FunctionGraph

FAQs

Issue	01
Date	2024-11-11

Contents

1 General FAQs

1.1 What Is FunctionGraph?

FunctionGraph allows you to run you code without provisioning or managing servers, while ensuring high availability and scalability. All you need to do is upload your code and set execution conditions, and FunctionGraph will take care of the rest. You pay only for what you use and you are not charged when your code is not running.

1.2 Do I Need to Apply for Any Compute, Storage, or Network Services When Using FunctionGraph?

When using FunctionGraph, you do not need to apply for or pre-configure any computing, storage, or network services, but need to upload and run code in supported runtimes. FunctionGraph provides and manages underlying compute resources, including server CPUs, memory, and networks. It performs configuration and resource maintenance, code deployment, automatic scaling, load balancing, secure upgrade, and resource monitoring.

1.3 Do I Need to Deploy My Code After Programming?

After programming, you only need to package your code into a ZIP file (Java, Node.js, Python, and Go) or JAR file (Java), and upload the file to FunctionGraph for execution.

When creating a ZIP file, place the handler file under the **root** directory to ensure that your code can be run normally after being decompressed.

If you edit code in Go, zip the compiled file, and ensure that the name of the dynamic library file is consistent with the plugin name of the handler. For example, if the name of the dynamic library file is **testplugin.so**, set the handler to **testplugin.Handler**.

1.4 How Do I Obtain a Token?

You can use a token for authentication when calling APIs. To obtain a token, use the standard API of Identity and Access Management (IAM).

- Run the following command to obtain the token in the **CN South-Guangzhou** region:

```
curl -k -i  -X POST   https://iam.cn-south-1.myhuaweicloud.com/v3/auth/tokens  -H
'Content-Type: application/json'  -d '{
 "auth": {
   "identity": {
    "methods": [
      "password"
    ],
    "password": {
      "user": {
       "name": "HUAWEI CLOUD account",
       "password": "Login password",
       "domain": {
         "name": "HUAWEI CLOUD account"
       }
      }
    }
   },
   "scope": {
    "project": {
      "name": "cn-south-1"
    }
   }
 }
}'
```

- Run the following command to obtain the token in the **CN North-Beijing1** region:

```
curl -k -i  -X POST   https://iam.cn-north-1.myhuaweicloud.com/v3/auth/tokens  -H
'Content-Type: application/json'  -d '{
 "auth": {
   "identity": {
    "methods": [
      "password"
    ],
    "password": {
      "user": {
       "name": "HUAWEI CLOUD account",
       "password": "Login password",
       "domain": {
         "name": "HUAWEI CLOUD account"
       }
      }
    }
   },
   "scope": {
    "project": {
      "name": "cn-north-1"
    }
   }
 }
}'
```

📖 **NOTE**

The value of **X-Subject-Token** in the response header is the token. A token obtained in one region can only be used to call FunctionGraph in this region.

For details, see **Obtaining a User Token**.

1.5 What Runtimes Does FunctionGraph Support?

Table 1-1 lists the runtimes supported by FunctionGraph.

Table 1-1 Supported runtimes and versions

Runtime	Version
Python	2.7, 3.6, 3.9, 3.10
Node.js	6.10, 8.10, 10.16, 12.13, 14.18, 16.17, 18.15
Java	8, 11
Go	1.x
C#.NET Core	2.1, 3.1
PHP	7.3

1.6 How Much Disk Space Is Allocated to Each FunctionGraph Function?

Each FunctionGraph function is allocated 512 MB ephemeral disk space. You can upload deployment packages up to 10 GB in size. For more information, see **Notes and Constraints**.

1.7 Does FunctionGraph Support Function Versioning?

Yes. For details, see **Managing Versions**.

1.8 How Does a Function Read or Write Files?

Background

A function can read files in the code directory. The working directory of a function is the upper-level directory of the handler file. Assume that you have uploaded a folder named **backend**. To read its **test.conf** file in the same level of directory as the handler file, use relative path **code/backend/test.conf** or use a full path (that is, the value of the **RUNTIME_CODE_ROOT** environment variable). To write a file (for example, to create or download a file), go to the **/tmp** directory or use the file system mounting feature provided by FunctionGraph.

> ☐ NOTE
>
> - If containers are reclaimed, file read/written content will become invalid.
> - Currently, FunctionGraph does not support instance persistence.

Typical Scenarios

- Download files stored in Object Storage Service (OBS) to the **/tmp** directory for processing.
- To store function execution data in OBS, create a file in the **/tmp** directory, write the data into the file, and then upload the file to OBS.

1.9 How Do I Set a Proxy When Using CLI?

Question

When using CLI to upload a ZIP code package, how do I set a proxy server and identity information to complete authentication through the proxy gateway on the internal network?

Answer

Run the following command to set a proxy:

export HTTP_PROXY="http://user:password@proxyIp:proxyPort"

For more information, see https://www.cyberciti.biz/faq/unix-linux-export-variable-http_proxy-with-special-characters/.

1.10 Does FunctionGraph Support Function Extension?

FunctionGraph has integrated non-standard libraries such as redis, http, and obs_client. You can directly use these libraries when developing functions. For more information, see **Developer Guide**.

Alternatively, use your own dependencies. For more information, see **Dependency Management**.

1.11 Which Permissions Are Required for an IAM User to Use FunctionGraph?

If you are prompted insufficient permissions when creating, deleting, modifying, or querying functions and triggers in FunctionGraph as an IAM user, contact the administrator to grant permissions to your user group. If you want to invoke other cloud services, such as OBS, configure an agency with the required permissions. For security purpose, do this by following the principle of least privilege. For details, see **Permissions Management**.

1.12 How Can I Create an ODBC Drive-based Python Dependency Package for Database Query?

For OS-dependent packages (for example, unixODBC), download the source code to compile dependency packages.

1. Log in to your ECS on the ECS console (ensure that the GCC and Make tools have been installed), and run the following command to download the source code package:

   wget *source code path*

   If you downloaded a **.zip** file, run the following command to decompress it:

   unzip xxx/xx.zip

   If you downloaded a **tar.gz** file, run the following command to decompress it:

   tar -zxvf xxx/xx.tar.gz

2. Run the following command to create the **/opt/function/code** directory:

   mkdir /opt/function/code

3. Go to the destination directory and run the following command:

   ./configure --prefix=/opt/function/code --sysconfdir=/opt/function/code;make;make install

4. Go to **/opt/function/code/lib/pkgconfig** and check whether the prefix directory is **/opt/function/code**.

   cd /opt/function/code/lib/pkgconfig

5. Copy all files in **/opt/function/code/lib** to **/opt/function/code**.

   cp -r /opt/function/code/lib/* /opt/function/code

6. Switch to **/opt/function/code** and compress all files in it to a **.zip** package.

   cd /opt/function/code
   zip -r xxx.zip *

1.13 What Is the Quota of FunctionGraph?

For details about the resource quota of FunctionGraph, see **Notes and Constraints**. For details about how to increase the quota, see **How Do I Apply for a Higher Quota?**

1.14 What Chinese Fonts Does FunctionGraph Support?

FunctionGraph supports the following Chinese fonts:

- NotoSansTC-Regular.otf
- NotoSerifTC-Regular.otf
- NotoSansSC-Regular.otf
- NotoSerifSC-Regular.otf

1.15 How Does FunctionGraph Resolve a Private DNS Domain Name?

FunctionGraph cannot directly parse private Huawei Cloud DNS domain names. To parse them, call DNS APIs and perform the following steps.

Resolving a Private DNS Domain Name

Ensure that a VPC and private DNS domain name have been created before performing the following steps:

Step 1 Associate a VPC with the private domain name and add record sets.

Log in to the DNS console and associate a VPC with the private domain name.

Click the domain name, and add a type A record set.

Step 2 Create a function.

Create a function whose runtime is Python 2.7. The following is sample code.

```
# -*- coding:utf-8 -*-
import json
import os
def handler(event, context):
    os.system("curl -iv www.test.com")
```

Step 3 Configure an agency with DNS and VPC permissions for the function.

On the IAM console, create an agency with **DNS ReadOnlyAccess** and **VPC Administrator** permissions for FunctionGraph.

☐ NOTE

You need to configure the permission to read DNS resource data because the function needs to obtain such data when parsing a domain name. Otherwise, the following error message is displayed, indicating that the DNS resource data failed to be obtained.

2020/08/20 10:37:12 GMT+08:00 Start invoke request 'a2f105b4-2e72-4fda-94a5-86d3837e961d', version: latest
[GET /v2/zones/{zone_id}/recordsets] failed, response: {"code":"DNS.1802","message":"Policy doesn't allow dns:recordset:list to be performed."}
2020/08/20 10:37:13 GMT+08:00 Finish invoke request 'a2f105b4-2e72-4fda-94a5-86d3837e961d', duration: 1030.072ms, billing duration: 1100ms, memory used: 77.039MB.

