Service Overview

Issue 01

Date 2025-10-14

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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.

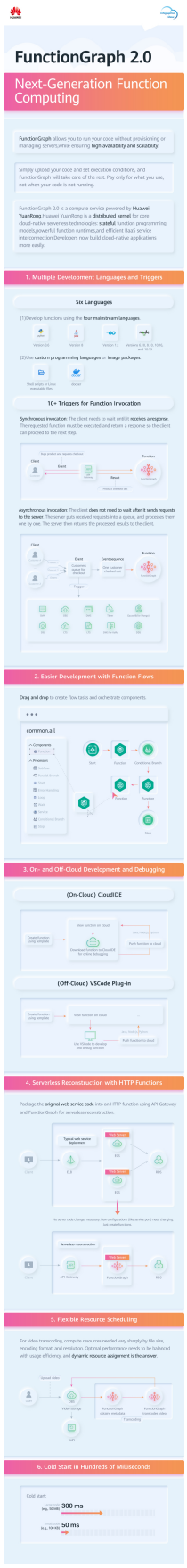
Huawei Cloud Computing Technologies Co., Ltd.

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

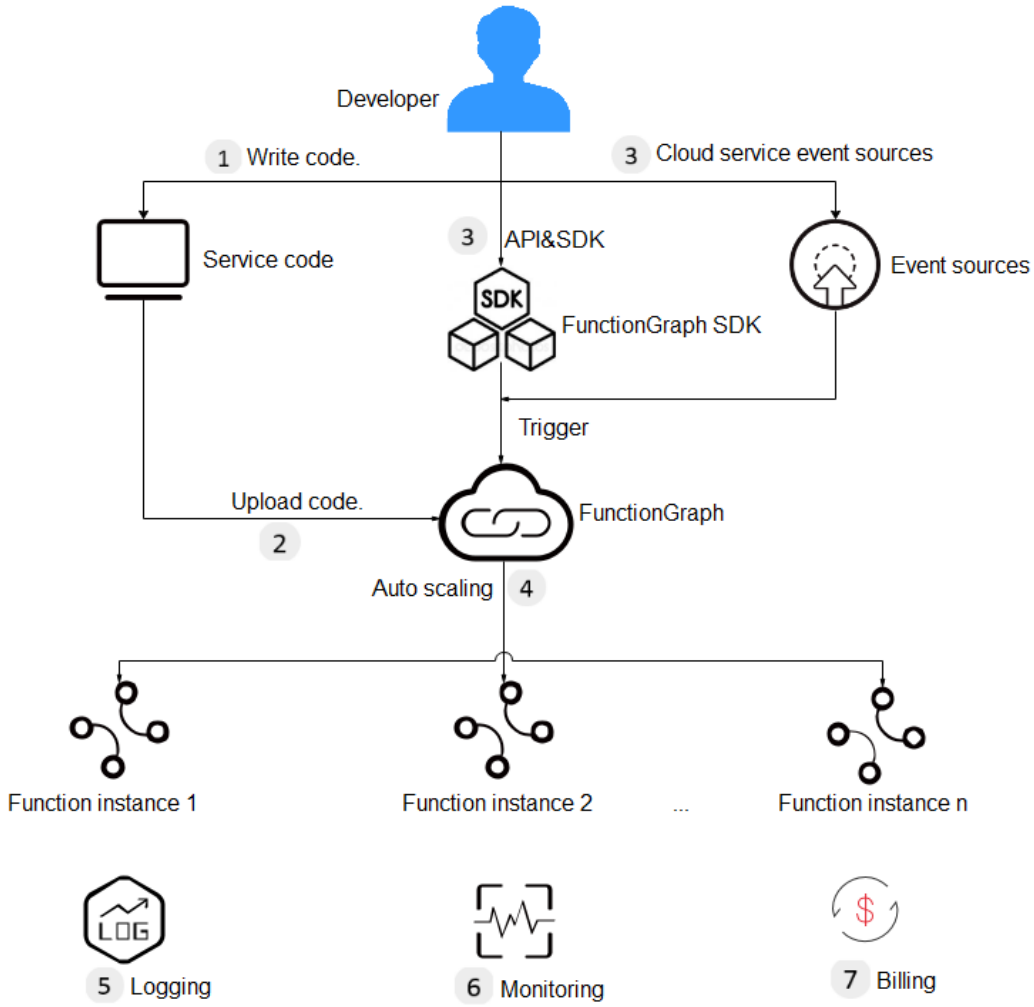

FunctionGraph 2.0
Next-Generation Function Computing
1. Multiple Development Languages and Triggers
Six Languages
10+ Triggers for Function Invocation
2. Easier Development with Function Flows
3. On- and Off-Cloud Development and Debugging
(On-Cloud) CloudIDE
(Off-Cloud) VSCode Plug-in
4. Serverless Reconstruction with HTTP Functions
5. Flexible Resource Scheduling
6. Cold Start in Hundreds of Milliseconds
300 ms
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 **FunctionGraph 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, 20.15
Python	2.7, 3.6, 3.9, 3.10, 3.12
Java	8, 11, 17, 21
Go	1.x
C#	.NET Core 2.1, .NET Core 3.1, .NET Core 6.0, .NET Core 8.0
PHP	7.3, 8.3
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**.
API Connect (APIC)	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)
Data Ingestion Service (DIS)	Asynchronous (default and cannot be changed)

