terceiros exigidos pela função. Você pode gerenciar suas dependências enviando-as para FunctionGraph. Se as dependências forem muito grandes, carregue-as pelo OBS. Para obter mais informações, consulte **Bibliotecas dependentes**. **NOTA** Exceto suas dependências privadas, o FunctionGraph fornece algumas dependências comuns, que você pode escolher ao criar uma função.
Code Entry Mode	Método de inserção do código de função. Para obter detalhes sobre os modos de entrada de código suportados, consulte **Tabela 8-3**.

Tabela 8-3 Modos de entrada de código

Tempo de execução	Modo de entrada de código	Descrição
Node.js 6.10/ Node.js 8.10/	Editar código em linha	Edite o código na caixa de código. Para obter mais informações, consulte **Tabela 8-1**.

Tempo de execução	Modo de entrada de código	Descrição
Node.js 10.16/ Node.js 12.13/ Node.js 14.18	Carregar arquivo ZIP	Clique em **Select File** e faça upload de um pacote de código local para FunctionGraph. O tamanho do arquivo ZIP a ser carregado não pode exceder 50 MB. Se exceder 50 MB, carregue o arquivo ZIP usando um intervalo OBS.
	Carregar arquivo do OBS	Cole o URL do link do bucket do OBS que armazena um arquivo ZIP de código.
Python 2.7/Python 3.6/Python 3.9	Editar código em linha	Edite o código na caixa de código. Para obter mais informações, consulte **Tabela 8-1**.
	Carregar arquivo ZIP	Clique em **Select File** e faça upload de um pacote de código local para FunctionGraph. O tamanho do arquivo ZIP a ser carregado não pode exceder 50 MB. Se exceder 50 MB, carregue o arquivo ZIP usando um intervalo OBS.
	Carregar arquivo do OBS	Cole o URL do link do bucket do OBS que armazena um arquivo ZIP de código.
Java 8/Java 11	Carregar arquivo ZIP	Clique em **Select File** e faça upload de um pacote de código local para FunctionGraph. O tamanho do arquivo ZIP a ser carregado não pode exceder 50 MB. Se exceder 50 MB, carregue o arquivo ZIP usando um intervalo OBS.
	Carregar arquivo JAR	Clique em **Select File** e carregue um arquivo JAR local para FunctionGraph. O tamanho do arquivo JAR a ser carregado não pode exceder 50 MB. Se exceder 50 MB, converta-o em um arquivo ZIP e faça o upload do arquivo ZIP para o OBS.
	Carregar arquivo do OBS	Cole o URL do link do bucket do OBS que armazena um arquivo ZIP de código.
Go 1.8/Go 1.x	Carregar arquivo ZIP	Clique em **Select File** e faça upload de um pacote de código local para FunctionGraph. O tamanho do arquivo ZIP a ser carregado não pode exceder 50 MB. Se exceder 50 MB, carregue o arquivo ZIP usando um intervalo OBS.
	Carregar arquivo do OBS	Cole o URL do link do bucket do OBS que armazena um arquivo ZIP de código.

Tempo de execução	Modo de entrada de código	Descrição
C#(.NET Core 2.0)/ C#(.NET Core 2.1)/ C#(.NET Core 3.1)	Carregar arquivo ZIP	Clique em **Select File** e faça upload de um pacote de código local para FunctionGraph. O tamanho do arquivo ZIP a ser carregado não pode exceder 50 MB. Se exceder 50 MB, carregue o arquivo ZIP usando um intervalo OBS.
	Carregar arquivo do OBS	Cole o URL do link do bucket do OBS que armazena um arquivo ZIP de código.
PHP 7.3	Carregar arquivo ZIP	Clique em **Select File** e faça upload de um pacote de código local para FunctionGraph. O tamanho do arquivo ZIP a ser carregado não pode exceder 50 MB. Se exceder 50 MB, carregue o arquivo ZIP usando um intervalo OBS.
	Carregar arquivo do OBS	Cole o URL do link do bucket do OBS que armazena um arquivo ZIP de código.

AVISO

Se o código a ser carregado contiver informações confidenciais (como senhas de conta), criptografe as informações confidenciais para evitar vazamentos.

Passo 5 Clique em **Save**.

----Fim

Modificação de configurações de função

Você pode definir variáveis para sua função e modificar as configurações da função.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em um nome de função.

