Serverless Cloud Function
Product Introduction

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Product Introduction
Overview

Last updated: 2024-06-14 14:56:18

Tencent Cloud's Serverless Cloud Function (SCF) is a serverless execution environment provided by Tencent Cloud for businesses and developers. It enables you to run code without the need to purchase and manage servers, making it an ideal computing platform for real-time file processing and data processing scenarios. All you need to do is write your core code in a language supported by the SCF platform and set the conditions for the code to run. This allows for flexible and secure code execution on Tencent Cloud's infrastructure. The following video will introduce you to SCF:
Watch video

Evolution of Computing Resources

With the development of cloud services and the high abstraction of computing resources, Tencent Cloud provides a wide variety of computing resource options from physical servers to cloud-based functions at different abstraction levels.

- Cloud Physical Machine (CPM): scaling is at the physical machine level. You enjoy the physical computing resources exclusively, which provides the best security.
- Cloud Virtual Machine: The unit of expansion is the cloud server, which virtualizes hardware devices. Users share physical machine resources with other tenants, but can still configure various metrics of the CVM independently, making deployment and iteration simpler.
- Tencent Kubernetes Engine (TKE): it virtualizes operating systems, and scaling is at the service level. The test and production environments are exactly the same, making testing and deployment very convenient.
- Serverless Cloud Function (SCF): it virtualizes runtime environments, and scaling is at the function level. This is the smallest unit of existing computing resources, which features full automation, one-click deployment, and high scalability, and is an ideal choice for lightweight service deployment.

Serverless

Serverless does not mean that there is no server. It is just that you do not need to care about the underlying resources, log in to a server, or optimize the server. You only need to care about the core code snippets while skipping the complex and cumbersome basic work. These code snippets are completely triggered by events or requests, and the platform automatically adjusts service resources in parallel based on the requests. The serverless architecture has

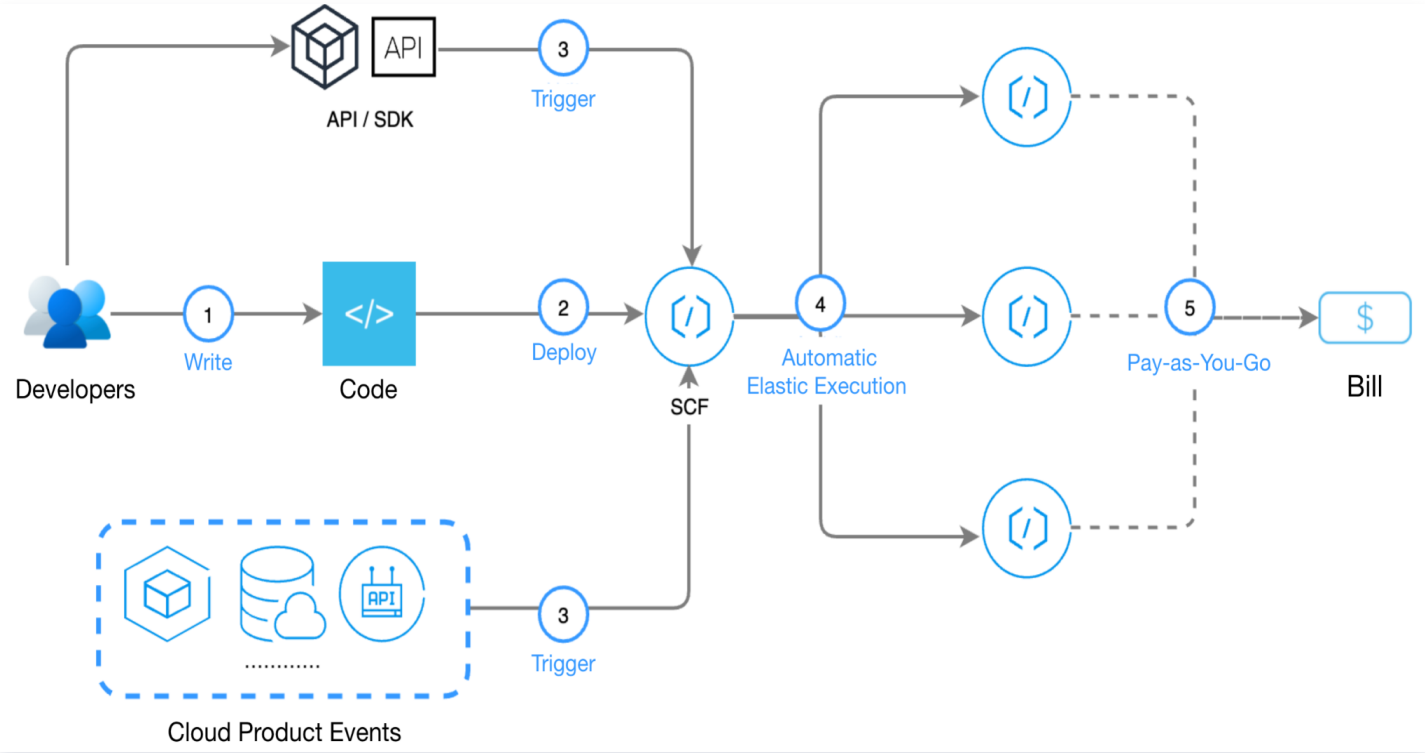
near-infinite scalability, and no resources will be executed when idle. The code is executed in a stateless manner, which easily enables fast iterations and rapid deployment.