Trigger	Invocation Mode
DMS (for Kafka)	Asynchronous (default and cannot be changed)
Kafka (Open-Source)	Asynchronous (default and cannot be changed)
DMS (for RabbitMQ)	Asynchronous (default and cannot be changed)
GeminiDB Mongo	Asynchronous (default and cannot be changed)
Log Tank Service (LTS)	Asynchronous (default and cannot be changed)
Simple Message Notification (SMN)	Asynchronous (default and cannot be changed)
EventGrid (EG)	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.

Function Flows

Function flows are used to orchestrate functions. Multiple functions can be orchestrated into a flow for coordinating the execution of multiple distributed function tasks.

On the orchestration page, you can connect event triggers, functions, and flow controllers in a flowchart through lines. The output of each node is used as the input of the next node. An orchestrated flow will be executed according to the sequence specified in the flowchart. After it is successfully executed, you can view its execution records for easy diagnosis and debugging.

Function flows have the following features and advantages:

- Features
  a. Visualized function orchestration
  b. Function flow execution engine
  c. Error handling
  d. Visualized monitoring
- Advantages
  a. Build apps with less code

     Function flows allow you to orchestrate functions into a complete application without compiling code. Quick construction and rollout When services are changed, you can quickly adjust flows and roll out services without writing any code.
  b. Comprehensive error handling

     Errors that occur in processes can be captured, retries are supported, and exceptions can be flexibly handled.
  c. Visualized orchestration and monitoring

     Flows can simply be orchestrated by dragging elements.

     You can view flows on the monitoring page to quickly locate problems.

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 and API Connect (APIC) triggers only.

Tracing

You can enable tracing for functions. Then you can go to the Application Performance Management (APM) console to view JVM and tracing information. Currently, only Java functions can be traced.

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 Function Selection

6.1 Function Type Selection

This section describes the scenarios and differences of function types supported by FunctionGraph.

Suggestion

FunctionGraph allows you to create event and HTTP functions by compiling code online, uploading code files, or using container images. And it supports GPU computing resources.

When using FunctionGraph, you can select a proper function type and runtime based on your service scenarios and technology stack preferences. For details about function type selection in common scenarios, see **Table 6-1**.

Table 6-1 Function type selection suggestions

Scenarios	Suggestion	Description
Web applications and API services	Creating an **HTTP function** by compiling code	HTTP functions support mainstream web application frameworks and can be accessed using browsers or directly invoked using URLs.
File processing and data stream processing	Creating an **event function** with the **built-in runtime**	Event functions can integrate with other Huawei Cloud services, such as Object Storage Service (OBS), Distributed Message Service (DMS) for RabbitMQ, and Log Tank Service (LTS).
Model inference scenarios such as chatbot and text-to-image generation	Creating an **HTTP function** based on a custom image **with GPU enabled**	Use container images of popular AI projects (such as Stable Diffusion, ComfyUI, and Ollama) to create HTTP functions and enable GPUs to quickly build AI model inference services.

Scenarios	Suggestion	Description
Asynchronous tasks such as scheduled tasks and audio/video transcoding	Creating an **event function** with the **built-in runtime**	Event functions can be triggered by specific events or scheduled events.

Function Type Comparison

For details about event functions and HTTP functions, see **Table 6-2**.

Table 6-2 Function type comparison

Item	Event Function	HTTP Function
Feature	Processes files and data streams triggered by events of various cloud products, such as **EG trigger (OBS Application Service)**,**Kafka trigger**, and **LTS triggers**. You can use it to process asynchronous requests and trace and save the status of each asynchronous invocation.	Supports popular web application frameworks and AI projects, which can be accessed through browsers or invoked through URLs.
Scenarios	• Cloud product integration: OBS real-time file processing and LTS log processing. • ETL data processing: database data cleaning and message queue processing. • Regular tasks: scheduled, periodic, and script tasks. • Multimedia processing: audio/video transcoding, live recording, and image processing.	• Application building based on popular web frameworks such as Express and Flask. • AI model inference service building based on Stable Diffusion, ComfyUI, and Ollama. • Migration of existing applications, such as HTML5 websites, REST APIs, BFF, mobile apps, applets, and game settlement.
Runtime	You are advised to use a **built-in runtime**.	You are advised to use a **custom runtime** or **custom image**.