Passo 3 (Opcional) Na página de detalhes da função, selecione a versão mais recente e clique na guia **Configuration**.

Passo 4 Defina os parâmetros descritos em **Tabela 8-4**.

Tabela 8-4 Parâmetros de configuração

Parâmetro	Descrição
Description	Descrição da função, que não pode exceder 512 caracteres. Modifique a descrição conforme necessário.
Agency	Usado para delegar FunctionGraph para acessar outros serviços de nuvem. Por exemplo, uma agência é necessária quando a FunctionGraph acessa serviços em nuvem, como OBS ou SMN. Selecione uma agência na lista suspensa. Se não houver nenhuma agência na lista suspensa, clique em **Create Agency** para criar uma no console do IAM. Para mais detalhes, veja **Configuração das permissões de agência**.
Memory (MB)	Valor: 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2560, 3072, 3584 ou 4096. Modifique a descrição conforme necessário.
Initialization Timeout (s)	Duração máxima que a função pode ser inicializada. Defina este parâmetro se você ativar a inicialização da função. O valor varia de 1s a 300s.
Execution Timeout (s)	Duração máxima que a função pode ser executada. O valor varia de 3s a 900s.
Max. Requests per Instance	Número de solicitações simultâneas suportadas por uma única instância. Intervalo do valor: 1 – 100. Para mais detalhes, veja **Configuração da multi-corrência de instância única**.
Max. Instances per Function	Número máximo de instâncias nas quais uma função pode ser executada. **–1** (predefinição): A função pode ser executada em qualquer número de instâncias. **0**: A função está desativada. **NOTA** As solicitações que excederem a capacidade de processamento das instâncias serão descartadas. Erros causados por solicitações excessivas não serão exibidos nos registrações de função. Você pode obter detalhes do erro consultando **Configuração da notificação de execução assíncrona**.
VPC Access	Se o acesso à VPC estiver ativado para a função, especifique uma agência com permissões de gerenciamento da VPC.
Encryption Settings	Pares chave-valor definidos para armazenar informações sensíveis. As teclas podem conter letras, dígitos e sublinhados (_), e devem começar com uma letra. Os valores serão criptografados para armazenamento e não serão exibidos em texto simples. Clique em **Add now?** e definir chaves e valores. Para excluir um par chave-valor, clique em **Delete** ao lado dele.
Environment Variables	Pares chave-valor definidos para a função. As teclas podem conter letras, dígitos e sublinhados (_), e devem começar com uma letra. Clique em **Add** e defina chaves e valores. Para excluir um par chave-valor, clique em **Delete** ao lado dele.

NOTA

- A duração máxima para retornar um resultado é de 90s. Se o tempo limite definido e a duração real de execução de uma função excederem 90s, uma mensagem será exibida indicando que a função expirou. No entanto, o backend ainda está em execução. Você pode exibir o resultado do retorno na página da guia **Logs**.
- Quando você define variáveis de ambiente, o FunctionGraph exibe todas as informações de entrada em texto simples. Para garantir a segurança, não inclua informações confidenciais.

Passo 5 Clique em **Save**.

----Fim

Exportação de uma função

Você pode exportar as funções que você criou.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique em um nome de função.

Passo 3 Na página de detalhes da função exibida, escolha **Operation** > **Export function** no canto superior direito.

NOTA

- Um usuário pode exportar apenas uma função por vez.
- O pacote de recursos de função exportado não pode exceder 50 MB.
- O nome do pacote de recursos de função exportado está no formato de *nome da função*+*valor MD5 da função code*.zip.
- O pacote de recursos de função exportado não inclui informações de alias.

----Fim

Desativação de uma função

As funções desativadas não podem mais ser executadas.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função que você deseja desativar.

Passo 3 Na página de detalhes da função exibida, clique em **Disable** no canto superior direito.

Passo 4 Na página exibida, clique em **Yes**. A função está desativada.

NOTA

- Apenas as funções da versão mais recente podem ser desativadas.
- Versões publicadas com base na última versão desativada de uma função também são desativadas e nunca podem ser ativadas.
- Depois de desativar uma função, você pode modificar seu código, mas não pode executar a função.

----Fim

Ativação de uma função

As funções desativadas podem ser ativadas novamente conforme necessário.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função que deseja ativar.