SCF Overview

SCF is a serverless execution environment provided by Tencent Cloud. All you need to do is write simple and single-purpose functions and associate them with events generated by your Tencent Cloud infrastructure and services.
When using SCF, all you need to do is write the code in a programming language (Python, Node.js, PHP, Go, Java, and Custom Runtime) supported by the platform. The underlying computing resources and tasks are fully managed by Tencent Cloud, including maintenance of server CPUs, memories, networks, and other configurations/resources, code deployment, elastic scaling, load balancing, security upgrade, and resource execution monitoring. However, this means that you cannot log in to or manage servers or customize the system or environment.
SCF is automatically deployed in multiple AZs in the same region with extremely high fault tolerance achieved. When a function is executed, scaling will be made based on the request load with no manual configuration or intervention required, helping meet the needs for service availability and stability in different scenarios. From several requests per day to thousands of requests per second, SCF can automatically scale at the underlying layer. You only need to pay for running functions, and if no function is running, no fees will be incurred.
You can customize the timing of running a function, such as when a file is uploaded to a COS bucket, when a file is deleted, when a message in CKafka is used, or when an application is called through an SDK. You can also configure the function to run regularly. Therefore, SCF can be used as a data processing trigger for the COS service to easily implement IFTTT logic. Scheduled automated tasks can also be flexibly built to free you from manual operations and easily construct an elastic and controllable software architecture.

SCF Features

Serverless helps you get rid of the tedious development and configuration work, so that you can only care about the writing of business code logic, without any infrastructure construction, management, and OPS overheads. This service model lowers the threshold of R&D and improves the efficiency of business construction; therefore, it has gained the recognition of high numbers of enterprises and developers.



Various development tools and languages supported for smooth development

The Tencent Cloud Serverless team works in many ways to provide convenient tools or capabilities that can meet the needs in a wide variety of development scenarios; for example:

- Utilize the Serverless Cloud Framework to create projects, debug and package locally, and deploy online with a single click, all within your local development environment.
- With the aid of visual operations based on the VS Code plugin and IDE, online/offline management of functions and code writing and debugging can be performed at one single place. The VS Code IDE and plugin also enables local management, development, and debugging as well as online release of functions.
- Supports Web IDE, allowing for real-time development and debugging on the console, providing an experience similar to local development and debugging, making code adjustments or viewing more convenient.
- SCF supports Python, Node.js, Go, PHP, Java, and Custom Runtime, so you can select a custom runtime environment as needed.

Multiple deployment methods for various environments

Supports deployment via the console, command line, SDK/API, direct deployment through Web IDE, and image deployment.

Diversified triggers to support more business scenarios

Trigger methods include APIs, SDKs, and events from various other cloud service products such as COS, API Gateway, etc. A variety of trigger options support a wider range of use cases.

Automated and flexible execution for better invocations

SCF can automatically scale according to the call volume, which is imperceptible to users, perfectly fits the invocation curve, and saves resources and costs to the greatest extent.

Pay-as-You-Go billing accurate to the millisecond level

SCF supports billing actually used resources at a time granularity of 1 ms, which can significantly reduce your costs compared to the time granularity of 100 ms.

