

Service Overview

Issue 01
Date 2026-01-23



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2026. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property 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 Cloud 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.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 FunctionGraph Infographics.....	1
2 What Is FunctionGraph?.....	3
3 Product Features.....	6
4 Product Advantages.....	10
5 Application Scenarios.....	12
6 Product Features.....	14
7 Function Types.....	16
7.1 Event Functions.....	16
7.2 HTTP Functions.....	17
8 Notes and Constraints.....	18
9 Security.....	21
9.1 Shared Responsibilities.....	21
9.2 Asset Identification and Management.....	23
9.3 Identity Authentication and Access Control.....	26
9.4 Data Protection.....	26
9.5 Audit and Logs.....	27
9.6 Resilience.....	28
9.7 Security Risk Monitoring.....	28
9.8 Certificates.....	29
9.9 Code Signature.....	29
9.10 Data Plane Assurance.....	30
10 Permissions Management.....	32
11 Concepts.....	38
12 Relationships Between FunctionGraph and Other Services.....	41

1 FunctionGraph Infographics

FunctionGraph 2.0
Next-Generation Function Computing

FunctionGraph allows you to run your code without provisioning or managing servers while ensuring **high availability and scalability**.

Simply upload your code and set execution conditions, and FunctionGraph will take care of the rest. The way for what you do, not when your code is not running.

FunctionGraph 2.0 is a compute service powered by Huawei YunRong. YunRong is a **distributed kernel** for container-native serverless technologies, **enables** function programming, multi-tenant full function runtime, and efficient load service distribution. Developers now build cloud-native applications more easily.

1. Multiple Development Languages and Triggers

Six Languages

1) Develop functions using the four mainstream languages.

- Python 3.8
- Node.js 12
- Java 8
- Go 1.12

2) Use custom programming language or image packages.

- ARM64 Linux Container
- ARM64 Linux

10+ Triggers for Function Invocation

Synchronous Invocation: The client needs to wait until it receives a response. The requested function must be executed and return a response as the client can proceed to the next step.

Asynchronous Invocation: The client does not need to wait after it sends requests to the server. The server jobs required requests first, it queues, and processes them one by one. The server then returns the processed results to the client.

2. Easier Development with Function Flows

Drag and drop to create flow charts and orchestrate components.

Commonall

- Component
- Flowchart
- Function
- Event
- Condition
- Parallel
- Sequence
- Loop
- Timeout
- Exception
- Retry
- Wait
- Log

3. On- and Off-Cloud Development and Debugging

(On-Cloud) CloudIDE

Cloud IDE (VS Code) → Push Function to Cloud → Push Function to Cloud → Push Function to Cloud

(Off-Cloud) VSCode Plug-In

Cloud IDE (VS Code) → Push Function to Cloud → Push Function to Cloud → Push Function to Cloud

4. Serverless Reconstruction with HTTP Functions

Package the original web service code into an HTTP function using API Gateway and FunctionGraph for serverless reconstruction.

Test with web management → API Gateway → FunctionGraph → API Gateway → FunctionGraph → API Gateway → FunctionGraph

Serverless reconstruction → API Gateway → FunctionGraph → API Gateway → FunctionGraph

5. Flexible Resource Scheduling

To take maximum compute resources under any strategy in the case of varying traffic, and maintain optimal performance needs to be balanced with usage efficiency, and **dynamic resource assignment to the server**.

Optimal server → API Gateway → FunctionGraph → API Gateway → FunctionGraph → API Gateway → FunctionGraph

6. Cold Start in Hundreds of Milliseconds

Cold start: 300 ms

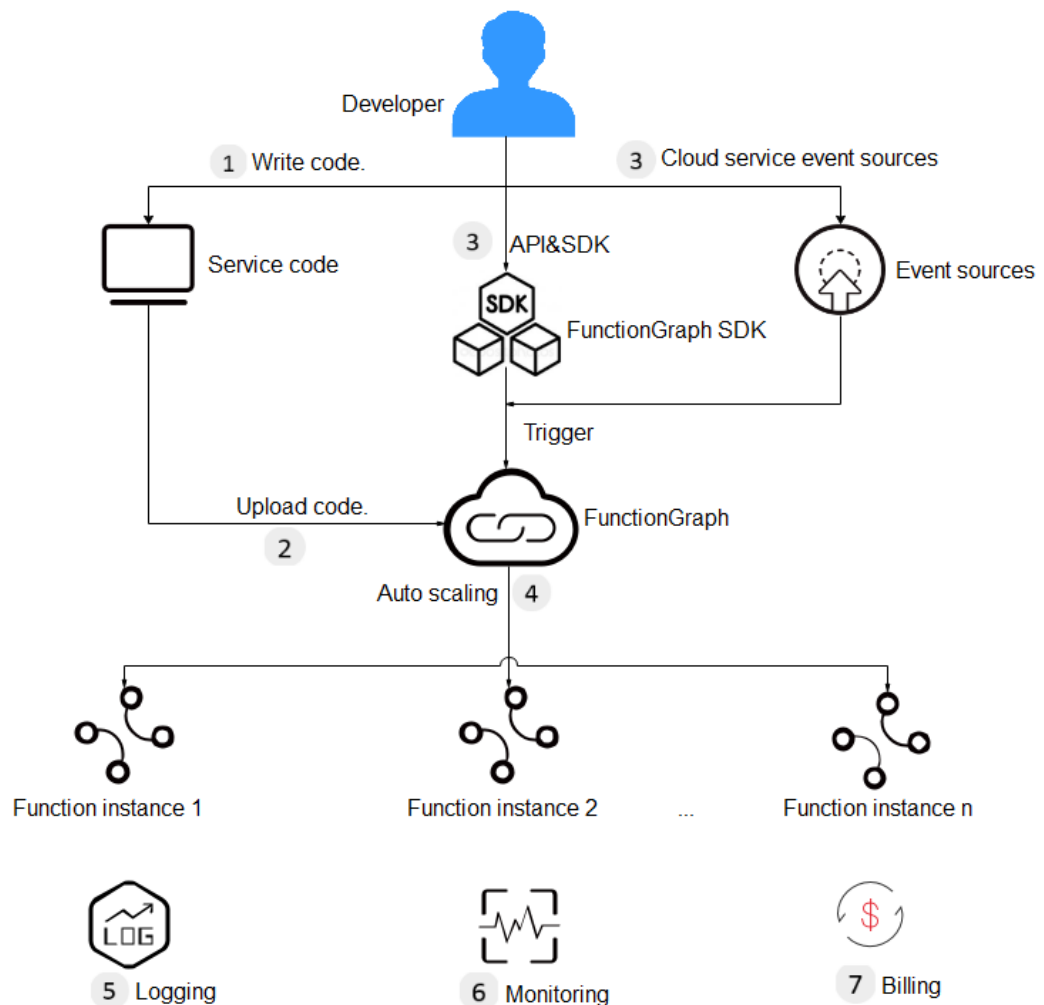
Hot start: 50 ms

2 What Is FunctionGraph?

FunctionGraph hosts and computes event-driven functions in a serverless context while ensuring high availability, high scalability, and zero maintenance. All you need to do is write your code and set conditions. You pay only for what you use and you are not charged when your code is not running.