Passo 3 Na página de detalhes da função exibida, clique em **Enable** no canto superior direito.

----Fim

Exclusão de uma função

Você pode excluir funções não utilizadas para liberar recursos.

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função que você deseja excluir.

Passo 3 Selecione um critério de pesquisa (nome da função, tempo de execução ou nome do aplicativo) no canto superior direito, insira uma palavra-chave e clique em para pesquisar a função que deseja excluir.

Passo 4 Clique em **Delete** na mesma linha da função.

Passo 5 Na caixa de diálogo **Delete Function**, clique em **Yes** para confirmar a exclusão.

----Fim

9 Gestão de dependências

O FunctionGraph permite gerenciar dependências de maneira unificada. Você pode carregar dependências de um caminho local ou por meio do OBS, se elas forem muito grandes, e especificar nomes para elas.

Para obter detalhes, consulte **Como posso criar dependências de função?**

NOTA

- O nome de cada arquivo no pacote de dependência não pode terminar com um til (~).
- Um pacote de dependência pode conter até 30.000 arquivos.
- Se sua dependência privada for grande, é aconselhável definir o tempo de execução da função para um valor grande.

Criação de uma dependência

Passo 1 Entre ao console do FunctionGraph, e escolha **Functions** > **Dependencies** no painel de navegação.

Passo 2 Clique em **Create Dependency**.

Passo 3 Defina os seguintes parâmetros:

- **Name**: Digite um nome de dependência.
- **Runtime**: Selecione um tempo de execução.
- **Description**: Insira uma descrição para a dependência. Este parâmetro é opcional.
- **Upload Mode**: Carregue um arquivo ZIP ou carregue um arquivo do OBS.

Passo 4 Clique em **OK**.

----Fim

Configuração de dependências para uma Função

Passo 1 Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.

Passo 2 Clique no nome da função desejada.

Passo 3 Na página de detalhes da função exibida, clique na guia **Code**, clique em **Add** na área **Dependencies**.

Passo 4 Na caixa de diálogo **Select Dependency** exibida, selecione dependências e clique em **OK**.

NOTA

- Você pode adicionar um máximo de 20 dependências para uma função.
- Exceto suas dependências privadas, o FunctionGraph fornece algumas dependências públicas, que você pode escolher ao criar uma função.

----Fim

Exclusão de uma dependência

Passo 1 Entre ao console do FunctionGraph, e escolha **Functions** > **Dependencies** no painel de navegação.

Passo 2 Click **Delete** in the **Operation** column of the target dependency.

Passo 3 Clique em **Sim**.

NOTA

Dependências referenciadas por funções não podem ser excluídas.

----Fim

Bibliotecas dependentes

Supported Dependent Libraries

O FunctionGraph oferece suporte a bibliotecas padrão e de terceiros.

- Bibliotecas padrão

  Ao usar bibliotecas padrão, você pode importá-las para seu código ou pacote embutido e enviá-las para FunctionGraph.

- Bibliotecas não-padrão suportadas

  O FunctionGraph fornece componentes de terceiros integrados listados em **Tabela 9-1** e **Tabela 9-2**. Você pode importar essas bibliotecas para o código embutido da mesma maneira que importa bibliotecas padrão.

Tabela 9-1 Componentes de terceiros integrados com o runtime do Node.js

Nome	Uso	Versão
q	Encapsulamento assíncrono do método	1.5.1
co	Controle assíncrono do processo	4.6.0
lodash	Biblioteca de ferramentas e métodos comuns	4.17.10
esdk-obs-nodejs	SDK do OBS	2.1.5

Nome	Uso	Versão
express	Framework simplificado de desenvolvimento de aplicativos baseado na web	4.16.4
fgs-express	Fornece uma estrutura de aplicativos Node.js para FunctionGraph e API Gateway para executar aplicativos sem servidor e as API de REST. Este componente fornece um exemplo do uso da estrutura Express para criar aplicativos ou serviços da web sem servidor e as API de RESTful.	1.0.1
request	Simplifica a invocação HTTP e suporta HTTPS e redirecionamento.	2.88.0

Tabela 9-2 Bibliotecas não-padrão suportadas pelo tempo de execução do Python

Módulo	Uso	Versão
dateutil	Processamento de data e hora	2.6.0
requests	Biblioteca HTTP	2.7.0
httplib2	Cliente HTTP	0.10.3
numpy	Computação matemática	1.13.1
redis	Cliente Redis	2.10.5
obsclient	Cliente de OBS	-
smnsdk	Acesso a SMN	1.0.1

- Outras bibliotecas de terceiros (O FunctionGraph não tem bibliotecas internas de terceiros não padrão, exceto as listadas na tabela anterior.)