Relevant Concepts

Last updated: 2023-09-27 14:21:59

Tencent Cloud's Serverless Cloud Function (SCF) is a Function as a Service (FaaS) product, offering a computing platform that is both serverless and FaaS-based. Its operation is contingent upon event triggering. Consequently, when integrated with an event source, the SCF can be activated by events generated by the trigger source. The following video will provide an overview of the SCF concept:
Watch video

Serverless

The origin of the term 'serverless architecture' can be elucidated from the article 'Serverless Architecture' authored by Mike Roberts and published on Martin Fowler's blog site.
Serverless does not imply computation without servers, but rather, it signifies that developers can utilize relevant resources without needing to understand the underlying server conditions, hence the term 'serverless'.
Serverless can also be recognized from a broader perspective. Cloud services that can be directly used without the need for configuration or understanding of the underlying servers can, to a certain extent, also be referred to as serverless.
Within the SCF product, we cater to the computational scenarios in serverless contexts. The SCF product offers FaaS capabilities in a serverless mode.

Function as a Service

Function as a Service provides the capability to run stateless, ephemeral code triggered by events directly on the cloud.
Function as a Service differs from traditional application architectures. It offers an event-triggered execution model. Cloud functions are not always running; instead, they are triggered by an event and process that event during a single run. Therefore, in the cloud function code, you only need to consider the processing flow for a single event. The platform supports high concurrency processing for a large number of events by implementing multiple instances of cloud functions concurrently.
To support high concurrency, the cloud function platform provides automatic elastic scaling capabilities. It will launch more instances to handle event requests when a large number of requests arrive, and it will reduce the number of function instances to zero when no events are incoming. Therefore, to match the automatic scaling capability, the function code needs to use a stateless development method, i.e., it does not retain related state data in the running memory of the cloud function and rely on these state data for multiple runs. The state data of

the cloud function can rely on external persistent storage capabilities such as cloud cache, cloud database, and cloud storage.

Triggers and Trigger Sources

Anything that can generate events and trigger the execution of cloud functions can be referred to as a trigger or trigger source. After generating an event, the trigger initiates the function execution by passing the event to the cloud function.
When triggering a function, the trigger can use either synchronous or asynchronous methods based on its characteristics. In synchronous triggering, the trigger waits for the function execution to complete and obtains the execution result. In asynchronous triggering, the trigger merely initiates the function and disregards the execution result.
When Tencent Cloud Functions interface with certain Tencent Cloud products or services, they also have some unique implementation methods, such as the Push mode and Pull mode.

- Push Mode: The trigger actively pushes the event to the cloud function platform and initiates the function execution.
- Pull Mode: The cloud function platform triggers the function execution by pulling events from the trigger through the pull module.