Figure 2-1 shows the process of using FunctionGraph.

Figure 2-1 Usage process



Feature Overview

1. Write code.

Write code in Node.js, Python, Java, C#, PHP, Go, or Custom runtime. For details, see the [FunctionGraph Developer Guide](#).

2. Upload code.

Currently, you can edit code inline, upload a ZIP or JAR file, or obtain a ZIP file from OBS. For details, see [Table 3-2](#).

3. Trigger functions by API calls or cloud service events.

Call RESTful APIs or use cloud service event sources to trigger function execution and generate instances to implement service functions. For details, see [Function Triggers](#).

4. Auto scaling is implemented.

During function execution, FunctionGraph scales automatically based on the number of requests without the need for configurations. For details about the

maximum number of function instances that can be run concurrently, see [Notes and Constraints](#).

5. View logs.

View run logs of function as FunctionGraph is interconnected with Log Tank Service (LTS). For details, see [Logs and Metrics](#).

6. View monitoring information.

Monitor function information in graphics as FunctionGraph is interconnected with Application Operations Management (AOM). For details, see [Logs and Metrics](#).

7. Billing mode

After the function execution is complete, you will be billed based on the number of requests and the execution duration. For details, see [Billing Overview](#).

3 Product Features

Function Management

FunctionGraph provides console-based function management.

- The Node.js, Java, Python, Go, C#, PHP, and custom runtimes are supported. [Table 3-1](#) provides the details.

 **NOTE**

You are advised to use the latest runtime version.

Table 3-1 Runtimes

Runtime	Supported Version
Node.js	6.10, 8.10, 10.16, 12.13, 14.18, 16.17, 18.15
Python	2.7, 3.6, 3.9, 3.10
Java	8, 11, 17
Go	1.x
C#	.NET Core 2.1, .NET Core 3.1, .NET Core 6.0
PHP	7.3
Cangjie	1.0
Custom	-

- Multiple code entry modes
FunctionGraph allows you to edit code inline, upload a ZIP file from Object Storage Service (OBS), or directly upload a ZIP or JAR file. [Table 3-2](#) lists the code entry modes supported for each runtime.

Table 3-2 Code entry modes

Runtime	Editing Code Inline	Uploading a ZIP File	Uploading a JAR File	Uploading a ZIP File from OBS
Node.js	Supported	Supported	Not supported	Supported
Python	Supported	Supported	Not supported	Supported
Java	Not supported	Supported	Supported	Supported
Go	Not supported	Supported	Not supported	Supported
C#	Not supported	Supported	Not supported	Supported
PHP	Supported	Supported	Not supported	Supported
Cangjie	Not supported	Supported	Not supported	Supported
Custom	Supported	Supported	Not supported	Supported

Function Triggers

Table 3-3 lists the invocation modes for different trigger types.

Table 3-3 Function trigger invocation

Trigger	Invocation Mode
API Gateway (Dedicated)	Synchronous invocation is used by default. You can change it to asynchronous invocation. For details, see Asynchronous Invocation .
Timer	Synchronous (default)
Cloud Trace Service (CTS)	Asynchronous (default and cannot be changed)
Document Database Service (DDS)	Asynchronous (default and cannot be changed)
DMS (for Kafka)	Asynchronous (default and cannot be changed)
Kafka (Open-Source)	Asynchronous (default and cannot be changed)
Log Tank Service (LTS)	Asynchronous (default and cannot be changed)

Trigger	Invocation Mode
Simple Message Notification (SMN)	Asynchronous (default and cannot be changed)

Logs and Metrics

FunctionGraph graphically displays function monitoring metrics and collects function running logs, enabling you to view function statuses, and locate problems by querying logs.

For details about how to query logs, see [Managing Function Logs](#).

For details about a single monitoring metric, see [Function Monitoring](#).

For details about tenant-level monitoring information, see [Introduction to Dashboard](#).

Function Initialization

The initializer interface is introduced to:

- Isolate function initialization and request processing to enable clearer program logic and better structured and higher-performance code.
- Ensure smooth function upgrade to prevent performance loss during the application layer's cold start initialization. Enable new function instances to automatically execute initialization logic before processing requests.
- Identify the overhead of application layer initialization, and accurately determine the time for resource scaling and the quantity of required resources. This feature makes request latency more stable when the application load increases and more function instances are required.

Unified Plug-in for Development and Debugging

VSCode plug-in (off-cloud):

Create a function using a template, view the function on the cloud, download the function to a local PC for debugging, use the VSCode plug-in to debug the function, and then push the function to the cloud.

HTTP Functions

This feature is supported only by FunctionGraph v2.

You can set **Function Type** to **HTTP Function** on the function creation page. HTTP functions are designed to optimize web services. You can send HTTP requests to URLs to trigger function execution. HTTP functions support APIG triggers only.

Custom Images

This feature is supported only by FunctionGraph v2.

You can directly package and upload container images. The images are loaded and started by the platform and can be called in a similar way as HTTP functions.

Unlike the previous code upload mode, you can use a custom code package, which is flexible and reduces migration costs.

4 Product Advantages

No Servers to Manage

FunctionGraph automatically runs your code and frees you from provisioning and managing servers, allowing you to focus on business innovation.

Auto Scaling

FunctionGraph automatically scales to suit fluctuations in resource demands and ensures that the service remains accessible even during peaks and spikes.

It automatically scales in/out resources based on the number of service requests, and distributes requests to function instances through automatic load balancing.

In addition, the system intelligently preheats instances for the traffic loads to reduce the impact of cold start on your services.

Event-based Triggering

FunctionGraph integrates with multiple cloud services using an event-based triggering mechanism to meet service requirements.

It is interconnected with the LTS and Cloud Eye services, allowing you to view function logs and metrics without the need for any configurations.

High Availability

If an instance becomes faulty, FunctionGraph starts another instance to process new requests and releases resources from the unhealthy instance.

Pay per Use

You will be billed based on the number of function requests and execution duration and will not be charged when your code is not running.

Reserved Instance Billing

Reserved instances can be created to initialize functions to eliminate the influence of cold start on your services. Reserved instances are always alive in the execution environment.

For the use of reserved instances, you will be billed based on the number of requests and the running duration of reserved instances. The minimum running duration is 60s.