Step 4 Configure the function.

On the details page of the function created in **Step 2**, click the **Configuration** tab and configure the following settings:

1. For **Permissions**, select the agency created in **Step 3**.
2. Enable VPC access and select the created VPC, subnet, and domain name. For details, see **Figure 1-1**.

Figure 1-1 Configure the function
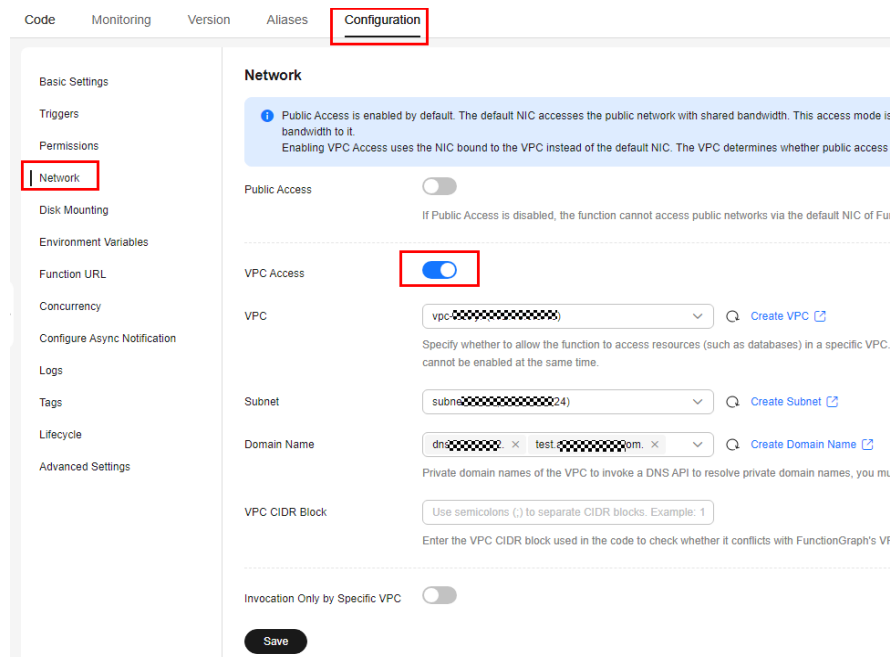
Step 5 Check the execution result.

Then all configured IPv4 domain names can be parsed.

----End

📖 **NOTE**

Changes to the IP addresses corresponding to the VPC domain names you configure will take effect in 10 minutes.

1.16 How Does a Container Image–based Function Resolve a Private DNS Domain Name?

FunctionGraph functions created with a container image cannot directly parse private Domain Name Service (DNS) domain names. However, you can call DNS APIs to achieve this purpose.

Resolving a Private DNS Domain Name

1. Obtain a private domain name and zone ID.

    This procedure uses a domain name with a record set as an example.

    a. Log in to the DNS console.

    b. Obtain a zone ID.

    Click ⚙, and select **Domain Name** in the search box to obtain a zone ID.

    c. Obtain the private domain name corresponding to a recording set.

    Click the domain name to go to the record set list, and select a record set.

2. Compile the resolution logic.

   Debug the API used to **query record sets in a zone**.

   – Set **zone_id** to the zone ID obtained in the preceding step, and click **Debug**. The IP address of the private domain name is displayed in the response body.

   – Switch to the **Sample Code** tab to obtain the complete code. For details about the dependencies, click **View SDK Details**.

1.17 How Do I Use a Domain Name to Access an API Registered with API Gateway (Dedicated)?

The domain name **www.test.com** is used as an example. The procedure is as follows:

Step 1 Log in to the API Gateway console, choose **Dedicated Gateways** in the navigation pane, and click the target gateway name. On the **Gateway Information** page, view **EIP** in the **Inbound Access** area to obtain the IP address of the API gateway.

Step 2 On the DNS console, configure an IPv4 rule for mapping **www.test.com** to an API gateway address.

Step 3 Configure domain name resolution by referring to **How Does FunctionGraph Resolve a Private DNS Domain Name?**. In this way, you can access the API registered with the API gateway by using domain name **www.test.com**.

----End

1.18 What Are the Common Application Scenarios of FunctionGraph?

1. Web applications: mini programs, web pages/apps, chatbots, and Backends for Frontends (BFF).

2. Event-driven applications: file processing, image processing, live video streaming/transcoding, real-time data stream processing, and IoT rule/event processing.

3. AI applications: third-party service integration, AI inference, and license plate recognition.

For details, see **Application Scenarios**.

1.19 Why Can't the API Gateway Domain Name Bound to a Service Be Resolved During Function Invocation?

Currently, FunctionGraph resolves only Huawei Cloud DNS domain names and POD domain names.

1.20 Does FunctionGraph Support Synchronous Transmission at the Maximum Intranet Bandwidth?

Not currently.

1.21 What If the VPC Quota Is Used Up?

A tenant can create up to 4 VPCs. To create more VPCs, **submit a service ticket**.

1.22 How Can I Print Info, Error, or Warn Logs?

Take Java as an example. You can use **this demo** to print logs.

1.23 Can I Set the Domain Name of an API to My Own Domain Name?

Yes. The procedure is as follows:

Step 1   Log in to the APIG console and bind a domain name by referring to **Binding a Domain Name**.

Step 2   On the **Domain Names** tab page of the created API group, click **Bind Independent Domain Name**. For example, set **xxxx.apig.x** to **test.com/user/get**.

----End

1.24 Can I Change the Runtime?

No. Once a function is created, its runtime cannot be changed.

1.25 Can I Change a Function's Name?

No. A function's name cannot be changed once the function is created.

1.26 Why Is Message "failed to mount exist system path" Displayed?

When you see this message, mount the file to a new path.

User ID/user group ID: Can be any number except 1000. The value **–1** will be automatically converted to **1003**. The two IDs control the directory permissions for accessing a remote file system.

File system/ECS name: Name of the file system or ECS to create. Ensure that you have specified a VPC and agency that you have been authorized to access.

Shared directory: To configure a remote shared directory for the mounted ECS, see **Creating an NFS Shared Directory on an ECS**.

Access path: Location where the file system is to be mounted in the function. Set a new two-level directory that starts with **/mnt**. For example, **/mnt/test**.

1.27 How Do I Obtain Uploaded Files?

Take Python as an example. If you use **os.getcwd()** to query the current directory, the directory will be **/opt/function**. However, code has actually been uploaded to **/opt/function/code**.

You can use either of the following methods to obtain uploaded files:

1. Run the **cd** command to switch to **/opt/function/code**.
2. Access the full path (value of the **RUNTIME_CODE_ROOT** environment variable).

   📖 **NOTE**

   You can obtain uploaded files by referring to the preceding methods when other languages are used.

1.28 Why Can't I Receive Responses for Synchronous Invocation?

If the E2E function execution latency exceeds 90s, asynchronous invocation is recommended. If synchronous invocation is used, no responses can be received after 90s due to gateway restrictions.

1.29 What Should I Do If the os.system("command &") Execution Logs Are Not Collected?

Do not use **os.system("command &")**. The background command output will not be collected. To obtain the command output, use **subprocess.Popen** instead.

1.30 Which Directories Can Be Accessed When a Custom Runtime Is Used?

By default, only the **/tmp** directory can be accessed, for example, for creating or downloading files.

1.31 Which Minor Versions of Python 3.6 and 3.9 Are Supported?

3.6.8 and 3.9.2.

1.32 Which Actions Can Be Used Instead of a VPC Administrator Agency for VPC Access?

The actions listed in **Table 1-2** can be used.

Table 1-2 Actions

Permission	Action
Deleting a port	vpc:ports:delete
Querying a port	vpc:ports:get
Creating a port	vpc:ports:create
Querying a VPC	vpc:vpcs:get
Querying a subnet	vpc:subnets:get

1.33 What Are the Possible Causes for Function Timeout?

- The code logic timed out. In this case, optimize the code or increase the timeout.
- The network timed out. To fix this issue, increase the timeout.
- It took a long time to load Java classes during cold start. In this case, increase the timeout or memory.

1.34 How Do I Obtain the Code of a Function?

1. Log in to the FunctionGraph console, and click the name of the target function to go to the details page. Choose **Operation** > **Export function** in the upper right, and click **Export Code**.
2. Alternatively, call the function export API.