Triggering event

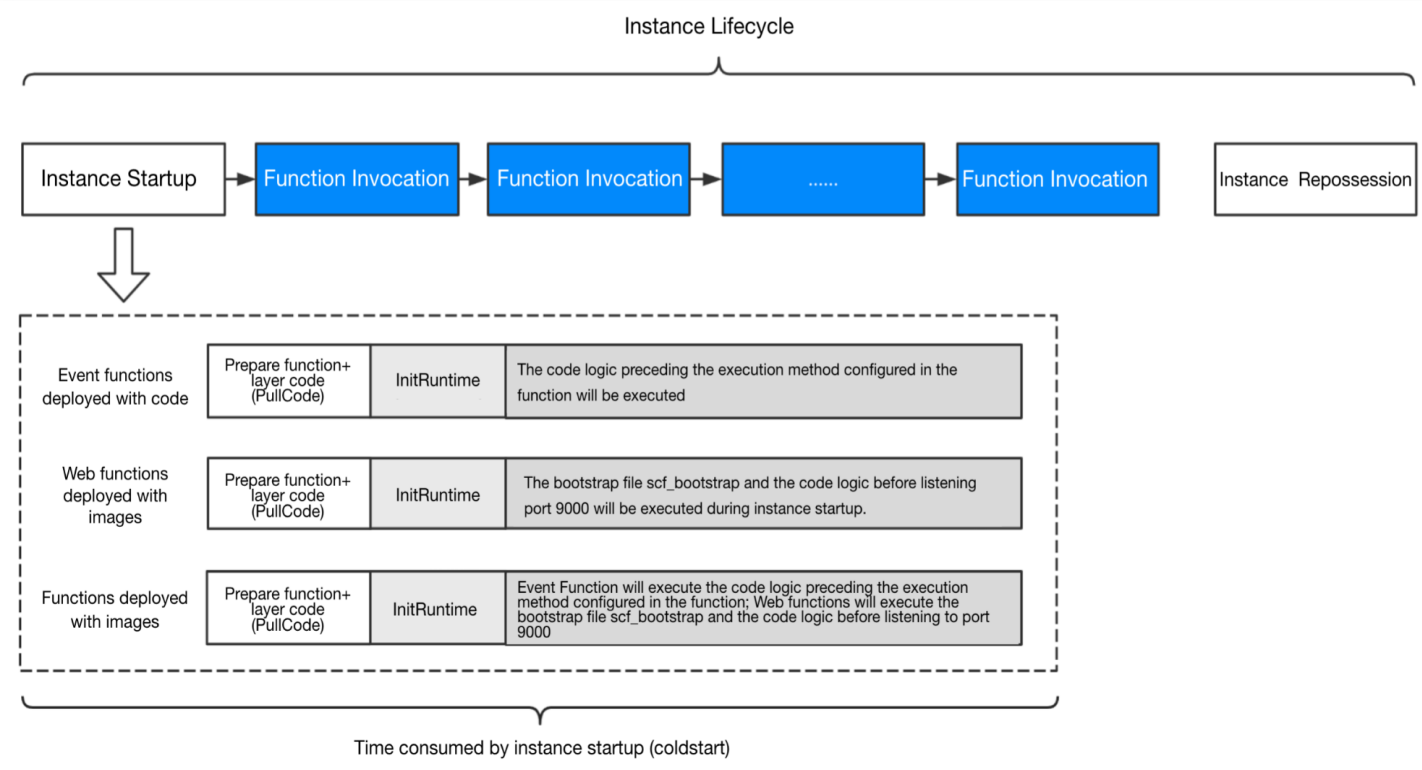
When a trigger initiates a function, it passes the event to the cloud function. The event is represented by a specific data structure during transmission, which is always in JSON format, and is passed to the cloud function as an event parameter.
The JSON data content of the triggered event will be converted into the data structure or object of the respective language environment, eliminating the need for manual conversion from JSON structure to data structure in the code.
For instance, in a Python environment, the JSON data content will be transformed into a complex dict object, meaning the function's parameter event is a complex Python dict object. In Golang or Java, the parameter is an object that can match the event data structure. More specific implementation methods can be found in Development Language Description.

How It Works

Last updated: 2024-06-14 14:56:37

Instance Model of Function Runtime

SCF will execute a function for you when the function receives a triggering request. Instance is the resource for SCF to execute the request. SCF will allocate resources based on the function configuration information (such as memory size) and launch one or multiple instances to process the function request. The SCF platform is responsible for the creation, management, and deletion of all function runtime instances, and you have no permissions to manage them.
The lifecycle of an instance is depicted as follows:



Starting instance

- If there is no running instance when a request arrives, the request will trigger the startup of an instance. Instance startup usually takes some time, which adds extra time to the invocation that triggers the instance startup. Generally, instance startup is triggered only when a function is invoked for the first time, updated, or invoked again after a long period of inactivity.
- The time consumed by instance startup will be reflected in the `Coldstart` field in the

Execution Log `Init Report` or `Provisioned Report` line of the function.

- You can use the provisioned concurrency feature to start instances in advance to avoid triggering the instance startup when the function request arrives.
- The instance startup time is limited by the function's initialization timeout period. If the former is longer than the latter, instance startup will fail. You can accelerate instance startup or increase the initialization timeout period accordingly as detailed below.