Dynamic Resource Adjustment

Resource specifications can be dynamically adjusted to minimize resource usage and reduce costs.

5 Application Scenarios

FunctionGraph is suitable for various scenarios, such as real-time file processing, real-time data stream processing, web & mobile application backends, and AI application.

Scenario 1: Event-Driven Applications

Services are executed in event-driven mode and resources are provisioned based on demands. Developers do not need to be concerned about service peaks or troughs. Idle resources are not billed, reducing O&M costs. Event-driven applications include live streaming/transcoding, real-time data stream processing, and IoT rule/event processing.

- **Real-time file processing**

When files are uploaded from a client to OBS, functions can be triggered to create image thumbnails in real time, convert video formats, aggregate and filter data files, or implement other file operations.

Advantages:

- FunctionGraph automatically allocates resources to run more function instances as the number of received requests increases.
- Files are uploaded to OBS to trigger file processing functions.
- You will be billed only for resources used to process files as needed (you are not billed for idle resources during lows in demand).

- **Real-time data stream processing**

FunctionGraph works with DIS to process data streams in real time.

FunctionGraph supports application activity tracking, sequential transaction processing, data stream analysis, data sorting, metric generation, log filtering, indexing, social media analysis, and IoT device data telemetry and metering.

Advantages:

- Data is collected by means of DIS streams to trigger data processing functions.
- FunctionGraph automatically allocates resources to run more function instances as the number of received requests increases.
- You will be billed only for resources used to process files as needed (you are not billed for idle resources during lows in demand).

Scenario 2: Web Applications

Interconnect FunctionGraph with other cloud services or your VMs to quickly build highly available and scalable web & mobile backends. Web applications include mini programs, web pages/apps, chatbots, and Backends for Frontends (BFF).

Advantages:

- FunctionGraph ensures high reliability of website data using OBS and CloudTable, and high-availability of website logic using API Gateway.
- FunctionGraph automatically allocates resources to run more function instances as the number of received requests increases.
- You will be billed only for resources used to process files as needed (you are not billed for idle resources during lows in demand).

Scenario 3: AI Applications

Intelligence evolution requires various services to be integrated for quick rollout. These services include third-party service integration, AI inference, and license plate recognition.

Advantages:

- FunctionGraph works with EI services for text recognition and content moderation to suit a wide range of scenarios – make adjustments whenever you need as demands change.
- You only need to apply for related services and write service code without having to provision or manage servers.
- You will be billed only for function execution and used EI services without having to pay for idle resources when service demands are low.

6 Product Features

FunctionGraph provides the following features. For detailed information on region availability of each feature, you can refer to the console.

Function Creation

Create functions from scratch or with a template on the FunctionGraph console.

HTTP Functions

HTTP functions optimize web services and can be triggered by sending HTTP requests to specific URLs.

Container Image

You can directly package and upload container images. The platform then loads and starts these images to create functions.

Function Management

Edit functions inline to modify their code or configurations.

Trigger Management

FunctionGraph supports multiple types of triggers, such as APIG, DIS, and LTS, and uses event sources to trigger function execution.

Environment Variables

Configure encryption settings and environment variables to dynamically pass settings to your function code and libraries without making any code changes.

Dependency Management

Manages all your dependencies. You can upload dependencies from a local path or through OBS if they are too large, and specify names for them. Some runtimes support dependency building.

Version Management

After function code is published, you can modify the version configuration.

Test Event Management

Event data is passed to the handler of your function as an input and persisted for later use.

Reserved Instance Management

Reserved instances can be created and released by you as required. After you create reserved instances for a function, FunctionGraph preferentially forwards requests to the reserved instances. If the number of requests exceeds the processing capability of the reserved instances, FunctionGraph will forward the excessive requests to on-demand instances and automatically allocates execution resources to these requests.

Single-Instance Multi-Concurrency

Some function types can process multiple requests at the same time.

Dynamic Memory

During function execution, resources are dynamically adjusted for the size of the data to process to reduce costs.

Logs and Metrics

You can view function logs and monitor function invocation and running in real time.

Function Development

Multiple runtimes, such as Node.js, Python, and Java, are supported.

API

Create, delete, query, and execute functions by calling APIs.

7 Function Types

7.1 Event Functions

NOTE

You need to choose between **Event Function** and **HTTP Function** when creating a function on FunctionGraph v2.

Overview

FunctionGraph supports event functions. An event can trigger function execution. Generally, it is in JSON format. You can create an event to trigger your function through the cloud service platform or CodeArts IDE Online. All types of triggers supported by FunctionGraph can trigger event functions.

NOTE

1. On the function creation page, **Function Type** is set to **Event Function** by default.
2. During testing, a function can be triggered by simply entering the specified event in JSON format.
3. You can also use triggers to trigger event functions.

Advantages

- **Easy single-node programming**
You can edit event functions on FunctionGraph or upload code packages there and deploy them with just a few clicks. There is no need for you to care about function concurrency or fault rectification.
- **High-performance, high-speed runtimes**
Event functions can be started, scaled, and called within milliseconds. Faults can be detected and rectified within seconds.
- **Complete tool chain**
FunctionGraph provides comprehensive logging, tracing, debugging, and monitoring, allowing developers to roll out functions in just three steps.

Restrictions

Event functions face event source restrictions. You need to comply with the function development rules of the function platform.

7.2 HTTP Functions

Notes and Constraints

This feature is supported only by FunctionGraph v2.

Overview

FunctionGraph supports event functions and HTTP functions. HTTP functions are designed to optimize web services. You can send HTTP requests to URLs to trigger function execution. HTTP functions support APIG and APIC triggers only.

NOTE

1. HTTP functions support the HTTP/1.1 protocol.
2. On the function creation page, **HTTP Function** is newly added.
3. The HTTP function must be set to **bootstrap**. You can directly write the startup command and **allow access over port 8000**.

Advantages

- Support for multiple frameworks
You can use common web frameworks, such as Node.js Express and Koa, to write web functions, and migrate your local web framework services to the cloud with least modifications.
- Fewer request processing steps
Functions can directly receive and process HTTP requests, eliminating the need for API Gateway to convert the JSON format. This accelerates request processing and improves web service performance.
- Premium writing experience
Writing HTTP functions is similar to writing native web services. You can also use native Node.js APIs to enjoy local development-like experience.

Restrictions

- HTTP functions support APIG (dedicated), APIG (shared), and APIC triggers only.
- Multiple API triggers can be bound to the same function, but all the APIs must belong to the same APIG service.
- For HTTP functions, the size of the HTTP response body cannot exceed 6 MB.
- HTTP functions cannot be executed for a long time, invoked asynchronously, or retried.