  Para usar funções de bibliotecas de terceiros, empacote essas bibliotecas e carregue-as em um intervalo OBS especificado e cole o URL do link OBS dessas bibliotecas ao criar uma função.

Importing Dependent Libraries

O código para processamento de imagens é o seguinte:

```
# -*- coding: utf-8 -*-
from PIL import Image, ImageEnhance

from com.obs.client.obs_client import ObsClient
```



```
import sys
import os

current_file_path = os.path.dirname(os.path.realpath(__file__))
# append current path to search paths, so that we can import some third
party libraries.
sys.path.append(current_file_path)
region = 'your region'
obs_server = 'obs.xxxxxxcloud.com'
def newObsClient(context):
    ak = context.getAccessKey()
    sk = context.getSecretKey()
    return ObsClient(access_key_id=ak, secret_access_key=sk,
server=obs_server,
                     path_style=True, region=region, ssl_verify=False,
max_retry_count=5, timeout=20)
def downloadFile(obsClient, bucket, objName, localFile):
    resp = obsClient.getObject(bucket, objName, localFile)
    if resp.status < 300:
        print 'download file', file, 'succeed'
    else:
        print('download failed, errorCode: %s, errorMessage: %s,
requestId: %s' % resp.errorCode, resp.errorMessage,
              resp.requestId)
def uploadFileToObs(client, bucket, objName, file):
    resp = client.putFile(bucket, objName, file)
    if resp.status < 300:
        print 'upload file', file, 'succeed'
    else:
        print('upload failed, errorCode: %s, errorMessage: %s,
requestId: %s' % resp.errorCode, resp.errorMessage,
              resp.requestId)
def getObjInfoFromObsEvent(event):
    s3 = event['Records'][0]['s3']
    eventName = event['Records'][0]['eventName']
    bucket = s3['bucket']['name']
    objName = s3['object']['key']
    print "*** obsEventName: %s, srcBucketName: %s, objName: %s",
eventName, bucket, objName
    return bucket, objName
def set_opacity(im, opacity):
    """Set the transparency."""
    if im.mode != "RGBA":
        im = im.convert('RGBA')
    else:
        im = im.copy()
    alpha = im.split()[3]
    alpha = ImageEnhance.Brightness(alpha).enhance(opacity)
    im.putalpha(alpha)
    return im
def watermark(im, mark, opacity=0.6):
    """Add a watermark."""
    try:
        if opacity < 1:
            mark = set_opacity(mark, opacity)
        if im.mode != 'RGBA':
            im = im.convert('RGBA')
        if im.size[0] < mark.size[0] or im.size[1] < mark.size[1]:
            print "The mark image size is larger size than original
image file."
            return False
        x = (im.size[0] - mark.size[0]) / 2
```



```
        y = (im.size[1] - mark.size[1]) / 2
        layer = Image.new('RGBA', im.size, )
        layer.paste(mark, (x, y))
        return Image.composite(layer, im, layer)
    except Exception as e:
        print ">>>>>>>>>>> WaterMark EXCEPTION:  " + str(e)
        return False
def watermark_image(localFile, fileName):
    im = Image.open(localFile)
    watermark_image_path = os.path.join(current_file_path,
"watermark.png")
    mark = Image.open(watermark_image_path)
    out = watermark(im, mark)
    print "**********finish water mark"
    name = fileName.split('.')
    outFileName = name[0] + '-watermark.' + name[1]
    outFilePath = "/tmp/" + outFileName
    if out:
        out = out.convert('RGB')
        out.save(outFilePath)
    else:
        print "Sorry, Save watermarked file Failed."
    return outFileName, outFilePath
def handler(event, context):
    srcBucket, srcObjName = getObjInfoFromObsEvent(event)
    outputBucket = context.getUserData('obs_output_bucket')
    client = newObsClient(context)
    # download file uploaded by user from obs
    localFile = "/tmp/" + srcObjName
    downloadFile(client, srcBucket, srcObjName, localFile)
    outFileName, outFile = watermark_image(localFile, srcObjName)
    # Upload converted files to a new OBS bucket.
    uploadFileToObs(client, outputBucket, outFileName, outFile)
    return 'OK'
```

Para bibliotecas padrão e bibliotecas não-padrão suportadas, você pode usá-las diretamente em sua função.