Function Running Environment Comparison

For details about the function running environment, see **Table 6-3**.

Table 6-3 Function running environment comparison

Item	Built-in Runtime	Custom Runtime	Custom Image
Development process	Write a request handling program based on the handler defined in FunctionGraph.	Develop applications based on the web application framework template and view the result through the public network access.	Upload a custom image to SWR and use the image, or use an existing image in SWR.
Supported instance types	CPU instance	CPU and GPU instances	CPU and GPU instances
Instance concurrency	Not supported	Supported	Supported
Cold start	Fastest The code package does not contain the runtime.	Fast The code package is an HTTP server program and is large in size. No container image needs to be pulled.	Slow The image needs to be pulled.
Code file size limit	The code package size cannot exceed 100 MB or 500 MB. For details, see the **Notes and Constraints**.		The size of an undecompressed image cannot exceed 10 GB.
Code file format	ZIP, JAR (Java)		Container images
Language	Node.js, Python, PHP, Java, C#, Go	Unlimited	Unlimited

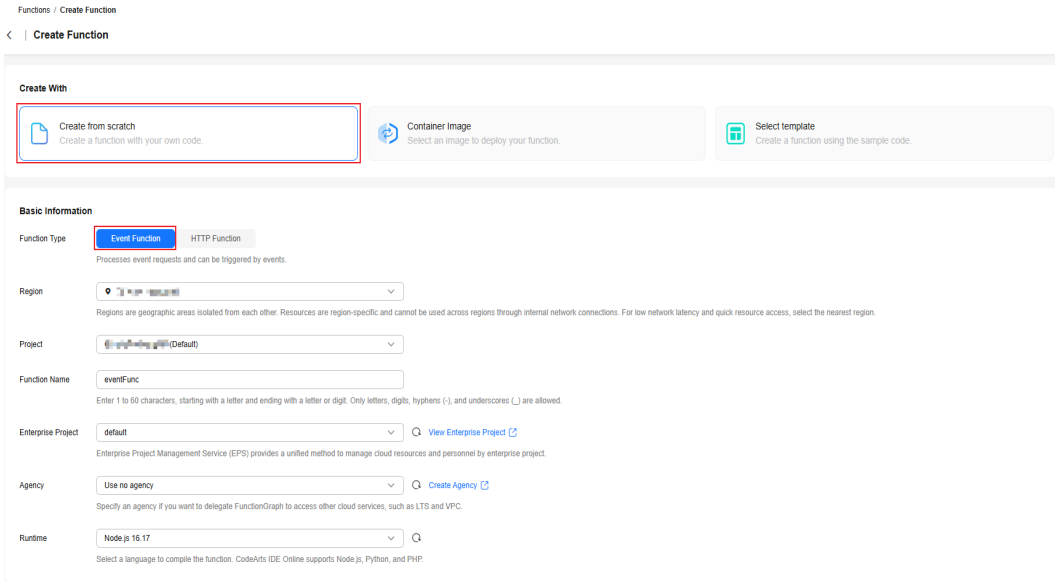
Creating a Function on the FunctionGraph Console

You can create the following functions on the FunctionGraph console:

Event Functions

If you want to invoke an associated function through a specific event or a timer trigger, create an event function on the FunctionGraph console, as shown in **Figure 6-1**. You are advised to select a **built-in runtime**. For details, see **Creating an Event Function**.