8 Notes and Constraints

Function Configuration

Table 8-1 Function configuration restrictions

Restriction Item	Description
Maximum number of versions allowed for a function	20 (including the latest version)
Maximum number of aliases allowed for a function	10 Each version can be associated with only one alias.
Maximum number of triggers allowed for a function version	10
Size of all environment variables of a function	4,096 characters
Maximum number of functions that can be created under an account	400
Maximum size of deployment packages allowed for an account	10 GB
Number of concurrent executions per account	100 For more concurrent executions, submit a service ticket .
Maximum number of reserved instances that an account can create	90 (Number of concurrent executions per account x 90%) For more reserved instances, submit a service ticket .
Maximum number of tags that can be created for a function	20 To use the pre-defined tags of Tag Management Service (TMS), enable this service.

Restriction Item	Description
Network	If VPC access is enabled, the default NIC is disabled and the NIC bound to the VPC will be used instead. Whether public access is supported depends on the VPC.
Async notification	To avoid cyclic invocation, do not set two functions as asynchronous execution targets of each other.
Logs	<ul style="list-style-type: none"> The default log group will be unavailable if you switch it to another one or disable Collect Logs. Each function can have a maximum of 10 tags.

Function Code

Table 8-2 Function code restrictions

Restriction Item	Description
Size of a code deployment package (in ZIP or JAR format) that can be uploaded to the FunctionGraph console	40 MB
Size of a code deployment package (in ZIP or JAR format) that can be edited inline during function API invocation	50 MB
Size of exported resources	≤ 50 MB
Size of an original code deployment package allowed during function API invocation	<ul style="list-style-type: none"> ZIP: 1,500 MB (after decompression) OBS bucket: 300 MB (after compression)
Maximum size of code that can be displayed on the console	20 MB
Private dependency	<ul style="list-style-type: none"> ZIP: 10 MB (If the file size exceeds 10 MB, upload it via OBS.) OBS: OBS URL with the file in ZIP format

Function Running Resources

Table 8-3 Function running resource restrictions

Restriction Item	Description
Ephemeral disk space (/tmp space)	512 MB
File descriptors	2048
Total number of processes and threads	1024
Maximum execution duration per request	259,200s If the execution takes longer than 90 seconds, use asynchronous invocation. For longer execution duration, submit a service ticket .
Valid payload size of invocation request body (synchronous invocation)	6 MB
Valid payload size of invocation response body (synchronous invocation)	6 MB By default, the size of the returned string or the JSON string serialized from the response body is less than or equal to 6 MB. The actual data size varies depending on the backend settings of FunctionGraph. The backend determines the size of the serialized data with a byte-level deviation. The actual valid payload size is 6 MB ± 100 bytes.
Valid payload size of invocation request body (asynchronous invocation)	256 KB
Image size per function	10 GB
Instances per tenant	1000 For more instances, submit a service ticket .
Max. memory per function	10 GB
Bandwidth	Unlimited
Single log size	Unlimited
Maximum execution duration of initializer	259,200s For longer initializer execution duration, submit a service ticket .

9 Security

9.1 Shared Responsibilities

Huawei guarantees that its commitment to cyber security will never be outweighed by the consideration of commercial interests. To cope with emerging cloud security challenges and pervasive cloud security threats and attacks, Huawei Cloud builds a comprehensive cloud service security assurance system for different regions and industries based on Huawei's unique software and hardware advantages, laws, regulations, industry standards, and security ecosystem.

Unlike traditional on-premises data centers, cloud computing separates operators from users. This approach not only enhances flexibility and control for users but also greatly reduces their operational workload. For this reason, cloud security cannot be fully ensured by one party. Cloud security requires joint efforts of Huawei Cloud and you, as shown in [Figure 9-1](#).

- **Huawei Cloud:** Huawei Cloud is responsible for infrastructure security, including security and compliance, regardless of cloud service categories. The infrastructure consists of physical data centers, which house compute, storage, and network resources, virtualization platforms, and cloud services Huawei Cloud provides for you. In PaaS and SaaS scenarios, Huawei Cloud is responsible for security settings, vulnerability remediation, security controls, and detecting any intrusions into the network where your services or Huawei Cloud components are deployed.
- **Customer:** As our customer, your ownership of and control over your data assets will not be transferred under any cloud service category. Without your explicit authorization, Huawei Cloud will not use or monetize your data, but you are responsible for protecting your data and managing identities and access. This includes ensuring the legal compliance of your data on the cloud, using secure credentials (such as strong passwords and multi-factor authentication), and properly managing those credentials, as well as monitoring and managing content security, looking out for abnormal account behavior, and responding to it, when discovered, in a timely manner.

Figure 9-1 Huawei Cloud shared security responsibility model

Cloud security responsibilities are determined by control, visibility, and availability. When you migrate services to the cloud, assets, such as devices, hardware, software, media, VMs, OSs, and data, are controlled by both you and Huawei Cloud. This means that your responsibilities depend on the cloud services you select. As shown in [Figure 9-1](#), customers can select different cloud service types (such as IaaS, PaaS, and SaaS services) based on their service requirements. As control over components varies across different cloud service categories, the responsibilities are shared differently.

- In on-premises scenarios, customers have full control over assets such as hardware, software, and data, so tenants are responsible for the security of all components.
- In IaaS scenarios, customers have control over all components except the underlying infrastructure. So, customers are responsible for securing these components. This includes ensuring the legal compliance of the applications, maintaining development and design security, and managing vulnerability remediation, configuration security, and security controls for related components such as middleware, databases, and operating systems.
- In PaaS scenarios, customers are responsible for the applications they deploy, as well as the security settings and policies of the PaaS middleware, database, and network access under their control.
- In SaaS scenarios, customers have control over their content, accounts, and permissions. They need to protect their content, and properly configure and protect their accounts and permissions in compliance with laws and regulations.

On-premises (On-Prem): Software and IT infrastructure that are deployed and managed by customers within their own data centers, rather than be deployed by remote cloud service providers.

Infrastructure as a Service (IaaS): Cloud service providers offer compute, network, storage, and more infrastructure services, including [Elastic Cloud Server \(ECS\)](#), [Virtual Private Network \(VPN\)](#), and [Object Storage Service \(OBS\)](#).

Platform as a Service (PaaS): Cloud service providers deliver platforms required for application development and deployment, such as [ModelArts](#). Customers do not need to maintain the underlying infrastructure.

Software as a Service (SaaS): Cloud service providers offer complete application software. Customers use the software directly without the need to install the application, maintain it, or manage its underlying platform or infrastructure.

9.2 Asset Identification and Management

FunctionGraph provides a secure running environment throughout the lifecycle of a function. You need to use the security mechanisms provided by FunctionGraph to ensure the security of your code, dependencies, and configurations.