1.35 Do You Have Sample Code for Initializers?

Yes. See the following examples:

- Node.js (**Initializer introduction**)

```
exports.initializer = function(context, callback) {
    callback(null, '');
    };
```

- Python (**Initializer introduction**)
```
def my_initializer(context):
    print("hello world!")
```
- Java (**Initializer introduction**)
```
public void my_initializer(Context context)
{
RuntimeLogger log = context.getLogger();
log.log(String.format("ak:%s", context.getAccessKey()));
}
```
- PHP (**Initializer introduction**)
```
<?php
Function my_initializer($context) {
    echo 'hello world' . PHP_EOL;
    }
?>
```

1.36 How Do I Enable Structured Log Query?

Scenario

To check the status of asynchronous invocation requests, view the records by choosing **Configuration** > **Configure Async Notification** on the function details page, as shown in **Figure 1-2**.

Figure 1-2 Asynchronous invocation records
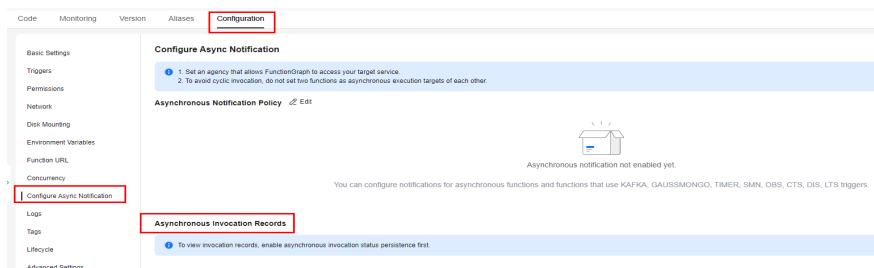
Prerequisites

You have enabled asynchronous invocation status persistence.

Procedure

Step 1 Contact customer service to add your account to the whitelist of this feature.

Step 2 On the **Configure Async Notification** page, click **Enable LTS**, as shown in **Figure 1-3**.

Figure 1-3 Enabling LTS
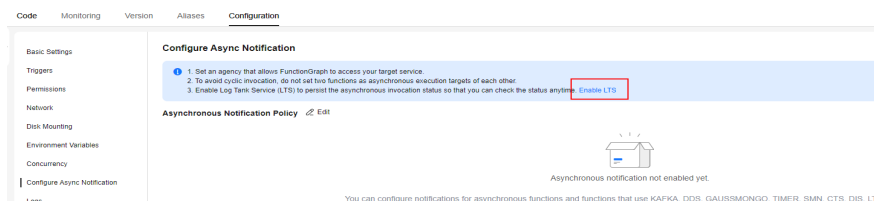
Step 3 Click **Edit** next to **Asynchronous Notification Policy**, and enable **Asynchronous Invocation Status Persistence**, as shown in and **Figure 1-5Figure 1-4**.

Figure 1-4 Configuring asynchronous notification policy
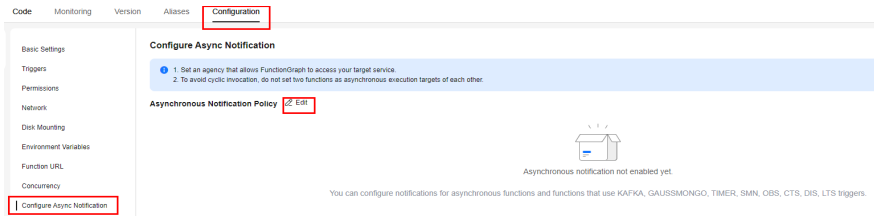
Figure 1-5 Enabling asynchronous invocation status persistence
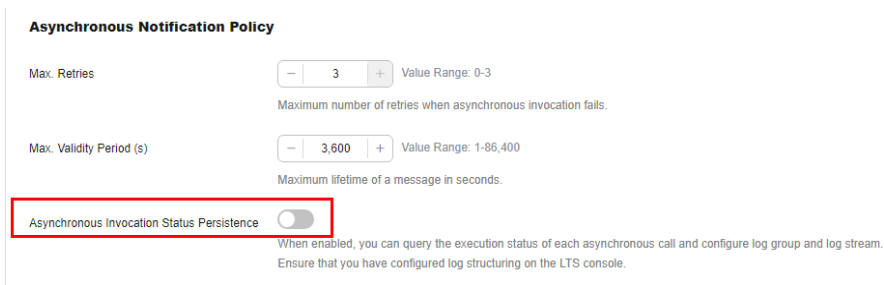
Step 4 Configure structured query on the LTS console.

1. On the function details page, view the log group and log stream. Press **F12**, choose **Network**, enter filter **async-status-log-detail**, and obtain the log group ID and log stream ID, as shown in **Figure 1-6**.

   Figure 1-6 Obtaining log group ID and log stream ID
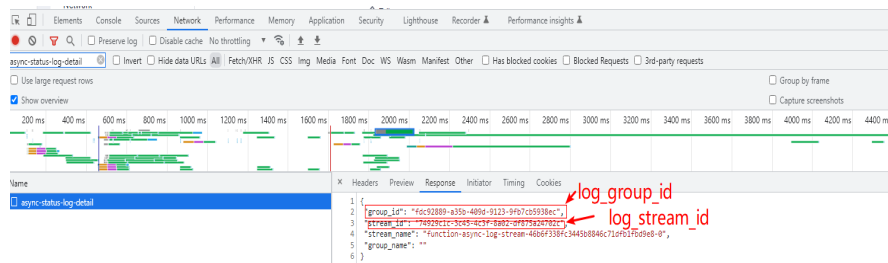
2. On the LTS console, locate the log group and log stream by their IDs, as shown in **Figure 1-7**.

   Figure 1-7 Viewing log stream
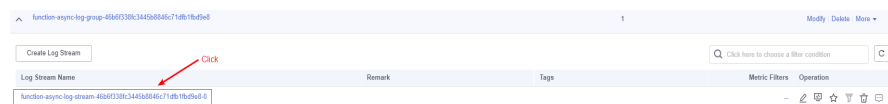
Figure 1-8 Viewing log stream
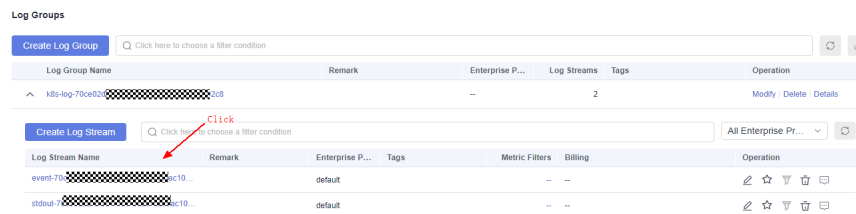
3. On the log stream details page, click the gear icon in the upper right, as shown in **Figure 1-9**.

Figure 1-9 Clicking the gear icon
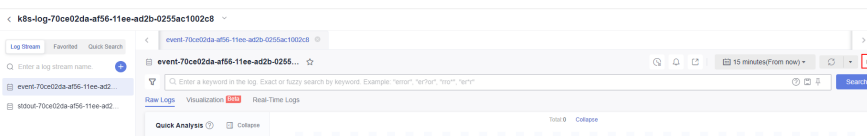
4. Configure log structuring, as shown in **Figure 1-10**.

Figure 1-10 Configuring log structuring
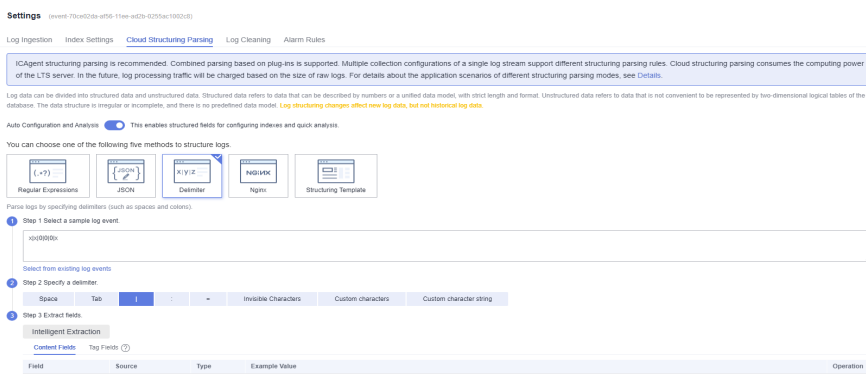
5. Click **Intelligent Extraction**, as shown in **Figure 1-11**.