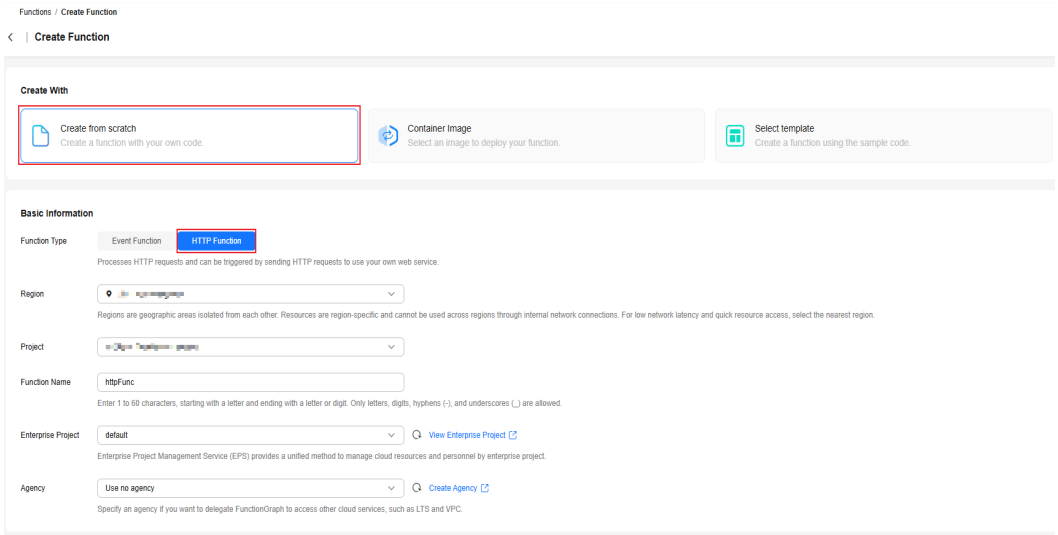
Figure 6-1 Create an event function



HTTP Functions

If you want to compile programs based on popular frameworks in various languages, you can create HTTP functions on the FunctionGraph console, as shown in **Figure 6-2**. You are advised to select a **custom runtime**. For details, see **Creating an HTTP Function**.

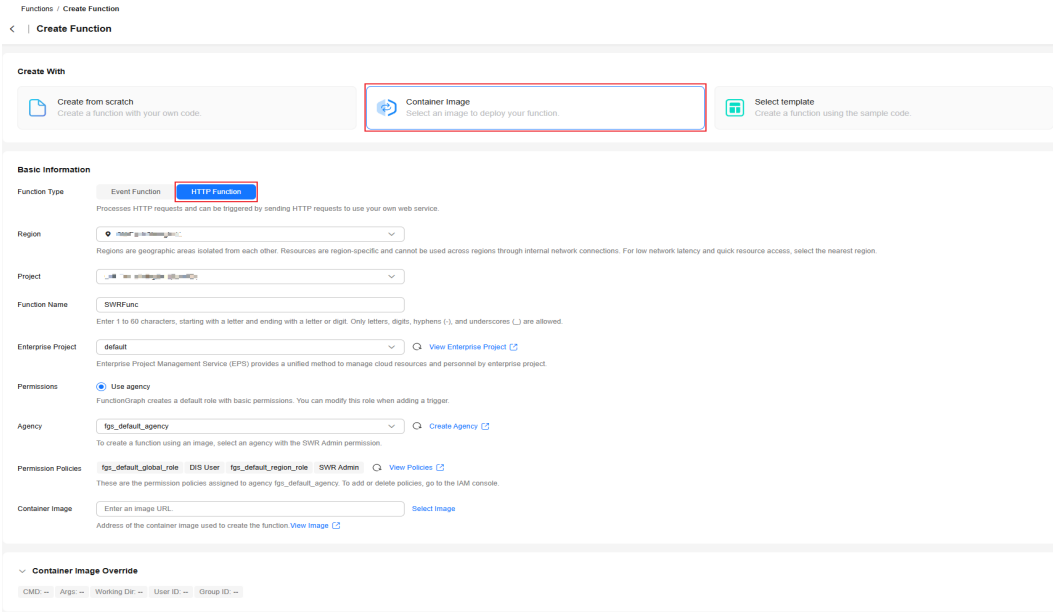
Figure 6-2 Creating an HTTP function



GPU Functions

If you want to use a container image of a popular AI project (such as Stable Diffusion WebUI, ComfyUI, and Ollama), you can create a GPU-accelerated and **image-based HTTP function** on the FunctionGraph console, as shown in **Figure 6-3**.