Method to accelerate the three phases of instance startup:

- **Code Preparation**: The platform retrieves the function code, layers, or images uploaded by the user to prepare for function execution. The time consumed in code preparation is directly proportional to the size of the code package, layers, or images. It is recommended to streamline the code package as much as possible, retaining only the necessary code files and dependencies for function execution, to minimize code preparation time. The time consumed in code preparation will be reflected in the `PullCode` field in the Execution Log `Init Report` or `Provisioned Report` line of the function.
- **Runtime Initialization**: The platform prepares the runtime environment required for function execution based on the user's function configuration. The time consumed in runtime initialization will be reflected in the `InitRuntime` field in the Execution Log `Init Report` or `Provisioned Report` line of the function.
- **Function Code Initialization**: The time consumed in code initialization is directly proportional to the complexity of the code logic. It is recommended to optimize the code logic as much as possible to minimize code initialization time. The time consumed in function code initialization will be reflected in the `InitFunction` field in the Execution Log `Init Report` or `Provisioned Report` line of the function.
  - For event functions with code deployment: During the instance startup phase, the code logic preceding the execution method configured in the function will be executed (for example, if the entry function is main_handler, all code logic before main_handler will be executed).
  - Code deployment–based HTTP–triggered function: the bootstrap file `scf_bootstrap` and the code logic before listening port 9000 will be executed during instance startup.
  - For functions deployed with images: Event–triggered functions will execute the code logic preceding the execution method configured in the function; HTTP–triggered functions will execute the bootstrap file `scf_bootstrap` and the code logic before listening to port 9000.

Instance reuse

In order to minimize the additional time caused by instance startup, the platform will try to reuse the instance for subsequent invocations. After the instance processes the function

request, it will be stored for a period of time according to the actual situation of the platform for next invocations and will be used first during this period.
The meaning of instance reuse is as follows:

- All declarations outside the execution method part in your code remain initialized and can be reused directly when the function is invoked again. For example, if a database connection is established in your function code, the original connection can be used directly when the container is reused. You can add logic to your code to check whether a connection already exists before creating a new one.
- Each container provides some disk space in the `/tmp` directory. The contents of this directory are retained when the container is retained, providing a temporary cache that can be used for multiple invocations. It is possible to use the contents of the disk directly when the function is invoked again. You can add extra code to check whether such data is in the cache.

> **Note**
> Do not assume that the instance is always reused in the function code, because reuse is related to the single actual invocation, and it cannot be guaranteed whether a new instance will be created or an existing one will be reused.

Repossessing instance

The platform will repossess instances that have not processed requests for a certain period of time.

Temporary Disk Space

During the execution of SCF functions, there is a 512MB temporary disk space `/tmp` available. Users can perform some read and write operations in this space during code execution, and can also create subdirectories. However, this data may **not** be retained after the function execution is completed. Therefore, if you need to persistently store the data generated during execution, please use Object Storage COS or external persistent storage such as Redis/Memcached.

Call Types

The SCF platform supports both sync and async calls of functions.

Sync invocation

A synchronous function call will continue to wait for the return of the execution result after the call request is issued.

Async invocation

- Async call will only send the request and get the request ID of the current request, but not wait for the result.
- When an async invocation occurs, the async event will be placed in the async queue built in SCF and then consumed by the event execution function in the async queue. Async queues have the following restrictions:
- Async queues are at the trigger level, and one function trigger has one queue.
- An async event can be retained in a queue for up to 6 hours.
- There can be up to 100,000 messages in an async queue.
- The retry policy may vary by async queue. For more information, please see Retry Policy.

Defining function invocation type

The call type is independent of the configuration of the function itself and can only be controlled when the function is called.
In the following call scenarios, you can freely define the call type of the function:

- Your application calls the SCF function. If you need a synchronous call, use the synchronous call interface InvokeFunction; if you need an asynchronous call, use the Invoke interface and pass in the parameter `invokeType=Event`.
- The SCF function is manually invoked (with API or CLI) for testing. The parameters for the invocation are the same as above.

When you use other Tencent Cloud services as event sources, the call type of the cloud service is predefined:

- Sync invocation: by API Gateway trigger, CLB trigger, and CKafka trigger, for example.
- Async invocation: such as COS trigger, Timer trigger, CMQ Topic trigger, etc. For more details, please refer to Trigger Overview.

Usage Restrictions

For quotas and environmental restrictions related to function usage, please refer to Quotas and Limitations.

Function Concurrency

The function concurrency is the number of executions of the function code in any given period of time. For the current SCF function, the request is executed once each time an event is published. Therefore, the number of events (i.e., requests) published by the trigger affects the function concurrency. You can use the formula below to estimate the total number of concurrent function instances.

```
Requests per second * function execution duration (in seconds)
```

For instance, consider a function that processes COS events. Assuming the function takes an average of 0.2 seconds (i.e., 200 milliseconds) to execute, and COS publishes 300 requests per second to the function. This would concurrently generate 300 * 0.2 = 60 function instances.