Operating Environment Security

FunctionGraph provides compute nodes and function instances for code execution, offering scalable and secure resources based on user demand.

Compute Node Security

Compute nodes provide the following Huawei Cloud standard security protection capabilities. For details, see the [Huawei Cloud Security Overview](#).

- **Multi-AZ and multi-cluster DR:** Compute nodes are deployed across multiple AZs and clusters within a region, ensuring availability.
- **Independent VPC environment:** Compute nodes are located in a separate, isolated VPC. You cannot directly access compute nodes.
- **Host security protection:** Compute nodes use Huawei Cloud [HSS](#) for vulnerability detection, security monitoring, and defense, with rapid response and remediation in collaboration with Huawei Cloud's security team.
- **Vulnerability fixing or security upgrade:** FunctionGraph fixes vulnerabilities and performs security upgrade for compute nodes. The upgrade process is transparent to users. If incompatibility risks exist, users will be notified via notice or SMS message and provided with adaptation solutions to ensure smooth service migration.

Function Instance Security

Function instances provide function-level isolation. Each instance can run only one function.

- **Network isolation:** Function instances cannot directly access each other, and function instances and nodes cannot directly access each other. You can determine whether to enable public access or VPC access. For details, see [Configuring the Network](#).
- **Instance freezing:** When malicious tenant attacks are detected, FunctionGraph can immediately freeze and isolate the malicious user's function instance to ensure the running environment's security.
- **Vulnerability fixing and security upgrade:** FunctionGraph fixes vulnerabilities and performs security upgrade for function instances. The upgrade process is transparent to users. If incompatibility risks exist, users will be notified via notice or SMS message and provided with adaptation solutions to ensure smooth service migration.
- **Runtime end of maintenance:** As community support ends, the runtimes provided by FunctionGraph will gradually be phased out. Users are prohibited

from creating functions with unsupported runtimes. Users are advised to migrate existing functions to new runtimes as soon as possible. FunctionGraph does not guarantee the continued normal operation of unsupported runtime versions.

User Code Security

- **Code sharing and download:** FunctionGraph provides users with temporary code and download addresses, and sets a validity period. Users should avoid the leakage of temporary download addresses to reduce the risk of code or library leakage.
- **Sensitive information leakage prevention:** Users need to avoid recording sensitive information, such as access keys (AKs), security keys (SKs), and database passwords, in plaintext in code or dependencies. Tokens and passwords should not be recorded in user code logs.
- **Code vulnerability prevention:** You need to ensure the security of your code, libraries, and dependencies, identify and fix vulnerabilities in a timely manner, and update your functions.

Huawei Cloud provides multiple security cloud services for FunctionGraph to enhance security capabilities such as code scanning and threat analysis.

Table 9-1 Huawei Cloud security cloud services

Service	Description
CodeArts Check	<p>Scans FunctionGraph code from multiple dimensions, covering code style, quality, and security issues. Its core capabilities include:</p> <ul style="list-style-type: none"> • Self-developed scanning engine: Supports mainstream languages like C/C++, Java, and Python, and detects security vulnerabilities (such as buffer overflows, unauthorized access, and encryption issues) as well as code standard issues. • Security standard support: Integrates standards like ISO 5055, CWE, and OWASP Top 10, and includes Huawei's own standards (such as <i>Huawei C/C++ Coding Standard</i>) built based on 30 years of R&D experience to ensure code meets security requirements. • Large-scale scanning: scans tens of billions of code items per day, supports elastic scheduling and disaster recovery, and is suitable for full code check of FunctionGraph.

Service	Description
<p>SecMaster</p>	<p>Combined with Huawei Cloud years of experience in security and based on cloud-native security capabilities, SecMaster provides cloud asset management, security posture management, security information and incident management, security orchestration, automatic responses, and other functions, helping you implement integrated and automatic security operations management.</p> <p>SecMaster analyzes logs of related FunctionGraph cloud services (such as OBS and VPC) to detect malicious behavior in real time.</p> <ul style="list-style-type: none"> • Multi-dimensional log analysis: Collects IAM, DNS, and CTS logs, and uses AI engines and threat intelligence to identify brute-force attacks and penetration attacks. • Alarm and response: generates threat alarms and outputs statistics, helping you handle potential risks in a timely manner and ensuring service stability.

With CodeArts Check and SecMaster, FunctionGraph can protect the entire code process and monitor runtime threats.

User Configuration Security

- **Sensitive information protection:** If your code or configuration contains sensitive information, use encrypted environment variables to prevent sensitive information from being displayed in plaintext on the UI or in the results returned by APIs.
- **Least privilege:** When configuring triggers, VPC access, custom images, or mounting disks, FunctionGraph needs permissions to interact with other cloud services. Follow the principle of least privilege when setting up agencies to minimize the impact of token leakage.
- **KMS-based dynamic encryption and decryption (DEW):** To decrypt sensitive data (such as database passwords and API keys) during function running, use the KMS SDK to dynamically manage keys. You can host the encryption and decryption keys in KMS, and create an agency in IAM to grant FunctionGraph the permission to access KMS (the authorization complies with the least privilege principle). The authorization policy is as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:huaweicloud:kms:REGION:ACCOUNT_ID:keyring/kms-ring-123456/key/kms-key-123456"
    }
  ]
}
```

Add the KMS SDK code snippet to obtain the key for encrypting and decrypting sensitive data. The following uses the Python code snippet as an example.

```
from huaweicloudsdkkms import KmsClient, models

def decrypt_data():
    kms_client = KmsClient(
        secret_id=os.getenv('KMS_SECRET_ID'),
        secret_key=os.getenv('KMS_SECRET_KEY'),
        region_name="cn-north-4"
    )