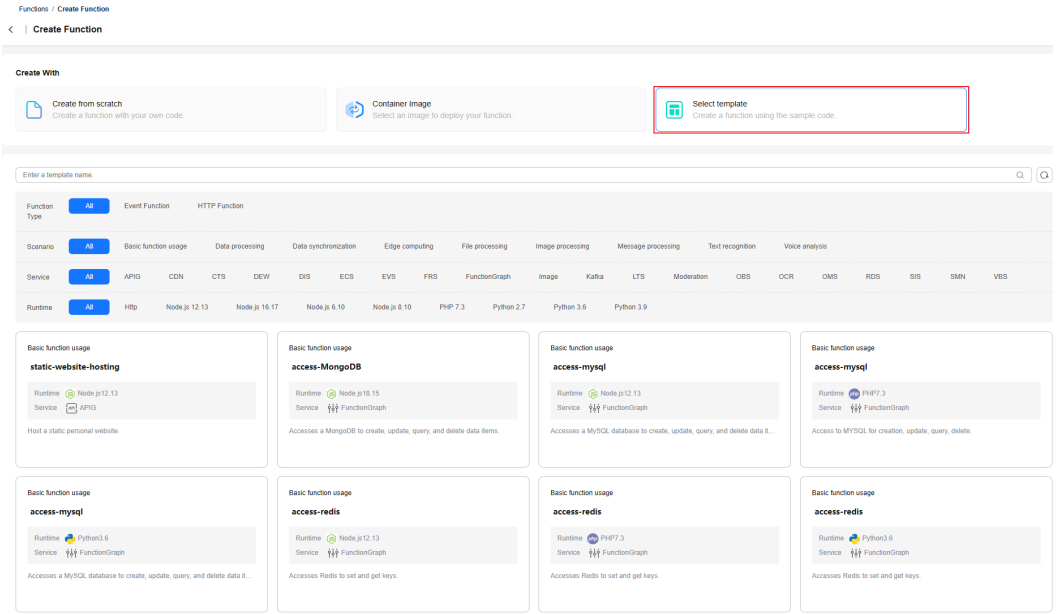
Figure 6-3 Creating a GPU function



Template Function

FunctionGraph provides function templates for various scenarios. With these templates, you can quickly build a function application with the code and environment variables automatically filled, as can be seen in **Figure 6-4**. For details, see **Creating a Function Using a Template**.

Figure 6-4 Creating a function using a template



6.2 Function Storage Selection

This section describes the application scenarios and differences of the storage types supported by FunctionGraph.

Function Storage Comparison

FunctionGraph provides various storage types, including Elastic Cloud Server (ECS), Scalable File Service (SFS) Turbo, Object Storage Service (OBS), Ephemeral Storage, and Dependency. For details about the storage comparison, see **Table 6-4**.

Table 6-4 Function storage comparison

Item	ECS	SFS Turbo	OBS	Ephemeral Storage	Dependency
Scenarios	Log and service file storage	Log and service file storage	Log and service file storage	Ephemeral service file storage	Release and deployment of public dependency libraries, runtime environments, and function extensions
Maximum space	Expandable	Scalable	Scalable	The default value is 512 MB. You can change the value to a maximum of 10 GB.	300 MB
Persistence	Persistent	Persistent	Persistent	Temporary	Persistent
Invocation sharing	Supported	Supported	Supported	Not supported	Supported
Storage content	Writable	Writable	Writable	Writable	Unwritable
Storage type	File system	File system	Object	File system	Code dependency archiving
Event source integration	Not supported	Not supported	Supported	Not supported	Not supported

Item	ECS	SFS Turbo	OBS	Ephemeral Storage	Dependency
Function access speed	Fast	Fast	Faster	Fastest	Faster
Billing	**ECS Billing Overview**	**SFS Turbo Billing Overview**	**OBS Billing Overview**	Free of charge when the disk size is less than or equal to 512 MB. For details, see **Billing Item**.	None

Storage Type Description

FunctionGraph supports the following storage types:

ECS

ECS provides secure, high-performance, reliable, and easy-to-use file storage services by enabling the NFS file service.

FunctionGraph can seamlessly integrate with ECS, allowing functions to mount ECS shared paths. Once configured, functions can access the specified ECS shared paths the same way as accessing a local file system.

Advantages of using ECS for function mounting include:

- Ephemeral files can be stored in ECS shared paths. Their size will not be limited by instance local disk space.
- Multiple functions can share the same ECS shared path.

For details, see **Configuring Disk Mounting**.

SFS Turbo

SFS Turbo provides scalable, high-performance shared file storage (NAS) for shared file access from ECSs, Bare Metal Servers (BMSs), Cloud Container Engine (CCE) containers, and Cloud Container Instance (CCI) containers.

FunctionGraph supports seamless integration with SFS Turbo, allowing functions to mount the SFS Turbo file system. After configuration, functions can access the specified file system the same way as accessing a local file system.

The advantages of using SFS Turbo for function mounting are as follows:

- Ephemeral files can be stored in the SFS Turbo file system. Their size will not be limited by instance local disk space.

- Multiple functions can share an SFS Turbo file system.

For details, see **Configuring Disk Mounting**.

OBS

OBS provides massive, secure, and cost-effective data storage for you to store data of any type and size. It is suitable for various scenarios, such as enterprise-level backup/archiving, video on demand (VoD), and video monitoring.

FunctionGraph integrates with OBS via EventGrid (EG) triggers. You can customize functions to process OBS events. For example, you can invoke functions to process image or audio data and save results to various storage services. You only need to focus on writing function logic. The system will process massive data concurrently in real-time.

For details, see **Creating an EG Trigger (OBS Application Service)**. In addition, you can use the SDK tool of the corresponding language to implement OBS read/write operation in your function code.

Ephemeral Storage

FunctionGraph provides two ephemeral storage specifications: 512 MB and 10 GB.

The lifecycle of the ephemeral storage is the same as that of the underlying function instance. If there are continuous requests, the instance remains. Therefore, the data previously stored on the disk is retained. If the function does not receive requests for a long time, the system will recycle the instance and the data on the disk will be lost.