Figure 1-11 Intelligent Extraction
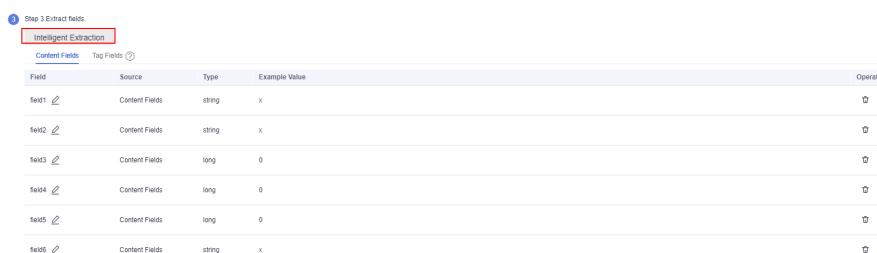
6. Click to modify the field definition as follows:

   a. Change **field1** to **function_urn** and its type to **string**.

   b. Change **field2** to **request_id** and its type to **string**.

   c. Change **field3** to **seq_status** and its type to **long**.

   d. Change **field4** to **operation_timestamp** and its type to **long**.

e. Change **field5** to **error_code** and its type to **long**.

f. Change **field6** to **error_message** and its type to **string**.

Enable **Quick Analysis**, as shown in **Figure 1-12**.

Figure 1-12 Enabling quick analysis
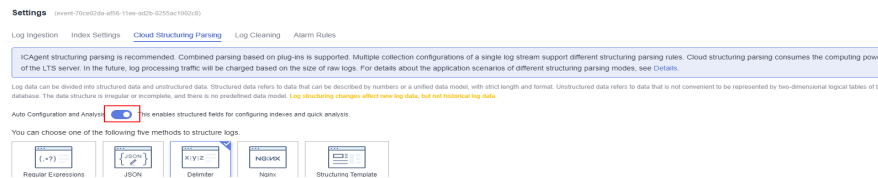
7. Click **Save**. **Figure 1-13** shows the configuration.

Figure 1-13 Saved configuration
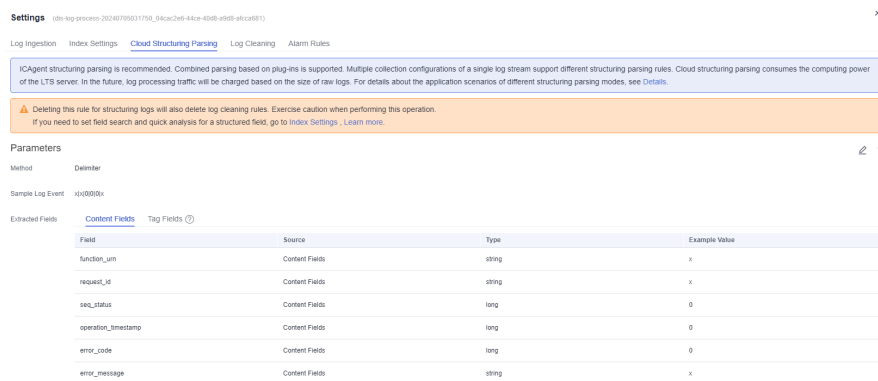
----End

1.37 Can I Enable a Listening Port in a Function to Receive External TCP Requests via EIP?

No. FunctionGraph does not support this feature currently. Functions are about serverless computing, and compute resources are allocated while functions are running. Customizing a listening port is not suitable.

1.38 Does FunctionGraph Support Domain Name Resolution?

No.

To configure custom domain names registered on Domain Name Service (DNS), first convert them to IP addresses in your function.

Call the DNS API to resolve domain name IP addresses, and use these IP addresses to access the corresponding services.

1.39 How Do I Obtain the Source IP Address of an HTTP Request Initiated by a Function?

Public Access

- **VPC Access** disabled for the function

  The SNAT address of FunctionGraph is used to access the public network. It is **fixed**. To obtain it, contact technical support.

  Figure 1-14 VPC access disabled
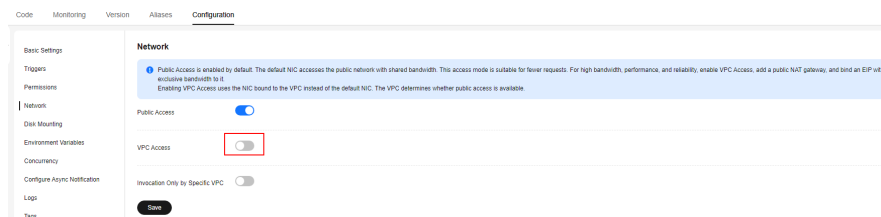
- **VPC Access** enabled for the function (For details about the VPC configuration, see **Configuring Networks**)

  Figure 1-15 VPC access enabled
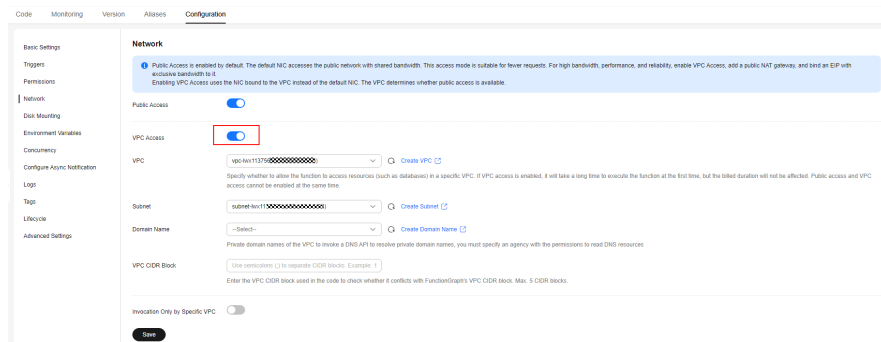
  The SNAT address configured in the user VPC is used to access the public network. It is **fixed**. To obtain the public IP address, perform the following steps:

  a. Log in to the NAT Gateway console, click ⊙ in the upper left corner, and select a region.

  b. In the navigation pane on the left, choose **NAT Gateway** > **Public NAT Gateway**. In the list on the right, click the target gateway name.

  c. On the **SNAT Rules** tab, obtain the public IP address in the rule list.

  Figure 1-16 Obtaining the public IP address

Intra-VPC Access

When **VPC Access** is enabled, functions can access resources in a VPC. (For details about the VPC configuration, see **Configuring Networks**)

The user VPC address mounted in the PAT is used to access resources in the VPC. The address changes dynamically. To view the private IP address, perform the following steps:

1. Log in to the VPC console, click ⊙ in the upper left corner, and select a region.

2. In the navigation pane on the left, choose **Virtual Private Cloud** > **Network Interfaces**. On the displayed page on the right, obtain the private IP address whose name is **FunctionGraph-Access-VPC-Port** and **Type** is **Extended network interface**. For details about elastic network interfaces, see **Elastic Network Interface**.

Figure 1-17 Obtaining the private IP address
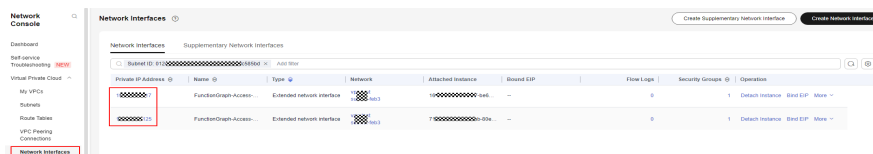
📖 **NOTE**

- When configuring the whitelist or inbound/outbound rules of a security group, ensure that the configured IP address is within the VPC CIDR block. For details about how to create a security group, see **Creating a Security Group**.

- Make sure to test the VPC function to display the **Extended network interface** in the **Type** column.

- The two private IP addresses obtained above are the active and standby addresses. For details, contact technical support.

2 Function Creation FAQs

2.1 Can I Add Threads and Processes in Function Code?

You can create additional threads and processes in your function by using runtime and OS features.

2.2 What Are the Rules for Packaging a Function Project?

In addition to inline code editing, you can create a function by uploading a ZIP or JAR file, or uploading a ZIP file from OBS. For details, see **Packaging Rules** and **Example ZIP Project Packages**.

Packaging Rules

In addition to inline code editing, you can create a function by uploading a local ZIP file or JAR file, or uploading a ZIP file from Object Storage Service (OBS). **Table 2-1** describes the rules for packaging a function project.

Table 2-1 Function project packaging rules

Runtime	JAR File	ZIP File	ZIP File on OBS
Node.js	Not supported.	- If the function project files are saved under the **~/Code/** directory, select and package all files under this directory to ensure that the function handler is under the root directory after the ZIP file is decompressed. - If the function project uses third-party dependencies, package the dependencies into a ZIP file, and import the ZIP file on the function code page. Alternatively, package the third-party dependencies and the function project files together.	Compress project files into a ZIP file and upload it to an OBS bucket.

Runtime	JAR File	ZIP File	ZIP File on OBS
PHP	Not supported.	• If the function project files are saved under the **~/Code/** directory, select and package all files under this directory to ensure that the function handler is under the root directory after the ZIP file is decompressed. • If the function project uses third-party dependencies, package the dependencies into a ZIP file, and import the ZIP file on the function code page. Alternatively, package the third-party dependencies and the function project files together.	Compress project files into a ZIP file and upload it to an OBS bucket.