Concurrency Restrictions

By default, SCF imposes certain concurrency limits on each function. You can view the Concurrency Management to understand the current concurrency limit of the function.
If an invocation causes the function's concurrency to exceed the default limit, the invocation will be blocked and SCF will not execute it. Depending on the function's invocation method, the handling of restricted invocations may vary:

- Sync invocation: if the function is restricted when invoked synchronously, a 432 error will be returned directly.
- Async invocation: if the function is restricted when invoked asynchronously, SCF will retry the restricted event according to a certain policy.

Execution Environment and Available Libraries

The current SCF execution environment is built based on the following:

- Standard CentOS 7.2

If you need to include executable binary files, dynamic libraries, or static libraries in your code, ensure they are compatible with this execution environment. Depending on the language environment, there are basic libraries and additional installed libraries under the SCF execution environment. You can view the additional libraries installed in the environment in the language instructions:

- Python
- Node.js
- Golang
- PHP
- Java

Deployment mode

Serverless Cloud Function (SCF) provides two deployment methods of code deployment and image deployment and supports two function types of event-triggered function and HTTP-triggered function. Different deployment methods and function types require different specifications during code development. This document describes the writing specifications and related concepts of event-triggered function in code deployment. For more information on image deployment and HTTP-triggered function, please see the corresponding documents.

SCF event-triggered function

There are three basic concepts in SCF event-triggered functions: execution method, function input parameters, and function return. These concepts correspond to the following in typical project development:

- **Execution method**: corresponds to the main function of the project and is the starting point of program execution.
- **Function Input Parameters**: These are the typical function input parameters. However, in the SCF environment, the input parameters of the entry function are fixed by the platform. For more details, see Function Input Parameters.
- **Function return**: corresponds to the returned value of the main function in the project. After the function returns, the code execution ends.

Execution Method

When the SCF platform invokes a cloud function, it first seeks the execution method as the entry point to execute the user's code. At this point, users need to set it in the format of **filename.executionMethodName**.
For instance, if the user sets the execution method as `index.handler`, the SCF platform will first seek the `index` file in the code package and start executing the `handler` method found in that file.
Within the execution method, users can process the input parameters of the entry function and freely call other methods in the code. The execution of an SCF function ends when the entry function has finished executing or if there is an exception during execution.

Function input parameters

Function input parameters refer to the content passed to the function when it is triggered and invoked. Typically, function input parameters include two parts: **event** and **context**. However, depending on the development language and environment, the number of input parameters may vary. For more details, see Development Language Description.

event

Usage

The `event` parameter is of `dict` type and contains the basic information that triggers the function. It can be in a platform-defined or custom format. After the function is triggered, the event can be processed inside the code.

Use Instructions

There are two ways to trigger an SCF function:

1. Trigger by calling TencentCloud API.
2. Trigger by binding a trigger.

These two SCF trigger methods correspond to two event formats:

- **Triggering function execution via TencentCloud API:**
  A dict type parameter can be customized between the caller and the function code. The caller inputs data according to the defined format, and the function code retrieves data in the same format.
  Example:
  Define a dict type data structure {"key":"XXX"}. When the caller inputs data {"key":"abctest"}, the function code can obtain the value 'abctest' through event[key].
- **Triggering function execution via a trigger:**
  SCF is integrated with various cloud services such as API Gateway, COS, and Ckafka. Function execution can be triggered by binding the corresponding cloud service triggers to the function. When a trigger initiates a function, the event is passed to the function as an event parameter in a predefined, unchangeable format. You can write code based on this format and retrieve information from the event parameter.
  Example:
  When a function is triggered by COS, the specific information of the object storage bucket and file is passed to the event parameter in JSON format. By parsing the event information in the function code, the trigger event can be processed.

Context input parameters

Usage

`context` is an input parameter provided by the SCF platform. It is passed to the execution method, by parsing which the code can get the runtime environment and related information of the current request.