Dependency

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 details about the operation and restrictions of function dependencies, see **Configuring Dependency Packages**.

7 Function Instance Types and Usage Modes

This section describes the CPU and GPU instance types, billing modes, and specifications.

Instance Type

Function instances are classified into the following types:

- CPU instances: Basic function workflow instances, which are suitable for burst traffic and compute-intensive scenarios.
- GPU instances: GPU instances based on the Turing architecture, which are suitable for audio/video, AI, and image processing scenarios. Different service loads are accelerated by GPU hardware to improve processing efficiency.

  GPU instances can be deployed only using container images or custom runtimes.

Usage Modes

CPU and GPU instances are available in two types: on-demand and reserved. The two modes are described as follows:

On-demand Mode

In on-demand mode, instances are automatically scaled by FunctionGraph based on invocations.

Instances are created by requests and destroyed after 1 minute of inactivity. Initial invocations involve a cold start.

Constraints:

By default, a single Huawei Cloud account (IAM account) can have a maximum of 1,000 instances in a region. If you need more instances, **submit a service ticket**.

Billing mode:

In on-demand mode, billing begins when the function is triggered by the request and ends when the request is completed. A single instance can process a single

request or multiple requests based on the configuration. For details about the execution duration, see **Table 7-1**. For details about how to configure concurrent processing, see **Configuring Single-Instance Multi-Concurrency**.

When no function is invoked, the system does not allocate compute resources and no fees are generated. You will be billed only when the function is invoked and executed. For details about service billing, see **FunctionGraph Billing Overview**.

Table 7-1 Execution duration description

Request Mode	Duration	Example
Single-instance single-concurrency	The execution duration starts from the time when the request arrives at the instance and ends when the request is completed.	• If the request arrives at 00:00:00 and ends at 00:00:05, the billing duration is 5 seconds. • If three requests arrive at the same time and each takes 5 seconds, the total duration is 3 x 5=15 seconds.
Single-instance multi-concurrency	The execution duration starts from the time when the first request arrives at the instance and ends when the last request is completed.	If the first request arrives at 00:00:00 and ends at 00:00:05, and the last request arrives at 00:00:03 and ends at 00:00:08. They are executed in the same instance and the total duration is 8 seconds.

Reserved Mode

In the reserved mode, you can manage the lifecycle of function instances to achieve flexible control over computing resources. When reserved instances are configured for a function, FunctionGraph prioritizes scheduling incoming requests to these instances. If traffic exceeds the capacity of reserved resources, FunctionGraph automatically scales resources to dynamically allocate on-demand instances, ensuring service continuity.

After reserved instances are created for a function, the code, dependencies, and initializer of the function are automatically loaded. Reserved instances are always alive in the execution environment, eliminating the influence of cold starts on your services. You are advised to configure **a fixed number of reserved instances** based on the service requirements, **scheduled scaling**, **metric scaling**, and **intelligent recommendation** policies based on traffic peaks and troughs.

Note:

To ensure service stability and reliability, do not rely on the initializer of the reserved instance to execute one-time services.

Billing mode:

For details, see the execution duration (reserved instances) billing item in **Billing Items** and **Reserved Instance Billing**.

Specifications

- CPU instance

  CPU instances include the following specifications.

  Table 7-2 CPU instance specifications

Memory	Max. Code Package Size	Max. Execution Duration	Max. Disk Size
128–32,768 MB The value must be a multiple of 64.	• ZIP file: 1.5 GB (after decompression) • OBS bucket: 300 MB (after compression)	259,200s If the execution takes longer than 900 seconds, use **asynchronous invocation**.	The value can be 512 MB (default) or 10 GB.

- GPU instance

  GPU instances have the following specifications.

  Table 7-3 GPU instance specifications

GPU	Total Graphics Memory	Compute (TFLOPS)		Optional Split Specifications		On-demand Mode	Reserved Mode	Idle Mode
NVIDIA T4	16 GB	**FP16**	**FP32**	**Graphics Memory (MB)**	**Memory (MB)**	Supported	Supported	Supported
		65	8	1024–16384 (1–16 GB) The value must be a multiple of 1024 MB.	128–32768 The value must be a multiple of 64.			

8 Notes and Constraints

Supported Regions

For regions supported by FunctionGraph, see **Regions and Endpoints**.

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**.

Restriction Item	Description
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.
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	• 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	• 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

Restriction Item	Description
Private dependency	• ZIP: 10 MB (If the file size exceeds 10 MB, upload it via OBS.) • OBS: OBS URL with the file in ZIP format

Function Flow

Function flow is available in **CN East-Shanghai1** and **AP-Singapore**.