Para bibliotecas de terceiros não padrão que não são fornecidas pelo FunctionGraph você pode usá-las executando as seguintes etapas:

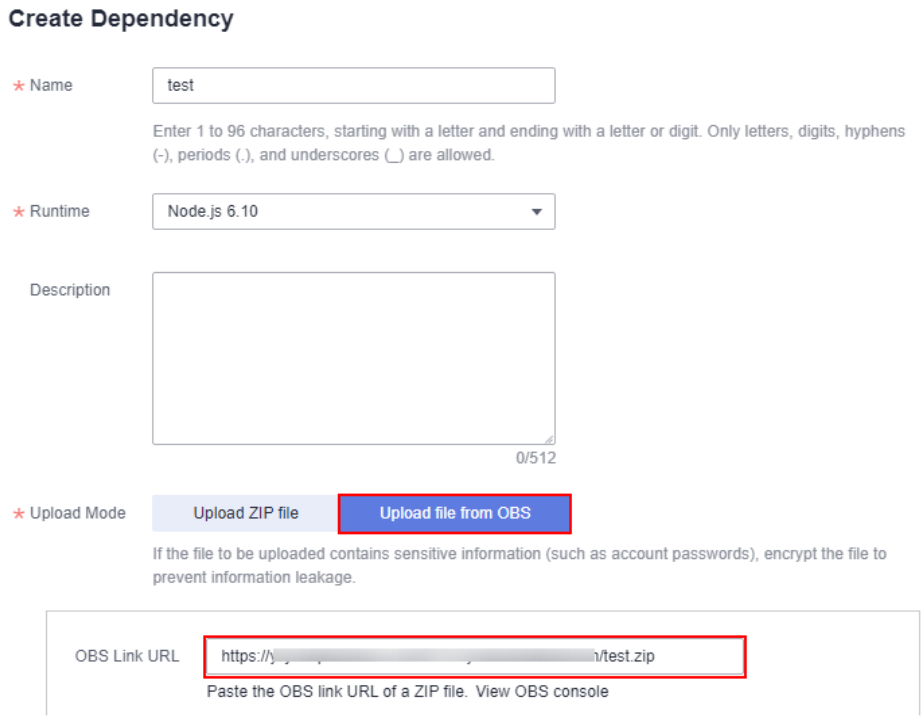
1. Empacote as bibliotecas dependentes em um arquivo ZIP, faça o upload do arquivo ZIP em um bucket do OBS e obtenha o URL do link do OBS.
2. Entre ao console do FunctionGraph, e escolha **Functions** > **Dependencies** no painel de navegação.
3. Clique em **Create Dependency**.
4. Defina o nome da dependência e o tempo de execução, especifique a URL do link do OBS e clique em **OK**.

   NOTA

   Para obter detalhes sobre como obter a URL do link OBS, consulte **Acesso de um objeto por usar seu URL**. (A figura a seguir é apenas para referência. Use o URL real do pacote de arquivos carregado.)

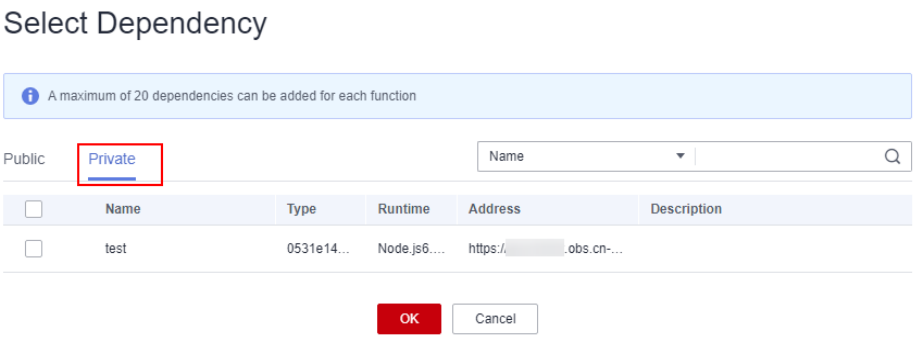
   Figura 9-1 Obtenção do URL do link do OBS

Figura 9-2 Configuração da dependência



5. Na página Detalhes da função, clique na guia **Code**, clique em **Add** na área **Dependencies**, selecione a dependência criada em **4** (consulte **Figura 9-3**), e clique em **OK**.