Runtime	JAR File	ZIP File	ZIP File on OBS
Python 2.7	Not supported.	• If the function project files are saved under the **~/Code/** directory, select and package all files under this directory to ensure that the function handler is under the root directory after the ZIP file is decompressed. • If the function project uses third-party dependencies, package the dependencies into a ZIP file, and import the ZIP file on the function code page. Alternatively, package the third-party dependencies and the function project files together.	Compress project files into a ZIP file and upload it to an OBS bucket.

Runtime	JAR File	ZIP File	ZIP File on OBS
Python 3.6	Not supported.	• If the function project files are saved under the **~/Code/** directory, select and package all files under this directory to ensure that the function handler is under the root directory after the ZIP file is decompressed. • If the function project uses third-party dependencies, package the dependencies into a ZIP file, and import the ZIP file on the function code page. Alternatively, package the third-party dependencies and the function project files together.	Compress project files into a ZIP file and upload it to an OBS bucket.
Java 8	If the function does not reference third-party components, compile only the function project files into a JAR file.	If the function references third-party components, compile the function project files into a JAR file, and compress all third-party components and the function JAR file into a ZIP file.	Compress project files into a ZIP file and upload it to an OBS bucket.

Runtime	JAR File	ZIP File	ZIP File on OBS
Go 1.x	Not supported.	Zip the compiled file and ensure that the name of the binary file is consistent with that of the handler. For example, if the name of the binary file is **Handler**, set the name of the handler to **Handler**.	Compress project files into a ZIP file and upload it to an OBS bucket.
C#	Not supported.	Compress project files into a ZIP file. The ZIP file must contain the following files: *Project_name*.**deps.json**, *Project_name*.**dll**, *Project_name*.**runtimeconfig.json**, *Project_name*.**pdb**, and **HC.Serverless.Function.Common.dll**.	Compress project files into a ZIP file and upload it to an OBS bucket.
Custom	Not supported.	Compress project files into a ZIP file. The ZIP file must contain a bootstrap file.	Compress project files into a ZIP file and upload it to an OBS bucket.
Cangjie	Not supported.	Zip the compiled file and ensure that the name of the binary file is consistent with that of the handler. For example, if the name of the binary file is **libuser_func_test_success.so**, set the name of the handler to **libuser_func_test_success.so**.	Compress project files into a ZIP file and upload it to an OBS bucket.

Example ZIP Project Packages

- Example directory of a Nods.js project package

```
Example.zip                 Example project package
|--- lib                    Service file directory
```

```
|--- node_modules                NPM third-party component directory
|--- index.js                    .js handler file (mandatory)
|--- package.json                NPM project management file
```

- Example directory of a PHP project package
```
Example.zip                      Example project package
|--- ext                         Extension library directory
|--- pear                        PHP extension and application repository
|--- index.php                   PHP handler file
```

- Example directory of a Python project package
```
Example.zip                      Example project package
|--- com                         Service file directory
|--- PLI                         Third-party dependency PLI directory
|--- index.py                    .py handler file (mandatory)
|--- watermark.py                .py file for image watermarking
|--- watermark.png               Watermarked image
```

- Example directory of a Java project package
```
Example.zip                      Example project package
|--- obstest.jar                 Service function JAR file
|--- esdk-obs-java-3.20.2.jar    Third-party dependency JAR file
|--- jackson-core-2.10.0.jar     Third-party dependency JAR file
|--- jackson-databind-2.10.0.jar Third-party dependency JAR file
|--- log4j-api-2.12.0.jar        Third-party dependency JAR file
|--- log4j-core-2.12.0.jar       Third-party dependency JAR file
|--- okhttp-3.14.2.jar           Third-party dependency JAR file
|--- okio-1.17.2.jar             Third-party dependency JAR file
```

- Example directory of a Go project package
```
Example.zip                      Example project package
|--- testplugin.so               Service function package
```

- Example directory of a C# project package
```
Example.zip                          Example project package
|--- fssExampleCsharp2.0.deps.json        File generated after project compilation
|--- fssExampleCsharp2.0.dll              File generated after project compilation
|--- fssExampleCsharp2.0.pdb              File generated after project compilation
|--- fssExampleCsharp2.0.runtimeconfig.json   File generated after project compilation
|--- Handler                      Help file, which can be directly used
|--- HC.Serverless.Function.Common.dll    .dll file provided by FunctionGraph
```

- Example directory of a Cangjie project package
```
fss_example_cangjie.zip                   Example project package
|--- libuser_func_test_success.so         Service function package
```

- Custom
```
Example.zip                      Example project package
|--- bootstrap                   Executable boot file
```

2.3 How Does FunctionGraph Isolate Code?

Each FunctionGraph function runs in its own environment and has its own resources and file system.

2.4 How Do I Create the Bootstrap File for an HTTP Function?

To create an HTTP function, create a bootstrap file. For details, see **Creating a Bootstrap File**.

3 Trigger Management FAQs

3.1 What Events Can Trigger a FunctionGraph Function?

For details, see **Supported Event Sources**.

3.2 What If Error Code 500 Is Reported When Functions that Use APIG Triggers Return Strings?

Ensure that the function response for an invocation by API Gateway has been encapsulated and contains **body(String)**, **statusCode(int)**, **headers(Map)**, and **isBase64Encoded(boolean)**.

The following is an example response returned by a Node.js function that uses an APIG trigger:

```
exports.handler = function (event, context, callback) {
    const response = {
        'statusCode': 200,
        'isBase64Encoded': false,
        'headers': {
            "Content-type": "application/json"
        },
        'body': 'Hello, FunctionGraph with APIG',
    }
    callback(null, response);
}
```

The following is an example response returned by a Java function that uses an APIG trigger:

```
import java.util.Map;

public HttpTriggerResponse index(String event, Context context){
    String body = "<html><title>FunctionStage</title>"
        + "<h1>This is a simple APIG trigger test</h1><br>"
        + "<h2>This is a simple APIG trigger test</h2><br>"
        + "<h3>This is a simple APIG trigger test</h3>"
```

```
            + "</html>";
        int code = 200;
        boolean isBase64 = false;
        Map<String, String> headers = new HashMap<String, String>();
        headers.put("Content-Type", "text/html; charset=utf-8");
        return new HttpTriggerResponse(body, headers, code, isBase64);
    }


class HttpTriggerResponse {
    private String body;
    private Map<String, String> headers;
    private int statusCode;
    private boolean isBase64Encoded;
    public HttpTriggerResponse(String body, Map<String,String> headers, int statusCode,
boolean isBase64Encoded){
        this.body = body;
        this.headers = headers;
        this.statusCode = statusCode;
        this.isBase64Encoded = isBase64Encoded;
    }
}
```

3.3 What Do LATEST and TRIM_HORIZON Mean in DIS Trigger Configuration?

Cursors **LATEST** and **TRIM_HORIZON** specify the start points for reading data in Data Ingestion Service (DIS) streams.

- **TRIM_HORIZON**: Data is read from the earliest valid record stored in the partition.

  For example, a tenant used a DIS stream to upload three pieces of data A1, A2, and A3. Assuming that A1 expires but A2 and A3 are still valid after a period of time, if the tenant specifies **TRIM_HORIZON** for downloading data, only A2 and A3 can be downloaded.

- **LATEST**: Data is read from the latest record in the partition. This option ensures that the most recent data in the partition is read.

  For details, see **Obtaining a Cursor**.

3.4 How Do I Use an APIG Trigger to Invoke a Function?

For details, see **Using an APIG (Dedicated) Trigger**.

3.5 How Does a Function Obtain the Request Path or Parameters When Using an APIG Trigger?

By default, the request path or parameters are included in **event**. A function invokes APIG using its event template. You can obtain the request path or parameters from the function execution result.

Example:
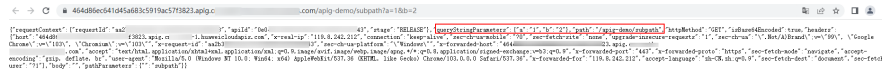
```
{ requestContext:
   { requestId: ▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨ ,
     apiId: ▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨ ,
     stage: 'RELEASE' },
  queryStringParameters: { a: '1', b: '2' },
  path: '/apig-demo/subpath' ,
  httpMethod: 'GET',
  isBase64Encoded: true,
```

- **queryStringParameters**: parameters following the question mark (?), separated with ampersands (&). These parameters are added in the URL of a GET request, and will be transferred in the URL string format when a GET request is initiated.

- **path**: API URL.