Table 8-3 Function flow restrictions

Restriction Item	Description
Max. function flows per account	200 For more function flows, **submit a service ticket**.
Max. nodes per function flow	100 For more function flow nodes, **submit a service ticket**.
Standard flows	For common services that take a long time to execute. Standard flows can only be invoked asynchronously.
Express flows	Better for services that take less than 5 minutes to execute. Express flows can be invoked synchronously or asynchronously, but their execution records are not persisted.

Function Running Resources

Table 8-4 Function running resource restrictions

Restriction Item	Description
Ephemeral disk space (**/tmp** space)	512 MB
File descriptors	2048
Total number of processes and threads	1024

Restriction Item	Description
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 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.

User Identity Credential

Users use identity credentials, such as IAM AK/SK and tokens to access cloud services. If these credentials are disclosed, service security cannot be ensured. You

need to use IAM to grant permissions based on the principle of least privilege to reduce the impact of attacks when identity credentials are disclosed.

User Function Code

User code and private dependencies are core assets. You need to ensure code reliability and security, avoid embedding sensitive information (such as account, password, AK/SK, token) in the code, and prevent logging sensitive information to avoid leakage.

User Function Configuration

- **Encrypted environment variables**

  Sensitive information in user code, such as the AK/SK for accessing other cloud services and the password for accessing the database, can be transferred through **encrypted environment variables**. FunctionGraph encrypts and stores the encrypted environment variables to prevent sensitive information leakage.

  Figure 9-2 Encrypting environment variables

- **Function public access configuration**

  Public access: By default, a function can access the public network. All tenants share the bandwidth, which may cause external network attacks. You can **configure a VPC** to access the public network through the VPC and exclusively use the network bandwidth.

  VPC access: To access cloud resources in a VPC, such as databases and cache services, you are advised to configure a VPC to prevent sensitive information leakage.

- **Minimal permissions**

  You need to **configure an agency** and permissions (for accessing other Huawei Cloud services, such as ECS and OBS) compliant with the **minimum permissions** to reduce security risks caused by authorization token leakage.

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 White Paper**.

- **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	Scans FunctionGraph code from multiple dimensions, covering code style, quality, and security issues. Its core capabilities include: • 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 *Huawei C/C++ Coding Standard*) 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.
SecMaster	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. SecMaster analyzes logs of related FunctionGraph cloud services (such as OBS and VPC) to detect malicious behavior in real time. • **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**: 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():
 # Initialize the KMS client.
 kms_client = KmsClient(
 secret_id=os.getenv('KMS_SECRET_ID'),
 secret_key=os.getenv('KMS_SECRET_KEY'),
 region_name="cn-north-4"
 )

# Decrypt data.
 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.

Technology	Description
Others	• 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 **Enabling CTS**.

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 **Configuring a Function Version**.

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.

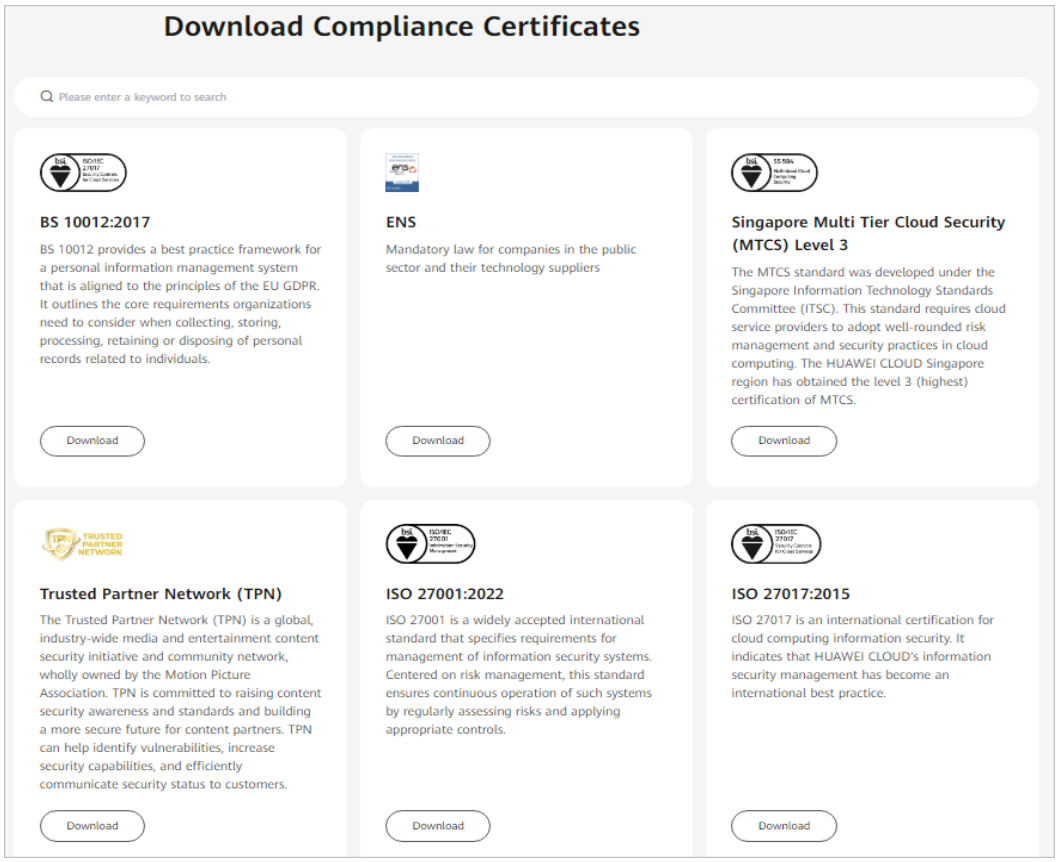
For details about FunctionGraph metrics, see **Monitoring**. To learn how to create alarm rules, see **Creating an Alarm Rule**.