Figura 9-3 Seleção de uma dependência



6. Clique em **Save**.

ATENÇÃO

Cada pacote de dependência não pode conter um arquivo com o mesmo nome de um arquivo de código. Caso contrário, os dois arquivos podem ser mesclados ou substituídos incorretamente. Por exemplo, se o pacote de dependência **depends.zip** contém um arquivo chamado **index.py**, o manipulador de uma função não pode ser definido como **index.handler**. Caso contrário, um arquivo de código também chamado **index.py** será gerado.

---

10 Gerenciamento de instâncias reservadas

O que é uma instância reservada?

O FunctionGraph fornece instâncias sob demanda e reservadas.

- As instâncias sob demanda são criadas e liberadas pelo FunctionGraph com base no uso real da função. Ao receber solicitações para chamar funções, o FunctionGraph aloca automaticamente recursos de execução para as solicitações.
- As instâncias reservadas podem ser criadas e liberadas por você, conforme necessário. Depois de criar instâncias reservadas para uma função, o FunctionGraph encaminha preferencialmente solicitações para as instâncias reservadas. Se o número de solicitações exceder a capacidade de processamento das instâncias reservadas, o FunctionGraph encaminhará as solicitações excessivas para instâncias sob demanda e alocará automaticamente recursos de execução a essas solicitações.

  Depois que as instâncias reservadas são criadas para uma função, o código, as dependências e o inicializador da função são carregados automaticamente. As instâncias reservadas estão sempre vivas no ambiente de execução, eliminando a influência de cold starts em seus serviços.

  NOTA

  Para usar instâncias reservadas, envie um ticket de serviço para adicionar sua conta à lista de permissões.

Você pode criar instâncias reservadas diretamente ou criá-las na página de configuração da função. As diferenças entre os dois modos são descritas na tabela a seguir.

Tabela 10-1 Diferenças entre os dois modos

Modo	Vantagem	Desvantagem
Criação de instâncias reservadas diretamente	Você pode criar uma instância reservada com apenas alguns cliques.	Apenas um número fixo de instâncias reservadas pode ser criado. As instâncias reservadas podem ser insuficientes durante o horário de pico e ficar ociosas durante o horário fora do horário de pico.

Modo	Vantagem	Desvantagem
Criação de instâncias reservadas na página de configuração da função	Você pode criar diferentes números de instâncias reservadas para diferentes períodos, evitando o desperdício de instâncias reservadas durante os horários de pico e garantindo instâncias reservadas suficientes durante os horários de pico.	O procedimento é complexo.

Criação de Diretamente um Número Fixo de Instâncias Reservadas

Verifique se a função, por exemplo, **Objective-func**, para a qual você deseja criar instâncias reservadas já existe no console do FunctionGraph.

1. Faça login no console do FunctionGraph e escolha **Functions** > **Reserved Instances** no painel de navegação.
2. Clique em **Configure Reserved Instance**.
3. Defina os seguintes parâmetros:

   Tabela 10-2 Informações de instância reservadas

Parâmetro	Descrição
Aplicação	Selecione o aplicativo ao qual a função **Objective-func** pertence.
Função	Selecione **Objective-func**.
Versão	Selecione uma versão da função **Objective-func**.
Quantidade de RI	Informe o número de instâncias reservadas a serem criadas. Para determinar quantas instâncias reservadas devem ser criadas, visualize o número de instâncias usadas na área **Instance Statistics** ou verifique o uso da função **Objective-func**.

4. Clique em **OK**.

   NOTA

   Depois de criar instâncias reservadas para uma função, você só pode alterar a quantidade de instâncias reservadas.

Criação de Instâncias Reservadas na Página Configuração da Função

O número de instâncias de função varia entre diferentes períodos. Para garantir instâncias reservadas suficientes durante os horários de pico e evitar o desperdício de instâncias reservadas durante os horários de pico, você pode criar um temporizador para invocar a função correspondente com um número diferente de instâncias reservadas em períodos diferentes.