You can call an API using its request path. Example: **https://464d86ec641d45a683c5919ac57f3823.apig.projectID.huaweicloudapis.com/apig-demo/subpath**

Alternatively, you can call an API by adding request parameters. Example:

https://464d86ec641d45a683c5919ac57f3823.apig.projectID.huaweicloudapis.com/apig-demo**/subpath?a=1&b=2**

3.6 Can I Configure a Kafka Trigger in a Different Subnet from My Function?

No. You can only create a Kafka trigger in the same VPC subnet as your function.

4 Dependency Management FAQs

4.1 What Is a Dependency?

A dependency is a program package and also an environment required for running a software package. The software package relies on and can only run in the environment.

4.2 When Do I Need a Dependency?

When you install a program or develop code that relies on an environment to run, you need to introduce the dependency.

4.3 What Are the Precautions for Using a Dependency?

- The name of each file in a dependency cannot end with a tilde (~).
- There should be no more than 30,000 files in a dependency.
- You can upload a ZIP dependency file within 10 MB on the function details page. For a larger dependency (max. 300 MB), upload it using OBS.
- If your function uses a large private dependency, increase the timeout by choosing **Configuration** > **Basic Settings** on the function details page.

4.4 What Dependencies Does FunctionGraph Support?

Supported Dependencies

FunctionGraph supports standard libraries and third-party dependencies.

- Standard libraries

  When using standard libraries, you can import them to your inline code, or package and upload them to FunctionGraph.
- Supported non-standard libraries

FunctionGraph provides built-in third-party components, as described in **Table 4-1** and **Table 4-2**. You can import these components to your inline code in the same way as you import standard libraries.

Table 4-1 Third-party components integrated with the Node.js runtime

Name	Description	Version
q	Asynchronous method encapsulation	1.5.1
co	Asynchronous process control	4.6.0
lodash	Common tool and method library	4.17.10
esdk-obs-nodejs	OBS sdk	2.1.5
express	Simplified web-based application development framework	4.16.4
fgs-express	Provides a Node.js application framework for FunctionGraph and APIG to run serverless applications and REST APIs. This component provides an example of using the Express framework to build serverless web applications or services and RESTful APIs.	1.0.1
request	Simplifies HTTP invocation and supports HTTPS and redirection.	2.88.0

Table 4-2 Non-standard libraries supported by the Python runtime

Module	Description	Version
dateutil	Date and time processing	2.6.0
requests	HTTP library	2.7.0
httplib2	httpclient	0.10.3
numpy	Mathematical computation	1.13.1
redis	Redis client	2.10.5

Module	Description	Version
obsclient	OBS client	-
smnsdk	SMN access (public cloud)	1.0.1

- Other third-party libraries

  For other third-party libraries not listed in the preceding tables, package and upload them to an OBS bucket or on the function details page. For details, see **How Do I Create a Dependency on the FunctionGraph Console?** These libraries will then be used in your function code.

4.5 Does FunctionGraph Support Class Libraries?

Yes. FunctionGraph supports both standard libraries and non-standard third-party libraries. For details, see **What Dependencies Does FunctionGraph Support?**

4.6 How Do I Use Third-Party Dependencies on FunctionGraph?

1. Package third-party libraries into a ZIP package by referring to **How Do I Create Function Dependencies?**

2. Create a dependency on the FunctionGraph console by referring to **How Do I Create a Dependency on the FunctionGraph Console?**

3. On the function details page, click the **Code** tab, and add the dependency by referring to **How Do I Add a Dependency to a Function?** Then you can use the dependency in the function code.

4.7 How Do I Create Function Dependencies?

You are advised to create function dependencies in EulerOS. If other OSs are used, an error may occur due to underlying dependent libraries. For example, the dynamic link library cannot be found.

☐ NOTE

If the modules to be installed need dependencies such as .dll, .so, and .a, archive them to a .zip package.

Setting Up the EulerOS Environment

EulerOS is an enterprise-grade Linux OS based on open-source technology. It features high security, scalability, and performance, meeting customers' requirements for IT infrastructure and cloud computing services. **Huawei Cloud EulerOS** is recommended.

1. Buy a EulerOS ECS on Huawei Cloud by referring to **Purchasing and Logging In to a Linux ECS**. On the **Configure Basic Settings** page, select **Public Image**, and select **Huawei Cloud EulerOS** and an image version.

2. Download the EulerOS image, and use virtualization software to set up the EulerOS VM on a local PC.

Creating a Dependency for a Python Function

Ensure that the Python version of the packaging environment is the same as that of the function. For Python 2.7, Python 2.7.12 or later is recommended. For Python 3.6, Python 3.6.3 or later is recommended.

To install the PyMySQL dependency for a Python 2.7 function in the local **/tmp/pymysql** directory, run the following command:

```
pip install PyMySQL --root /tmp/pymysql
```

After the command is successfully executed, go to the **/tmp/pymysql** directory:

```
cd /tmp/pymysql/
```

Go to the **site-packages** directory (generally, **usr/lib64/python2.7/site-packages/**) and then run the following command:

```
zip -rq pymysql.zip *
```

The required dependency is generated.

☐ NOTE

To install the local wheel installation package, run the following command:

```
pip install piexif-1.1.0b0-py2.py3-none-any.whl --root /tmp/piexif
//Replace piexif-1.1.0b0-py2.py3-none-any.whl with the actual installation package name.
```

Creating a Dependency for a Node.js Function

Ensure that the corresponding Node.js version has been installed in the environment.

To install the MySQL dependency for a Node.js 8.10 function, run the following command:

```
npm install mysql --save
```

The **node_modules** folder is generated under the current directory.

● Linux OS

Run the following command to generate a ZIP package.

```
zip -rq mysql-node8.10.zip node_modules
```

The required dependency is generated.

● Windows OS

Compress **node_modules** into a ZIP file.

To install multiple dependencies, create a **package.json** file first. For example, enter the following content into the **package.json** file and then run the following command:

```
{
    "name": "test",
    "version": "1.0.0",
    "dependencies": {
        "redis": "~2.8.0",
```

```
        "mysql": "~2.17.1"
    }
}
npm install --save
```

> **NOTE**
>
> Do not run the **CNPM** command to generate Node.js dependencies.

Compress **node_modules** into a ZIP package. This generates a dependency that contains both MySQL and Redis.

For other Node.js versions, you can create dependencies in the way stated above.

Creating a Dependency for a Java Function

When you compile a function using Java, dependencies need to be compiled locally. For details about how to add dependencies, see **Developing Functions in Java (Using an IDEA Java Project)**.

Creating a Dependency for a PHP Function

EulerOS 2.9.6 is recommended.

By default, Composer and PHP 7.3 have been installed in the environment. Install Protobuf 3.19 using Composer.

Create the **composer.json** file with the following content:

```
{
    "require": {
        "google/protobuf": "^3.19"
    }
}
```

Run the following command:

```
Composer install
```

The **vendor** folder is generated with the **autoload.php**, **composer**, and **google** subfolders in the current directory.

- Linux

  Run the following command to generate a ZIP package.

  ```
  zip –rq vendor.zip vendor
  ```

- Windows

  Compress **vendor** into a ZIP file.

  If multiple dependencies need to be installed, specify them in the **composer.json** file, compress the **vendor** folder into a ZIP file and upload it.

  > **NOTE**
  >
  > To use third-party dependencies downloaded using Composer in PHP project code, load the dependencies through **require "./vendor/autoload.php"**. By default, files decompressed from the uploaded ZIP package are placed in a directory at the same level as the project code.

4.8 How Do I Create a Dependency on the FunctionGraph Console?

1. Log in to the FunctionGraph console, and choose **Functions** > **Dependencies** in the navigation pane.
2. Click **Create Dependency**.
3. Set the following parameters.

Table 4-3 Dependency configuration parameters

Parameter	Description
Name	Dependency name.
Code Entry Mode	Upload a ZIP file directly or through OBS. ● **Upload ZIP file**: Click **Select File** to upload a ZIP file. ● **Upload from OBS**: Specify an OBS link URL. For details about how to obtain the URL, see **Accessing an Object Using Its URL**.
Runtime	Select a runtime.
Description	Description of the dependency. This parameter is optional.

4. Click **OK**. By default, a new dependency is version **1**.
5. Click the dependency name, and view all versions and related information on the displayed page. Each dependency can have multiple versions.
   - To create a dependency version, click **Create Version** in the upper right corner of the page.
   - To view the address of a version, click the version.
   - To delete a version, click the delete icon in the same row.

Figure 4-1 Deleting a dependency version

4.9 How Do I Add a Dependency to a Function?

1. On the function details page, click the **Code** tab, and click **Add** in the **Dependencies** area.
   - **Public**: Public dependencies are provided by FunctionGraph and can be directly added.
   - **Private**: Private dependencies are those you created and uploaded.
2. Click **OK**.