Use Instructions

The fields and descriptions of the `context` input parameter provided by SCF are as follows:

Field Name	Description
memory_limit_in_mb	Function Memory Configuration
time_limit_in_ms	Function Execution Timeout Period

request_id	Function Execution Request ID
environment	Function Namespace Information
environ	Function Namespace Information
function_version	Function Version Details
function_name	Function Name
namespace	Function Namespace Information
tencentcloud_region	Function's Region
tencentcloud_appid	Tencent Cloud Account's APPID for the Function
tencentcloud_uin	Tencent Cloud Account ID Associated with the Function

Note

To ensure compatibility, the context retains the description methods for namespaces at different stages of SCF. The structure of the context will expand with the development iterations of the SCF platform.

You can print the context information through the standard output statement in the function code. Take the `python` runtime environment as an example:

```
# -*- coding: utf8 -*-
import json
def main_handler (event, context):
    print (context)
    return ("Hello World")
```

The following context information can be obtained:

```
{"memory_limit_in_mb": 128, "time_limit_in_ms": 3000, "request_id": "f03dc946-
3df4-45a0-8e54-xxxxxxxxxxxx", "environment": "
{\"SCF_NAMESPACE\":\"default\"}", "environ":
"SCF_NAMESPACE=default;SCF_NAMESPACE=default", "function_version":
"$LATEST", "function_name": "hello-from-scf", "namespace": "default",
"tencentcloud_region": "ap-guangzhou", "tencentcloud_appid": "12xxxxx384",
"tencentcloud_uin": "10000xxxxx36"}
```

After understanding the basic usage of `event` and `context` input parameters, you should pay attention to the following points when writing function code:

- To ensure uniformity across various development languages and environments, both `event` and `context` input parameters are encapsulated using the JSON data format.
- Different triggers pass different data structures when triggering functions. For more information, please refer to Function Trigger Description.
- If the function does not need any input, you can ignore the `event` and `context` parameters in your code.

Function Return

The SCF platform will get the returned value after the function is executed and handle according to different trigger type as listed below.

Trigger Method	Handling Method
Synchronous Trigger	The function is triggered synchronously when invoked through the API Gateway or TencentCloud APIs. For functions triggered synchronously, the SCF platform will not return the trigger result during execution. Upon completion of the function execution, the SCF platform encapsulates the function's return value into a JSON format and returns it to the caller.
Asynchronous Trigger	For cloud functions triggered asynchronously, the SCF platform will return the trigger request ID upon receiving the trigger event. Upon completion of the function execution, the return value is encapsulated into a JSON format and stored in the logs. Upon completion of the function execution, users can retrieve the return value of the asynchronously triggered function by querying the logs with the returned request ID.

When the code in a function returns a specific value, it usually returns a specific data structure; for example:

Runtime Environment	Return Data Structure Type
Python	Simple or dict data structure
Node.js	JSON Object
PHP	Array structure

GO	Simple data structure or struct with JSON description

To ensure uniformity across various development languages and environments, the function return will be uniformly encapsulated in **JSON data format**. Upon receiving the return value of the function from the above runtime environment, the SCF platform will convert the returned data structure into JSON and return the JSON content to the caller.

> ⚠ **Note**
> - You should ensure that the returned value of the function can be converted to JSON format. If the object is returned directly and there is no JSON conversion method, SCF will fail when executing JSON conversion and prompt an error.
> - For example, the returned value in the above runtime environment does not need to be converted to JSON format before it is returned; otherwise, the output string will be converted again.

Exception Handling

If an exception occurs during testing and executing a function, the SCF platform will handle the exception as much as possible and write the exception information into the log. Exceptions generated by function execution include caught exceptions (handled errors) and uncaught exceptions (unhandled errors).

Solutions

You can log in to the SCF console and follow the steps below to test exception handling:

1. Create a function and copy the following function code without adding any triggers.
2. Click **Test** in the console and select the "Hello World" test sample for testing.

This document provides the following three ways to throw exceptions, and you can choose how to handle exceptions in the code based on your actual needs.

Throw exceptions explicitly

Example

```
def always_failed_handler (event,context):
    raise Exception ('I failed!')
```

Note

This function will raise an exception during its execution and return the following error message. The SCF platform will record this error message in the function log.

```
File "/var/user/index.py", line 2, in always_failed_handler
raise Exception ('I failed!')
Exception: I failed!
```

Inherit the `Exception` class

Example

```
class UserNameAlreadyExistsException (Exception): pass
def create_user (event):
    raise UserNameAlreadyExistsException ('
        The username already exists,
        please change a name!')
```

Explanation

You can define how to handle errors in your code to ensure the robustness and scalability of your application.