Verifique se a função, por exemplo, **Objective-func**, para a qual você deseja criar instâncias reservadas já existe no console do FunctionGraph.

1. Faça login no console do FunctionGraph e escolha **Functions** > **Function List** no painel de navegação.
2. Clique em **Create Function**.
3. Defina as seguintes informações.

Parâmetro	Descrição
Modelo	Selecione **Create from scratch**.
Nome da função	Insira um nome para identificar a função.
Aplicação	Selecione **default**.
Agência	Selecione **Use no agency**.
Descrição	Insira uma descrição para a função. Este parâmetro é opcional.
Tempo de execução	Selecione **Python 2.7**.
Manuseador	Insira **index.handler**.
Modo de entrada de código	Ao criar uma função, selecione **Default code**. Na página de guia **Configuration**, selecione **Edit code inline** e digite o seguinte código:

```
# -*- coding:utf-8 -*-
import json
import requests
def handler (event, context):
    domainId = "https://{Endpoint}"
    url = "/v2/{project_id}/fgs/functions/{func-urn}/reservedinstances"
    token = context.getToken()
    requrl = domainId + url
    headerdata = {"Content-Type":"application/json","x-auth-token":token}
    r = requests.put(requrl, data=event["user_event"],
headers=headerdata,verify=False)
    return r.json
```

Substitua os seguintes parâmetros pelos valores reais:

- *Endpoint*: Ponto final da função **Objective-func**. Para obter detalhes, consulte **Regiões e endpoints**.
- *id_do_projeto*: ID do projeto ao qual a função **Objective-func** pertence. Para obter detalhes, consulte **Obtenção de um ID de projeto**.
- *func-urna*: URN da função **Objective-func**.

4. Clique em **Create Now**.
5. Na página de guia **Configuration**, clique em **Create Agency**.
6. Crie uma agência e conceda a ela a permissão Usuário **FunctionGraph User**. Para mais detalhes, veja **Configuração das permissões de agência**.
7. Na página de guia **Configuration**, selecione a agência criada em **6** e clique em **Save**.
8. Na página de guia **Triggers**, clique em **Create Trigger**.
9. Defina as seguintes informações.

Parâmetro	Descrição
Tipo de fatilho	Selecione **Timer**.
Nome do temporizador	Digite um nome para identificar o temporizador.
Regra	Selecione **Cron expression** e insira uma expressão cron conforme necessário.
Ativar gatilho	Por predefinição, esta função está ativada. Mantenha a configuração padrão.
Informações adicionais	Informe o número de instâncias reservadas necessárias em períodos diferentes com base na regra de gatilho.

Figura 10-1 Criação de um gatilho de timer

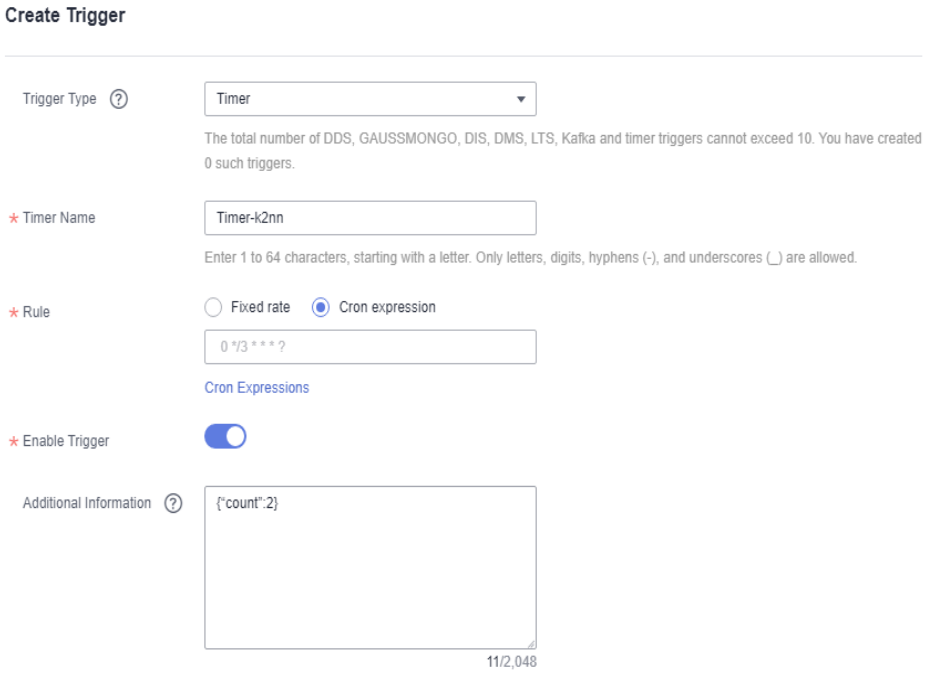


Figura 10-1 mostra um acionador de temporizador que solicita que o FunctionGraph crie duas instâncias reservadas a cada 3 minutos.