5 Function Execution FAQs

5.1 How Long Does It Take to Execute a FunctionGraph Function?

Within 900s for synchronous execution and 72 hours for asynchronous execution.

The default execution timeout is 3s. You can set the timeout (unit: s) to an integer from 3 to 259,200. If you set the timeout of a function to 3s, it will be terminated after 3s.

5.2 Which Steps Are Included in Function Execution?

Function execution includes two steps:

1. Select an idle instance with required memory.
2. Run specified code.

5.3 How Does FunctionGraph Process Concurrent Requests?

FunctionGraph automatically scales in or out function instances based on the number of requests. If the number of concurrent requests increases, FunctionGraph allocates more function instances to process the requests. If that number decreases, FunctionGraph allocates fewer function instances accordingly.

Number of function instances = Function concurrency/Concurrency per instance

- Function concurrency: the number of requests concurrently executed by a function at a certain time point.
- Concurrency per instance: the maximum number of concurrent requests allowed by a single instance. This is equivalent to the **Max. Requests per Instance** parameter on the **Concurrency** page.

📖 **NOTE**

> For details about related services that may be involved in concurrent requests, see **Supported Event Sources**.

5.4 What If Function Instances Have Not Been Executed for a Long Time?

If a function has not been executed for a period of time, all instances related to the function will be released.

5.5 How Can I Speed Up Initial Access to a Function?

C# and Go support a lower startup speed than other languages due to mechanism issues. You can use the following methods to speed up initial access to a function:

- Allocate more memory to the function.
- Simplify function code, for example, delete unnecessary dependency packages.
- When using C# in non-concurrent scenarios, you can also:

  Create a one-minute timer trigger to ensure that there is at least one active instance.

5.6 How Do I Know the Actual Memory Used for Function Execution?

The returned information about a function contains the maximum memory consumed. For more information, see **SDK APIs** in the *FunctionGraph Developer Guide*. Alternatively, check the memory usage in the execution result.

5.7 Why Is My First Request Slow?

Functions are cold-started. If initialization or a lengthy operation is performed during the first function execution, the first request will be delayed. However, subsequent requests before container deletion will be faster. If there is no request within one minute, the container will be deleted.

5.8 What Do I Do If an Error Occurs When Calling an API?

Rectify the fault by referring to **Error Codes**. If the fault persists, contact technical support.

5.9 How Do I Read the Request Header of a Function?

The first parameter in the function handler contains the request header. You can print the function execution result to obtain required fields.

As shown in the following figure, **event** is the first parameter in the function handler, and **headers** is the request header.

```
index.py ×
1    # -*- coding:utf-8 -*-
2    import json
3    def handler (event, context):
4        body = "<html><title>Functiongraph Demo</title><body><p>Hello, FunctionGraph!
5        print(body)
6        return {
7            "statusCode":200,
8            "body":body,
9            "headers": {
10               "Content-Type": "text/html",
11           },
12           "isBase64Encoded": False
13       }
```

5.10 Can the Synchronous Execution Interface Be Invoked on a Private Network?

Yes, the interface is invoked on a private network by default. For cross-region invocation, enable public network access.

5.11 Why Does a Function Use More Memory Than Estimated and Even Trigger the Out of Memory Alarm?

1. Event parsing and cache consume extra memory during function invocation.
2. After the invocation is complete, reclaimed memory is often put in the internal pool instead of back to the OS, resulting in high memory usage. This is more obvious in the case of high concurrency.

5.12 How Do I Check the Memory Usage When Seeing "runtime memory limit exceeded"?

Check the used memory in the response.

Figure 5-1 Checking the used memory
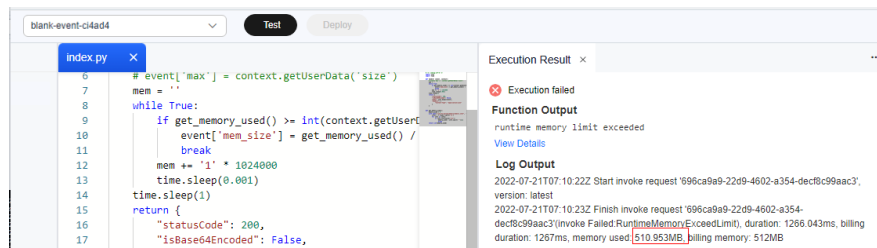
5.13 How Do I Troubleshoot "CrashLoopBackOff"?

The message "CrashLoopBackOff: The application inside the container keeps crashing" is displayed when a custom image execution failure occurs. In this case, perform the following operations:

1. Analyze the causes.

   Figure 5-2 Viewing the execution result

   ```
   function invocation exception, error: CrashLoopBackOff: The application inside the container keeps crashing:
   Traceback (most recent call last):
     File "app.py", line 1, in <module>
       from flask import Flask, request, g
   ModuleNotFoundError: No module named 'flask'
   ```

2. Verify the container image by referring to **Deploying a Function Using a Container Image**.
3. Check whether the image uses the Linux x86 architecture. Currently, only Linux x86 images are supported.

5.14 After I Updated an Image with the Same Name, Reserved Instances Still Use the Old Image. What Can I Do?

Use a non-latest tag to manage image updates, and do not use the same image name.

6 Function Configuration FAQs

6.1 Can I Set Environment Variables When Creating Functions?

Yes. Set variables to dynamically pass settings to your function code and libraries without changing your code. For more information, see **Configuring Environment Variables**.

6.2 Can I Enter Sensitive Information in Environment Variables?

FunctionGraph displays all the information you enter in plain text. Therefore, do not enter insensitive information such as passwords when you define environment variables.

## 6.3 How Do I Use a Function to Invoke a Subfunction?

For details, see the example **Using a Function to Invoke a Subfunction**.

6.4 How Do I Use the Versions and Aliases of an HTTP Function with an APIG Trigger for Gray Upgrade?

1. Create versions V1 and V2 based on **latest**, create an alias named **alias** for V1 with a 100% weight, and create an APIG trigger for **alias**.

Figure 6-1 Publishing versions V1 and V2
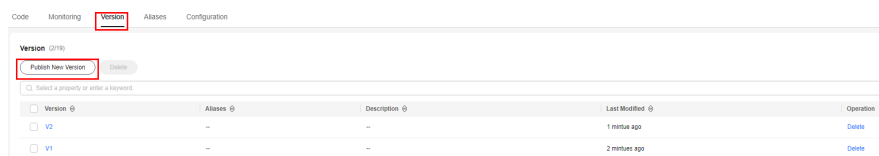
Figure 6-2 Creating alias and binding V1 to it
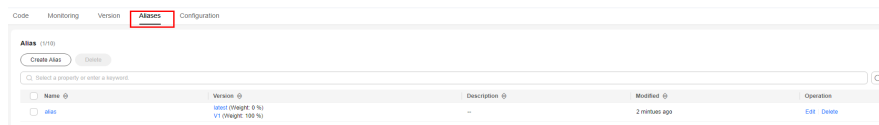
Figure 6-3 Creating a trigger for alias
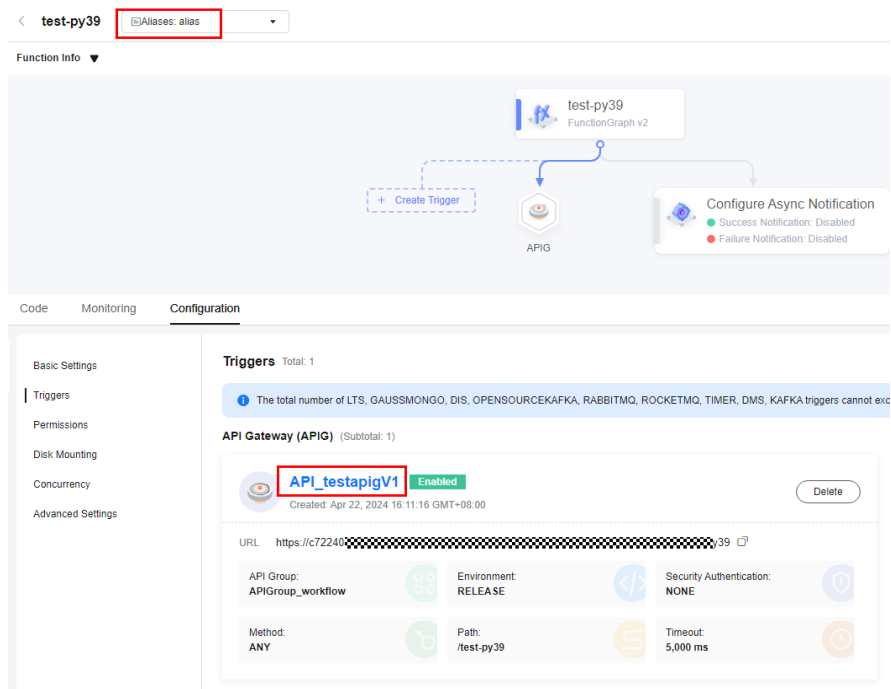
□□ NOTE

You can create APIG triggers for a function alias or version. By default, a trigger name is **API_***{Function name}{Version}*, and the request path is **/***{Function name}-{Version}*.

