FunctionGraph

FAQs

Issue 01

Date 2021-08-28

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

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.

1.5 What Runtimes Does FunctionGraph Support?

Table 1-1 lists the runtimes supported by FunctionGraph.

Table 1-1 Supported runtimes and versions

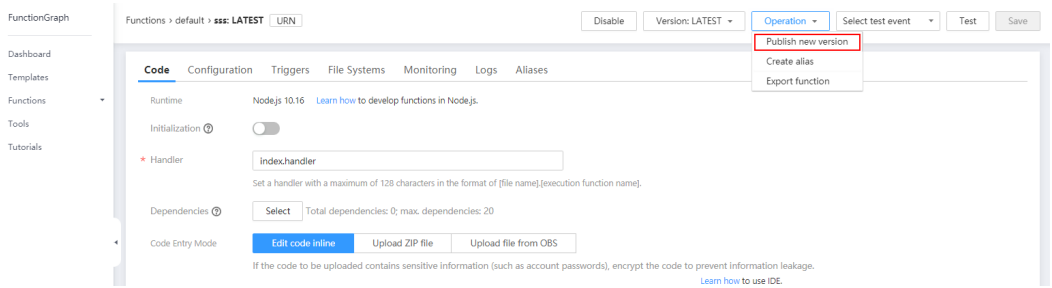
Runtime	Version
Python	2.7 and 3.6
Node.js	6.10, 8.10, 10.16, and 12.13
Java	8
Go	1.8 and 1.x
C#.NET Core	2.0, 2.1, and 3.1
PHP	7.3

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

512 MB ephemeral disk space is allocated to each FunctionGraph function. A tenant can upload deployment packages up to 10 GB in size. For more information, see **Quotas and Usage Restrictions**.

1.7 Does FunctionGraph Support Function Versioning?

FunctionGraph supports function versioning. You can choose **Operation** > **Publish new version** on the function details page.



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. To read the **test.conf** file in the same directory level as the handler file, use the relative path **code/test.conf** or an absolute path (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 function provided by FunctionGraph.

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 a message indicating insufficient permission is displayed when you create, delete, modify, or query functions and triggers as an IAM user in FunctionGraph,

contact the administrator to obtain required permissions. For example, you need to obtain the **Tenant Administrator** permission to create OBS buckets and triggers. Permissions assignment should follow the principle of least privilege to ensure security. For more information, see **Permissions Description**.

1.12 How Do I Create Function Dependencies?

You are advised to create function dependencies in CentOS 7 or EulerOS.

NOTE

- If the modules to be installed need dependencies such as **.dll**, **.so**, and **.a**, archive them to a **.zip** package.
- 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 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 dependency packages, 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. In this way, you can generate a dependency that contains both MySQL and Redis.

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

1.13 What Is the Quota of FunctionGraph?

For details about the resource quota of FunctionGraph, see **Quotas and Usage Restrictions**. 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 Private DNS Domain Names?

FunctionGraph cannot directly parse private Domain Name Service (DNS) domain names. To parse them, call DNS APIs and perform the following steps.

Procedure

Step 1 Obtain the private domain name and ID.

To obtain the private domain name and ID:

1. Log in to the DNS console.
2. Obtain the domain ID.
3. Obtain the private domain name.

Step 2 Create a function.

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

The italic content in bold indicates the DNS endpoint. Change it based on the site requirements. You can obtain the DNS endpoint from **Regions and Endpoints**. The following example uses the **CN North-Beijing4** region.

```
# -*- coding:utf-8 -*-
import json
import os
import requests

def handler(event, context):
    zone_id = context.getUserData("zone_id")
    domain =  context.getUserData("domain")
    token=context.getToken()
    ips = domainResolved(zone_id, domain,token)
    if ips != []:
        print(ips)
        return ips

# GET /v2/zones/{zone_id}/recordsets
def domainResolved(zone_id, domain,token):
    url = "https://dns.cn-north-1./v2/zones/%s/recordsets" % zone_id
    headers = {'content-type': 'application/json',
               "X-Auth-Token": token}
    resp = requests.get(url, headers=headers)
    if resp.status_code == 200:
        text = json.loads(resp.text)
        for record in text["recordsets"]:
            if record["name"] == domain:
                ips = record["records"]
                return ips
    else:
        print("[GET /v2/zones/{zone_id}/recordsets] failed, response: %s" % resp.text)
        return []
```

Step 3 Create a DNS agency for the function.

On the IAM console, create an agency with the **DNS ReadOnlyAccess** system policy configured.

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 a function.

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

1. For **Agency**, select the agency created in **Step 3**.
2. Enter the environment variables, that is, the domain name and domain ID obtained in **Step 1**.

Step 5 Save the configuration and execute the function.

If the function is successfully executed, the corresponding IP address is returned. If an error occurs during DNS API invocation, no IP address is returned.

----End

2 Function Creation FAQs

2.1 Does FunctionGraph Support Class Libraries?

FunctionGraph supports both standard libraries and non-standard third-party libraries. For details about the non-standard third-party libraries integrated in FunctionGraph, see **Dependency Management**.

2.2 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.

2.3 What Are the Rules for Packaging a Function Project?

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

2.4 How Does FunctionGraph Isolate Code?

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

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 What If Garbled Characters Are Reported for Chinese Comments in Python Functions?