    decrypt_request = models.DecryptRequest(
        key_id="kms-key-123456",
        ciphertext=b"encrypted_data_base64",
        encryption_algorithm="AES_256_CBC"
    )
    response = kms_client.decrypt(decrypt_request)
    return response.plaintext.decode('utf-8')
```

9.3 Identity Authentication and Access Control

FunctionGraph uses [Identity and Access Management \(IAM\)](#) to authenticate user identities and control access to Huawei Cloud resources.

Identity Authentication

You can access FunctionGraph through the FunctionGraph console, APIs, or SDKs. All these access modes are implemented through REST APIs provided by FunctionGraph. FunctionGraph supports authentication using [token and AK/SK](#).

Access Control

FunctionGraph uses IAM for access control and fine-grained permission management. For details, see [Permissions Management](#).

User authorization should comply with the principle of **least privilege** to effectively reduce the attack scope and minimize the impact on services when credentials are disclosed.

- **Event source configuration:** Create a trigger for an event source and assign permissions to trigger the function.
- **Cloud service access:** To access other cloud services, such as OBS and LTS, grant FunctionGraph with the corresponding access permissions.
- **IAM account authorization:** FunctionGraph can use IAM to grant different function operation permissions to IAM users.

9.4 Data Protection

To prevent your data, such as code and function metadata, from being obtained by unauthorized or unauthenticated entities or individuals, FunctionGraph encrypts the data for transmission.

Data Protection Technologies

[Table 9-2](#) lists the data protection technologies used in FunctionGraph.

Table 9-2 Data protection technologies

Technology	Description
Encrypted transmission	All API requests and internal communications are encrypted using TLS 1.2 or later.
Encrypted storage	Function sensitive information and user code cache are encrypted with AES and decrypted when used.
Others	<ul style="list-style-type: none"> • When you create a function or dependency, your code is stored in a private OBS bucket. You can configure an ACL for each object to ensure that only the specified tenant can read and write the object, isolating access from other tenants. • When you create a function using a custom image, the image is stored in your own SWR. Only your own account can download the image. • Function instances are isolated at the function level, with different functions using separate instances to ensure strict data isolation. • After you stop invoking a function, the backend recycles the function instance after a specified time to prevent data leakage from instance reuse.

9.5 Audit and Logs

Audit

Cloud Trace Service (CTS) records operations on the cloud resources in your account. You can use the logs generated by CTS to perform security analysis, trace resource changes, audit compliance, and locate faults.

After you enable CTS and configure a tracker, CTS records management traces of FunctionGraph for auditing.

For details about how to enable and configure CTS, see [CTS Overview](#).

With CTS, you can record operations associated with FunctionGraph for later query, audit, and backtracking. For details, see [Operations Logged by CTS](#).

Logs

FunctionGraph is interconnected with Log Tank Service (LTS). After enabling LTS, you can query function execution logs on the monitoring page or in LTS to better manage functions. For details, see [Function Logs](#).

9.6 Resilience

Resource Partition Deployment

Huawei Cloud's data centers are deployed around the world. All data centers are running properly. Data centers in two cities are deployed as disaster recovery center for each other. If a data center in city A is down, the data center in city B automatically takes over the job and serves your applications and data in compliance with the regulations to ensure service continuity.

FunctionGraph resources are deployed in multiple zones for higher availability, fault tolerance, and scalability.

FunctionGraph Architecture Features

Based on the Huawei Cloud infrastructure, FunctionGraph provides multiple features to support data fault recovery and backup.

Version management: Manage versions in FunctionGraph to save function code and configurations during development, and to perform blue/green and rolling deployment through aliases. For details, see [Managing Versions](#).

Scalability: When a function receives a new request while processing a previous one, FunctionGraph uses another instance to handle increased load. It can scale up resources to process a maximum of 1,000 concurrent executions in a region, with quotas adjustable as needed. For details, see [Configuring Single-Instance Multi-Concurrency](#).

High availability: FunctionGraph runs functions in multiple availability zones (AZs) to ensure that events can still be processed when services are interrupted in a certain region. If the function is configured to connect to a VPC in your account, specify a subnet for high availability. For details, see [Configuring VPC Access](#).

Retry: For asynchronous invocations and triggers from other services, FunctionGraph automatically retries upon errors (with delays). For synchronous invocations, retries are handled by other clients or Huawei Cloud services. For details, see [Configuring Asynchronous Execution Notification](#).

Dead letter queue: For asynchronous invocations, if all retries fail, FunctionGraph can send requests to a dead letter queue for troubleshooting or reprocessing. (This feature is restricted by the whitelist. [Submit a ticket](#) if needed.)

9.7 Security Risk Monitoring

- FunctionGraph uses Cloud Eye to help you monitor your functions and receive alarms and notifications in real time. The following metrics can be monitored: invocations, errors, duration (maximum, average, and minimum), throttles, and instance statistics.
- Cloud Trace Service (CTS) collects, stores, and queries cloud resource operation records. You can use these records to perform security analysis, track resource changes, audit compliance, and locate faults. CTS can track the entire lifecycle of function resource operations, report alarms for important

configuration changes in real time, and quickly detect and track function resource changes.

- Cloud Firewall (CFW) provides threat prevention at the network layer. It integrates the IPS/IDS feature and supports DDoS attack defense based on traffic analysis, automatic blocking of malicious IP addresses, and automatic optimization suggestions for access control policies. When you configure a VPC and access the public network, you can use CFW to enhance public access security.

For details about FunctionGraph metrics, see [Monitoring](#). To learn how to create alarm rules, see [Creating an Alarm Rule](#).

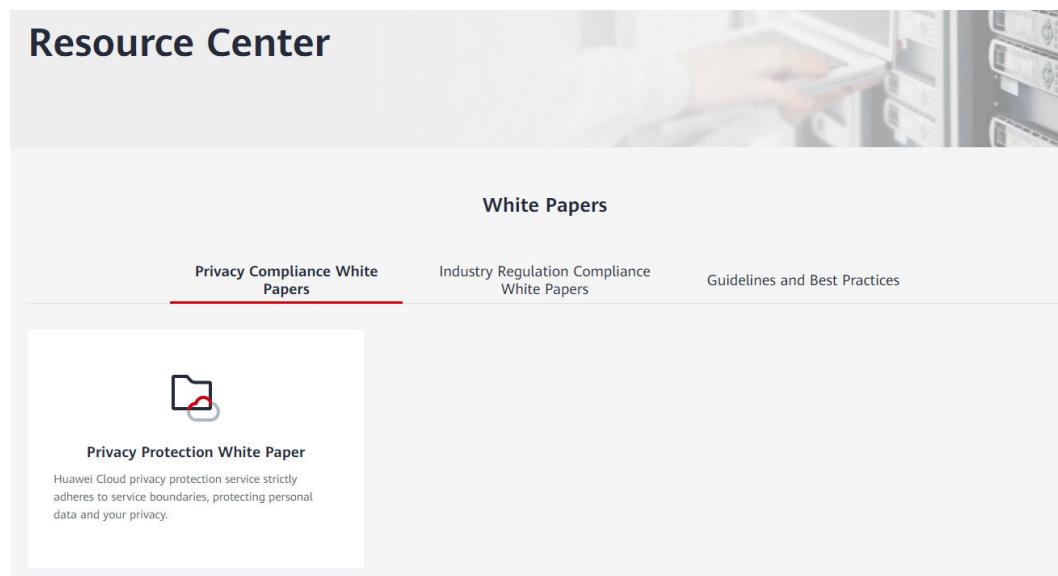
9.8 Certificates

Resource Center

Huawei Cloud also provides the following resources to help users meet compliance requirements.

For details, see [Resource Center](#).

Figure 9-2 Resource center



9.9 Code Signature

When a function is created or modified, FunctionGraph encrypts the function code and generates a signature to prevent inconsistency caused by code file damage or tampering. The signature is stored in the function metainformation.

Code Attributes	
Code Size	293.00 B
Code Signature	876db1994a00115a9f01228ab1992000446c22864576846e1aaf024a4d5984607801164f661
Last Modified	Apr 25, 2023 14:45:30 GMT+08:00
Basic Information	
Runtime	Node.js 14.18
Execution Timeout (s)	3s
Memory (MB)	128 MB
Handler	index.handler

When executing a function, FunctionGraph generates a signature for the current code and compares it with the signature in the metainformation. Only code that