An APIG trigger URL will only be used either for an alias or its matched version. For example, assume that V1 has been bound to **alias**. If you create a trigger with the default name **API_testapigV1** for V1 and then create another trigger with the same name and URL for **alias**, the **API_testapigV1** trigger will not be displayed in the trigger list of V1.

2. Go back to the **Aliases** tab page of **latest**, click **Edit** in the row of **alias**, and set **Additional Version** to **V2** with a custom weight. This is for gray upgrade from V1 to V2.

   □□ NOTE

   Weights indicate the percentage of received data that will be allocated to corresponding versions. You can set a weight that meets your service requirements.

Figure 6-4 Editing an alias

3. Create version V3, click **Edit** in the row of **alias** on the **Aliases** tab page, and change **Version** to **V2** and **Additional Version** to **V3** with a custom weight. This is for gray upgrade from V2 to V3.

Figure 6-5 Creating V3

Figure 6-6 Editing an alias

NOTICE

After publishing a version based on **latest**, you can publish more versions only after making a configuration or code change on the function.

Aliases bound with triggers cannot be deleted.

---

7 External Resource Access FAQs

7.1 How Does a Function Access the MySQL Database?

Perform the following operations:

1. Check whether the MySQL database is deployed in a VPC.
   - Yes: Configure the same VPC and subnet as the MySQL database for the function by referring to **Configuring VPC Access**.
   - No: See **How Do I Configure External Network Access?**

2. Search for MySQL templates and select the one with the desired runtime, as shown in **Figure 7-1**. Set the parameters as required and click **Create Function**.

   Figure 7-1 Selecting a function template
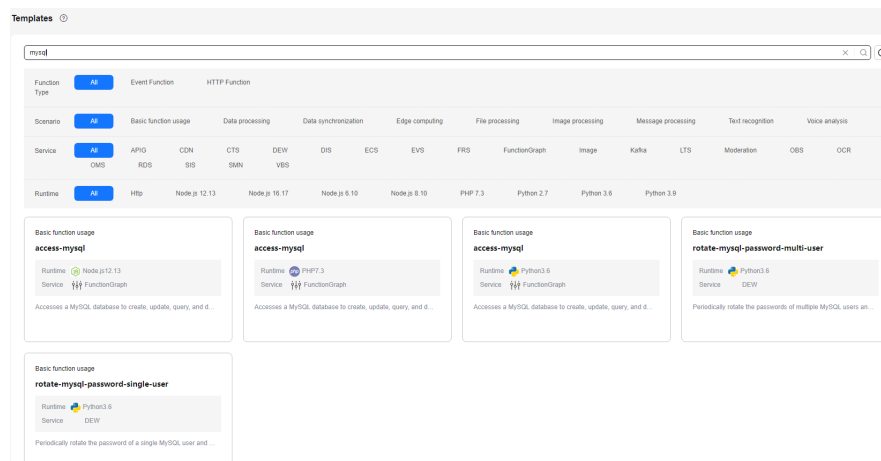
3. After the MySQL function is created, choose **Configuration** > **Environment Variables**, enable encryption as required (see **Figure 7-2**), and click **Save**.

Figure 7-2 Enabling encryption

> **□ NOTE**
>
> If the function needs to access RDS APIs, **create an agency** and grant required permissions.

7.2 How Does a Function Access Redis?

Perform the following operations:

1. Check whether the Redis instance is deployed in a VPC.
   – If the Redis instance is deployed in a VPC, configure the same VPC and subnet as the Redis instance for the function by referring to **Configuring VPC Access**.
   – If the Redis instance is built on a public network, obtain its public IP address.

2. Compile code for connecting a function to the Redis instance.

   FunctionGraph has integrated third-party library **redis-py** in its Python 2.7 and Python 3.6 runtimes. Therefore, you do not need to download any other Redis libraries.

   ```
   # -*- coding:utf-8 -*-
   import redis
   def handler (event, context):
       r = redis.StrictRedis(host="host_ip",password="passwd",port=6379)
       print(str(r.get("hostname")))
       return "^_^"
   ```

   > **□ NOTE**
   >
   > ● If the function fails to access to the Redis instance on a public network, perform the following operations:
   >   ● Modify the **redis.conf** file to allow access from any IP addresses.
   >   ● Set a password for accessing the Redis instance in the **redis.conf** file.
   >   ● Disable the firewall.
   > ● If the function needs to access DCS APIs, **create an agency** and grant required permissions.

7.3 What Do I Do If My Function Cannot Connect to Redis Through a VPC?

Symptom

With the same code, the function can connect to Redis 1 through VPC 1, but cannot connect to Redis 2 through VPC 2.

Solution

1. Check the current configuration: Redis1 is in VPC1, Redis2 is in VPC2, and the Redis client uses the same set of code.

2. Check the log, which shows that the Redis IP address appears as garbled characters when the client tries to connect Redis2 through VPC2.

3. Analyze Redis1 and Redis2. They are the same except for their IP addresses and passwords. The Redis2 password was found to contain an at sign (@). As a result, the IP address was incorrectly intercepted and garbled when the Redis was requested.

4. Change the password of Redis2 to rectify the fault.

7.4 How Do I Configure External Network Access?

By default, functions deployed in a VPC are isolated from the Internet. If a function needs to access both internal and external networks, add a public NAT gateway for the VPC.

Prerequisites

1. You have created a VPC and subnet according to **Creating a VPC**.

2. You have obtained an elastic IP address according to **Assigning an EIP**.

Procedure of Creating a Public NAT Gateway

Step 1 Log in to the NAT Gateway console, and click **Buy Public NAT Gateway**.

Step 2 On the displayed page, enter gateway information, select a VPC and subnet (for example, **vpc-01**), and confirm and submit the settings to buy a NAT gateway. For details, see **Buying a Public NAT Gateway**.

Step 3 Click the public NAT gateway name. On the details page that is displayed, click **Add an SNAT Rule** and click **OK**.

----End

8 Other FAQs

8.1 How Do I View the Alarm Rules Configured for a Function?

Log in to the Cloud Eye console and view alarm rules.

8.2 Does FunctionGraph Support ZIP Decompiling During Video Transcoding?

No. Please decompile your files before uploading them.

8.3 Will Resources Created During FunctionGraph 2.0 OBT Be Automatically Released When They Expire? Will Them Be Billed?

FunctionGraph 2.0 automatically releases resources when they expire and you will not be billed for them.

8.4 What Is an App in FunctionGraph?

An app acts like a folder. After a function is created, it is automatically categorized into the **default** app and cannot be switched to other apps. In the future, functions will be managed by label for better experience.

8.5 Do I Need to Pay for Cold Start Time?

No. Cold start time will not be metered and you do not have to pay for it.

8.6 Why Am I Seeing a Message Indicating that My Account Was Suspended When Creating a Function?

Your account is in arrears.

8.7 Will the Requests of All My Functions in Different Regions Be Billed?

Yes.

9 Migration from FunctionGraph V1 to V2

9.1 What Compatibility Issues Exist During the Migration?

1. **args** difference

   V1:
   ```
   args = parser.parse_args()
   ```
   After migration to V2, it will be:
   ```
   args = parser.parse_args(args=[])
   ```
   The **sys.argv** of Python runtime varies between the two versions.

   For V2, it is **['/home/snuser/runtime/python3.6/server.py', '127.0.0.1:31536', '/opt/function/code']**, whose last two parameters are not available for V1.

2. **asyncio** difference

   V1:
   ```
   loop = asyncio.get_event_loop()
   loop.run_until_complete(func(arg1, arg2))
   loop.close()
   ```
   After migration to V2, it will be:
   ```
   loop_tmp = asyncio.new_event_loop()
   asyncio.set_event_loop(loop_tmp)
   loop = asyncio.get_event_loop()
   loop.run_until_complete(func(arg1, arg2))
   loop.close()
   ```
   In V1, **asyncio.get_event_loop()** obtains the current event loop from the OS thread (main thread). It will throw **RuntimeError** in V2, because Python runtime of V2 does not run functions in the main thread.

   To use **asyncio** in V2, create an event loop.

9.2 Why Can't I Print Logs of a Python 2.7 Function After the Execution of reload(sys)?

Try printing logs with **context.getLogger()**.

```
log = context.getLogger()
log.info("test")
```