When you use Python to edit code online and need to output Chinese, add the following code to the editor:

```
# -*- coding:utf-8 -*-
import json
def handler (event, context):
output = 'Hello message: ' + json.dumps(event,ensure_ascii=False)
return output
```


4 Function Execution FAQs

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

For function invocation, the execution duration is within 900s. The default execution timeout duration is 3s. You can set the timeout duration (unit: s) to an integer ranging from 3 to 900. If you set the execution timeout duration of a function to 3s, it will be terminated after 3s.

4.2 Which Steps Are Included in Function Execution?

Function execution includes two steps:

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

4.3 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.

4.4 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.

4.5 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:

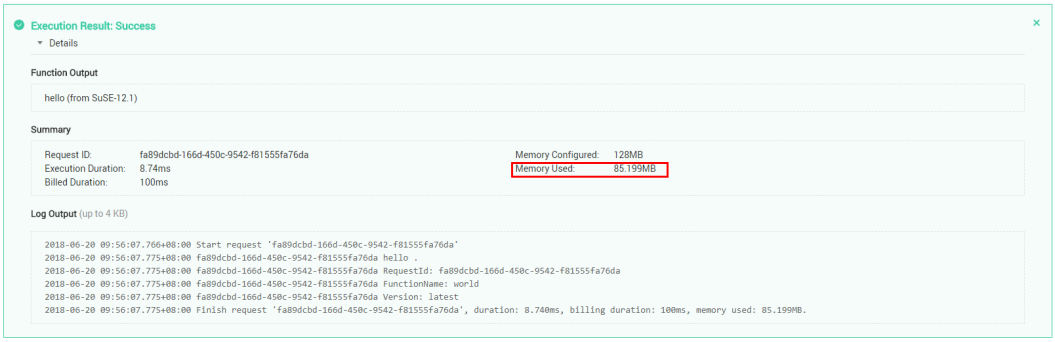
- Allocate more memory to the function.
- Simplify function code, for example, delete unnecessary dependency packages.
- When using C# in non-concurrent scenarios, you can also:

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

4.6 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, as shown in **Figure 4-1**.

Figure 4-1 Actual memory usage



5 Function Configuration FAQs

5.1 Can I Set Environment Variables When Creating Functions?

You can set environment variables to dynamically pass settings to your function code and libraries without the need to change your code. For more information, see **Environment Variables**.

5.2 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 External Resource Access FAQs

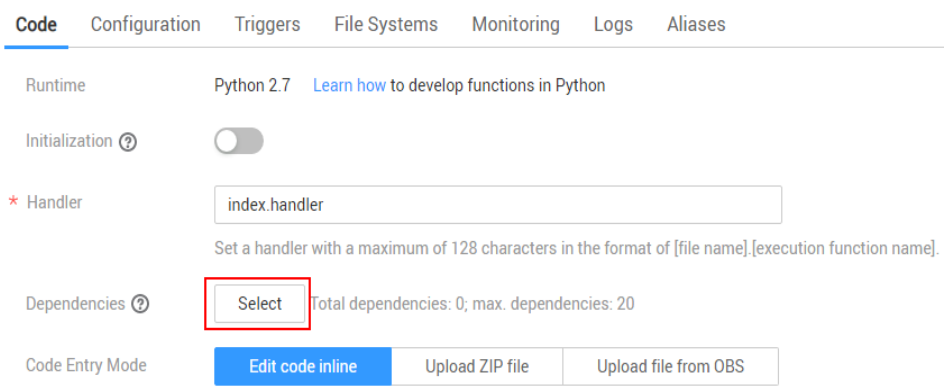
6.1 How Does a Function Access the MySQL Database?

Perform the following operations:

1. Check whether the MySQL database is deployed in a VPC.
   - If the MySQL database is deployed in a VPC, configure the same VPC and subnet as the MySQL database for the function by referring to **Configuring VPC Access**.
   - If the MySQL database is built on a public network, obtain its public IP address.
2. Compile code for connecting a function to the MySQL database.
   a. Create a function, and add the **pymysql** dependency to the function on the **Code** tab page, as shown in **Figure 6-1**.

      For details about how to upload the **pymysql** dependency, see **Creating a Dependency**.

      Figure 6-1 Adding a dependency



NOTE

FunctionGraph provides public dependency **pymysql**. Therefore, you can use it directly.

b. Edit the following code to connect the function to the MySQL database:

```
# -*- coding:utf-8 -*-
import pymysql.cursors

def handler (event, context):
    # Connect to the database
    connection = pymysql.connect(host='host_ip',
                                 user='user',
                                 password='passwd',
                                 db='db',
                                 charset='utf8mb4',
                                 cursorclass=pymysql.cursors.DictCursor)

    try:
        with connection.cursor() as cursor:
            # Create a new record
            sql = "INSERT INTO `users` (`email`, `password`) VALUES (%s, %s)"
            cursor.execute(sql, ('webmaster@python.org', 'very-secret'))

        # connection is not autocommit by default. So you must commit to save
        # your changes.
        connection.commit()

        with connection.cursor() as cursor:
            # Read a single record
            sql = "SELECT `id`, `password` FROM `users` WHERE `email`=%s"
            cursor.execute(sql, ('webmaster@python.org',))
            result = cursor.fetchone()
            print(result)
    finally:
        connection.close()
        output = '^_^'
    return output
```

NOTE

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

6.2 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.