passes the consistency check is executed. If the check fails, FunctionGraph will not execute the code but return an error.

9.10 Data Plane Assurance

Load Balancing and Network

- Load balancing for high availability
Load balancing distributes traffic, preventing single points of failure and improving system reliability.
- VPC configuration for network isolation
Computing nodes are in isolated VPCs, strictly separated from external networks for security.
- Flexible network configuration
By default, functions have public access, but users can configure access to specific VPC resources.

Scheduling

- Multi-cluster for enhanced DR capability
The multi-cluster and multi-AZ architecture allows resource migration during AZ fault, ensuring continuous operations.
- Intelligent scheduling
The intelligent algorithm predicts traffic and automatically scales up resources to quickly respond to burst traffic.
- Elastic and reserved instances
Function instances are classified into elastic and reserved instances. Elastic instances can be dynamically created based on real-time service load changes and automatically released during off-peak hours to avoid resource waste. Reserved instances are configured and created in advance by users based on service requirements and are not automatically released. You can set the maximum number of elastic and reserved instances based on service requirements.

Function Invocation

- Synchronous invocation
Requests are processed directly without caching, suitable for scenarios requiring high real-time performance.
- Asynchronous invocation
Requests are cached in message queues to guarantee execution. Queues are isolated by account or function to avoid data interference among users. The system retries failed invocations up to three times by default. And you can customize the retry times.

Runtime Environment

- Vulnerability fixing and security upgrade

FunctionGraph regularly scans and fixes vulnerabilities in compute nodes and function instances, ensuring a secure and stable runtime environment.

- Immutable code

The code modification takes effect only for newly generated instances and does not affect running instances, ensuring code consistency and stability.

- Non-persistence environment

File systems and memory are released together with the instance, preventing data residue and enhancing resource utilization.

- Exception information collection

The runtime environment automatically collects exceptions and logs to help quickly identify and resolve issues, improving troubleshooting efficiency.

10 Permissions Management

To assign different permissions to employees in your enterprise to access your FunctionGraph resources, IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you secure access to your cloud resources.

With IAM, you can use your account to create IAM users for your employees, and assign permissions to the users to control their access to specific resource types. For example, some software developers in your enterprise need to use FunctionGraph resources but must not delete them or perform any high-risk operations. To achieve this result, you can create IAM users for the software developers and grant them only the permissions required for using FunctionGraph resources.

If your account does not need individual IAM users for permissions management, you may skip over this chapter.

IAM can be used free of charge. You pay only for the resources in your account. For more information about IAM, see [IAM Service Overview](#).

Notes and Constraints

If an IAM user granted the **FunctionGraph FullAccess** permission has no permission to create a certain type of trigger or use a certain function, the relevant service or function does not support fine-grained authentication. In this case, grant the admin permission for this service or function to the user. These services and functions include:

- CTS and APIG: These do not support fine-grained authentication. Add the admin permission for them.
- SMN: This supports fine-grained authentication in some regions. If needed, add the admin permission for this service.
- IoTDA: This is a new trigger type and is not covered in **FullAccess**. When you create an IoTDA trigger, you will be prompted to create an agency and add the **iam:agencies:list** and **iam:agencies:createAgency** permissions.
- TMS, DNS, BSS, Cloud Eye, EG, and DMS: These are new functions and are not covered in **FullAccess**. Add the permissions for them as required.

For more information about the permissions required to use these triggers and relevant functions, see [Table 10-2](#).

Why Is "Insufficient Permission" Displayed After Enterprise Project Authorization?

IAM project/Enterprise project: A custom policy can be applied to IAM projects or enterprise projects or both. Policies that contain actions supporting both IAM and enterprise projects can be assigned to user groups and take effect in both IAM and Enterprise Management. Policies that only contain actions supporting IAM projects can be assigned to user groups and only take effect for IAM. Such policies will not take effect if they are assigned to user groups in Enterprise Management. For details, see [Differences Between IAM Projects and Enterprise Projects](#).

In FunctionGraph, only function resource APIs support enterprise project authorization. For other APIs that support only IAM project authorization:

1. Click **By IAM Project** during authorization.

Figure 10-1 Viewing authorization records by IAM project



2. When selecting the authorization scope, select **Region-specific projects** according to the minimum authorization principle.

FunctionGraph Permissions

By default, new IAM users do not have any permissions assigned. You need to add a user to one or more groups, and assign permissions policies to these groups. The user then inherits permissions from the groups it is a member of. This process is called authorization. After authorization, the user can perform specified operations on FunctionGraph based on the permissions.

FunctionGraph is a project-level service deployed and accessed in specific physical regions. To assign FunctionGraph permissions to a user group, specify the scope as region-specific projects and select projects (such as **cn-north-1**) in relevant regions (such as **CN North-Beijing1**) for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing FunctionGraph, the users need to switch to a region where they have been authorized to use the FunctionGraph service.

You can grant users permissions by using roles and policies.

- **Roles:** A type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. This mechanism provides only a limited number of service-level roles for authorization. When using roles to grant permissions, you may also need to assign other roles on which the permissions depend. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- **Policies:** A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization, meeting requirements for secure access control.

Table 10-1 lists all the system policies supported by FunctionGraph.

Table 10-1 Permissions description

Role/Policy Name	Description	Category	Dependency
FunctionGraph Administrator	This role has the permissions to manage functions, flows, and triggers, and invoke functions. (It will be unavailable soon and therefore not recommended.)	System-defined role	Tenant Guest
FunctionGraph Invoker	This role has the permissions to query functions, flows, triggers, and invoke functions.	System-defined role	N/A
FunctionGraph FullAccess	This policy grants all permissions for FunctionGraph.	System-defined policy	N/A
FunctionGraph ReadOnlyAccess	This policy grants read-only permissions for FunctionGraph.	System-defined policy	N/A
FunctionGraph CommonOperations	This policy grants permissions to query functions and triggers, and invoke functions.	System-defined policy	N/A