9.8 Certificates

Compliance Certificates

Huawei Cloud services and platforms have obtained various security and compliance certifications from authoritative organizations, such as International Organization for Standardization (ISO). You can **download** them from the console.

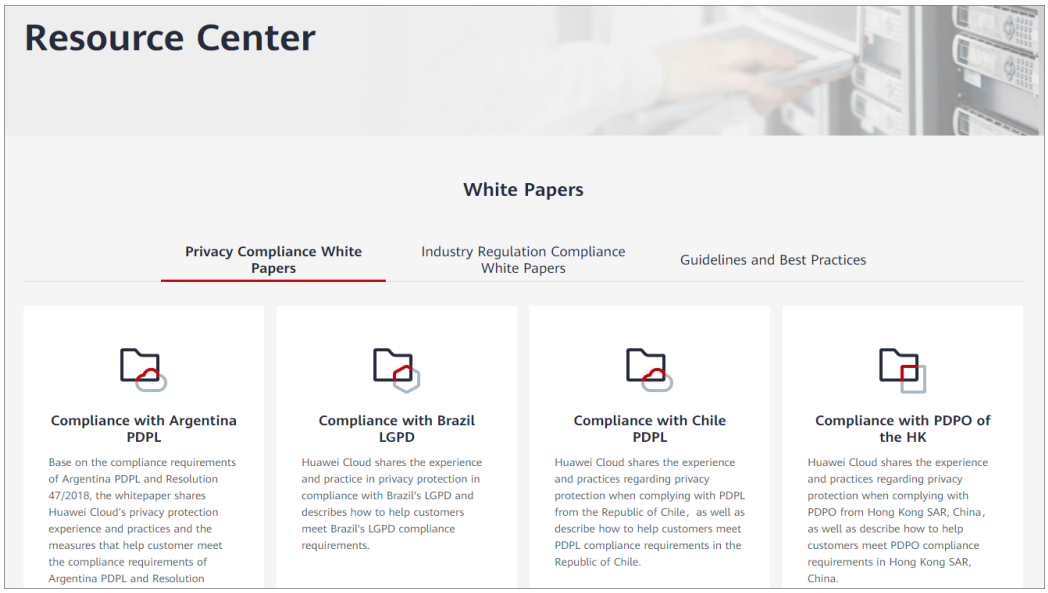
Figure 9-3 Downloading compliance certificates



Resource Center

Huawei Cloud also provides the following resources to help users meet compliance requirements. For details, see **Resource Center**.

Figure 9-4 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.

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, DIS, 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	apig:groups:get apig:groups:list apig:apis:create apig:apis:delete apig:apis:update apig:apis:publish apig:apis:list apig:apis:get apig:apis:offline apig:apps:list apig:envs:list

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
DDS	dds:instance:get dds:instance:list
DIS	dis:streams: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

Trigger/Function	Permission
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
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 ReadOnlyAccess	FunctionGraph CommonOperations	FunctionGraph FullAccess
Creating functions	×	√	×	×	√
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.

For details, see **Function Runtimes**.

Synchronous Invocation

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

For details, see **Invoking a Function**.

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.

For details, see **Invoking a Function**.

Trigger

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

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

Function Flow

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

For details, see **Flow Management**.

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 **Configuring Dependency Packages**.

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. For details, see **DMS User Guide**.
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 **DIS User Guide**.

Service	Description
CTS	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. • 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	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**. • For details about function metrics supported by Cloud Eye, see **Monitoring Configuration**.
VPC	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**.
AOM	Monitor function information in graphics. For details, see **Viewing FunctionGraph Metrics**.