10. Clique em **OK**.

    Após a criação do gatilho, diferentes números de instâncias reservadas serão criados para a função **Objective-func** em diferentes períodos com base na regra do gatilho.

11 Aumento de cotas de recursos

Visão geral

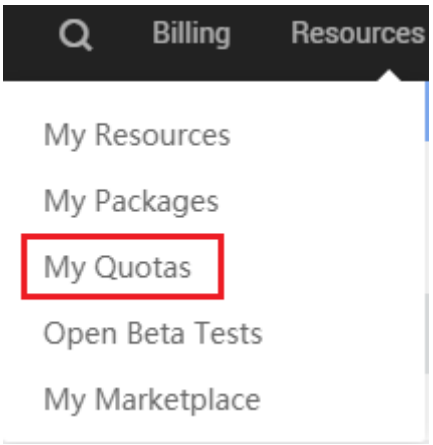
Uma cota é um limite na quantidade ou capacidade de um determinado tipo de recursos de serviço que você pode usar. Por exemplo, o número máximo de discos ECS ou EVS que podem ser criados.

Se uma cota não puder atender às suas necessidades, solicite uma cota mais alta.

Visualização de cotas

1. Efetue login no console de gerenciamento.
2. No canto superior direito da página, escolha **Resources** > **My Quotas**.

   Figura 11-1 Indo para a página **Service Quota**

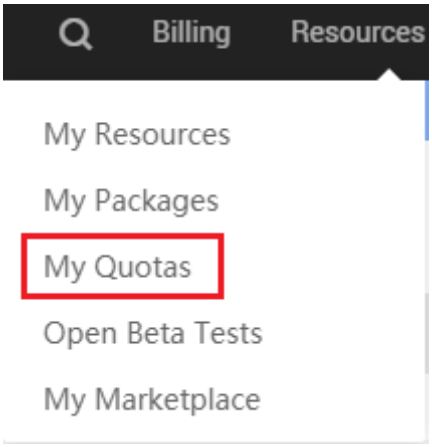
3. Na página **Service Quota**, exiba as cotas usadas e totais de recursos.

   Se uma cota não puder atender às suas necessidades, solicite uma cota mais alta realizando as seguintes operações.

Aumento das cotas

1. Efetue login no console de gerenciamento.
2. No canto superior direito da página, escolha **Resources** > **My Quotas**. A página **Service Quota** é exibida.

Figura 11-2 Indo para a página **Service Quota**



3. Clique em **Increase Quota**.
4. Na página **Create Service Ticket**, defina os parâmetros.
   Na área **Problem Description**, insira a cota necessária e o motivo do ajuste.
5. Leia os contratos e confirme que concorda com eles e, em seguida, clique em **Submit**.

12 Auditoria

12.1 Operações registradas pelo CTS

Tabela 12-1 lista as operações do FunctionGraph que podem ser registradas pelo CTS.

Tabela 12-1 Operações registradas pelo CTS

Operação	Tipo de recurso	Nome do evento
Criação de uma função	Função	CreateFunction
Exclusão de uma função	Função	DeleteFunction
Modificação de informações da função	Função	ModifyFunctionMetadata
Pública de uma versão de função	Versão da função	PublishFunctionVersion
Exclusão de um alias de versão de função	Alias da versão da função	DeleteVersionAlias
Exclusão de um gatilho de função	Gatilho	DeleteTrigger
Criação de um gatilho de função	Gatilho	CreateTrigger
Desativação de um gatilho de função	Gatilho	DisabledTrigger
Ativação um gatilho de função	Gatilho	habilitadoTrigger

12.2 Visualização de registros de auditoria

Para obter detalhes, consulte **Consulte de rastreamentos em tempo real**.