Table 10-2 Permissions required to use triggers and relevant functions

Trigger/Function	Permission
APIG (dedicated)	apig:instances:get apig:instances:create apig:instances:update apig:instances:list apig:sharedInstance:operate
CTS	cts:notification:create cts:notification:delete cts:notification:update cts:operation:list cts:tracker:list cts:trace:list

Trigger/Function	Permission
DDS	dds:instance:get dds:instance:list
IoTDA	iotda:routingrules:create iotda:routingrules:delete iotda:routingrules:queryList iotda:routingrules:query iotda:routingactions:create iotda:routingactions:delete iotda:routingactions:query iotda:routingactions:queryList iotda:subscriptions:queryList iotda:rules:modifyStatus iotda:apps:queryList
LTS	lts:groups:create lts:groups:get lts:groups:list lts:groups:put lts:logstreams:delete lts:logstreams:list lts:topics:get lts:subscriptions:create lts:subscriptions:delete lts:subscriptions:put lts:structConfig:create lts:structConfig:get
OBS	obs:bucket:GetBucketLocation obs:bucket:GetBucketNotification obs:bucket:PutBucketNotification obs:bucket:ListBucket
SMN	smn:topic:list smn:topic:update
TMS	tms:predefineTags:list tms:tagValues:list

Trigger/Function	Permission
DNS	dns:recordset:create, dns:recordset:list, dns:recordset:update, dns:zone:create, dns:zone:delete, dns:zone:get, dns:zone:list
BSS	bss:bill:view bss:renewal:view
CES	ces:alarms:get ces:alarms:list ces:alarms:create
DMS	dms:instance:get
EG	eg:subscriptions:get eg:subscriptions:list eg:sources:list eg:sources:get eg:agency:create eg:subscriptions:create eg:subscriptions:delete eg:subscriptions:operate
Distributed Message Service (DMS) for Kafka	dms:instance:list dms:instance:get dms:group:delete

Table 10-3 lists the common operations supported by each system-defined policy of FunctionGraph. Please choose proper system-defined policies according to this table.

Table 10-3 Common operations supported by each system-defined policy

Operation	FunctionGraph Invoker	FunctionGraph Administrator	FunctionGraph ReadOnly Access	FunctionGraph Common Operations	FunctionGraph FullAccess
Creating functions	×	√	×	×	√

Operation	FunctionGraph Invoker	FunctionGraph Administrator	FunctionGraph ReadOnly Access	FunctionGraph Common Operations	FunctionGraph Full Access
Querying functions	√	√	√	√	√
Modifying functions	×	√	×	×	√
Deleting functions	×	√	×	×	√
Invoking functions	√	√	×	√	√
Querying function logs	√	√	√	√	√
Viewing function metrics	√	√	√	√	√

Helpful Links

- [IAM Service Overview](#)
- [Creating a User Group and User and Granting Permissions](#)
- [Permissions Policies and Supported Actions](#)

11 Concepts

Function

Functions are code defined to handle events.

Event Source

An event source is a public cloud service or custom application that publishes events.

Runtime

The runtime provides an execution environment for the corresponding programming language to pass function invocation events, context information, and responses.

FunctionGraph currently supports Node.js, Python, Java, Go, C#, PHP, Cangjie, and custom runtimes.

Synchronous Invocation

Clients wait for explicit responses to their requests from a function. Responses are returned only after the function is invoked.

Asynchronous Invocation

Clients do not care about the function invocation results of their requests. After receiving a request, FunctionGraph puts it in a queue, returns a response, and processes other requests when there are idle resources.

Trigger

A trigger is an event that triggers function execution. Triggers can also be used to trigger functions when specified cloud service events occur.

Function Flow

You can drag, configure, and connect components on the UI and create function flow tasks to complete orchestration in complex scenarios.

Single-Instance Multi-Concurrency

The number of requests that can be concurrently processed by an instance.

Custom Images

You can directly package and upload container images. The platform then loads and starts these images to create functions.

Custom Function Execution

You can customize scripts and files to execute functions.

Function Logs

Logs generated during function invocation.

Function Monitoring

Monitoring information generated during function execution.

Function Version

FunctionGraph allows you to publish one or more versions throughout the development, testing, and production processes to manage your function code. The code and environment variables of each version are saved as a snapshot. After the function code is published, modify settings when necessary.

Function Alias

You can create an alias for a specific function version. To roll back to a previous version, use the corresponding alias to represent the version instead of modifying the function code.

Each function alias can be bound to a major version and an additional version for traffic shifting.

Dependency Package

A dependency contains public libraries that support the running of function code. You can encapsulate the required public libraries into a dependency for easier management, sharing, and smaller deployment sizes.

For more information about function dependencies, see [How Do I Create Function Dependencies](#).

Tracing

Service calls can be traced and recorded, so that the execution paths and statuses of service requests in distributed systems can be restored to quickly locate performance bottlenecks and faults.

Bootstrap File

The **bootstrap** file is the startup file of an HTTP function. The HTTP function can only read **bootstrap** as the startup file name. If the file name is not **bootstrap**, the service cannot be started.

12 Relationships Between FunctionGraph and Other Services

Table 12-1 describes the cloud services that have been interconnected with FunctionGraph.

Table 12-1 Interconnected services

Service	Description
SMN	FunctionGraph functions are constructed to process SMN notifications. For details, see the SMN User Guide .
DMS	FunctionGraph functions are configured to automatically poll DMS queues for messages and process any new messages.
API Gateway	FunctionGraph functions are invoked over HTTPS by defining REST APIs with specified backend services. For details, see the APIG User Guide .
OBS	FunctionGraph functions are created to process OBS bucket events, such as object creation or deletion events. For example, when an image is uploaded to the specified bucket, OBS invokes the function to read the image and create a thumbnail. For details, see OBS User Guide .
DIS	FunctionGraph functions are created to periodically poll DIS streams for new records, such as website click streams, financial transactions, social media streams, IT logs, and location-tracking events. For details, see the <i>DIS User Guide</i> .

Service	Description
CTS	<p>FunctionGraph functions are defined to analyze and process key information in logs according to the event notifications of specified service type and operations configured in CTS.</p> <ul style="list-style-type: none">• With CTS, you can record operations associated with FunctionGraph for later query, audit, and backtracking. For details, see Operations Logged by CTS.• CTS starts recording operations on cloud resources once being enabled. View traces of the last seven days on the CTS console.
Cloud Eye	<p>FunctionGraph is interconnected with Cloud Eye to report monitoring metrics, allowing you to view function metrics and alarm messages through Cloud Eye. For details, see the Cloud Eye User Guide.</p> <p>For details about function metrics supported by Cloud Eye, see Monitoring Configuration.</p>
VPC	<p>Functions can be configured to access resources in Virtual Private Clouds (VPCs) or to access the Internet through source network address translation (SNAT) by binding elastic IP addresses. For details, see the VPC User Guide.</p>