Use the `Try` statement to capture errors

Example

```
def create_user (event):
    try:
        createUser (event [username],event [pwd])
    except UserNameAlreadyExistsException,e: //catch error and do something
```

Explanation

You can define how to handle errors in your code to ensure the robustness and scalability of your application.

Returns error message

When exception handling and error capturing are not performed in the user's code logic, the SCF platform will attempt to capture errors as much as possible. For instance, if a user function suddenly crashes during execution and the platform is unable to capture the error,

the system will return a generic error message.
The table below displays some common errors that may occur during code execution:

Error Scenario	Error Message
`raise` is used to throw an exception	{File "/var/user/index.py", line 2, in always_failed_handler raise Exception ('xxx') Exception: xxx}
The handler does not exist	{'module' object has no attribute 'xxx'}
The dependent module does not exist	{global name 'xxx' is not defined}
Timed out	{"time out"}

Log

SCF platform stores all function call records and all outputs from the function code in the logs. Please use the print output statement or log statement in the programming language to generate output logs for easy debugging and troubleshooting. For more details, see Log Management.

Precautions

Given the characteristics of SCF, you must write your function code in a **stateless** style. Local file storage and other state features within the function lifecycle will be destroyed after the function call ends. Therefore, it is recommended to store persistent states in relational databases like TencentDB, object storage COS, cloud database Memcached, or other cloud storage services.

Development Process

For more information on the function development process, please see Getting Started.

Technical Selection
Function Deployment Selection

Last updated: 2023-09-27 14:25:42

To meet the needs of users in different usage scenarios, SCF offers two deployment methods: code deployment and image deployment. This document introduces the differences and applicable scenarios of these deployment methods for developers to reference when making their selection.

Analysis of Deployment Method Selection

The following compares the characteristics of the two different SCF deployment methods from various perspectives:

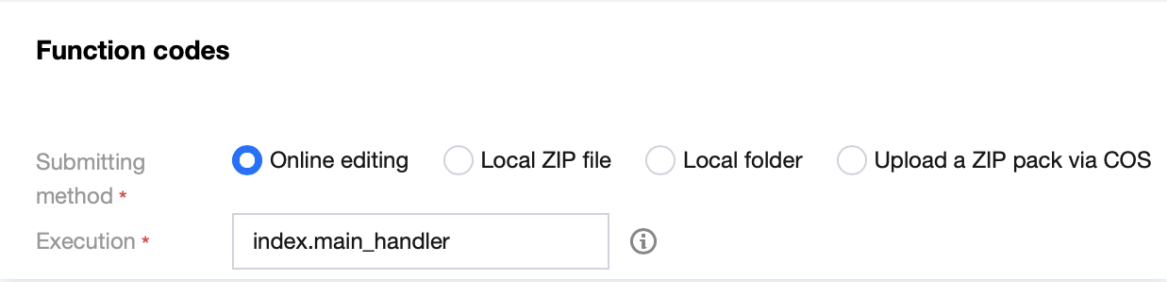
Entry	Code Deployment	Image Deployment
Description	Upload the code files to SCF to serve as the function's operational code, which runs on the base image provided by SCF.	Designate a user-created image as the operational environment for the function.
Size Limit	The combined size of the function code and associated layer version code is less than or equal to 500MB (before compression).	1GB (before decompression)
Format/Image Repository Support	zip , jar (java)	Tencent Container Registry
Running environment	• Python 3.7 • Python 3.6 • Python 2.7 • Node.js 16.13 • Node.js 14.18 • Node.js 12.16 • PHP 8.0 • PHP 7.4 • PHP 7.2 • PHP 5.6	No limit

	• Java11 • Java8 • Golang 1 • CustomRuntime	
Custom Startup File	Not supported	Supported
Setting the Listening Port	Not required	Fixed port 9000 must be set to listen.
Scenario	The basic runtime environment can accommodate scenarios where custom startup commands are not required.	The basic runtime environment does not currently support scenarios such as rapidly migrating original image-based operations to cloud functions.

Deployment mode

Code deployment

SCF offers three methods for code deployment:



- Upload a zip package online, supporting only code packages under 50 MB.
- Upload a folder online, supporting only folders under 250 MB.
- Upload a ZIP pack via COS

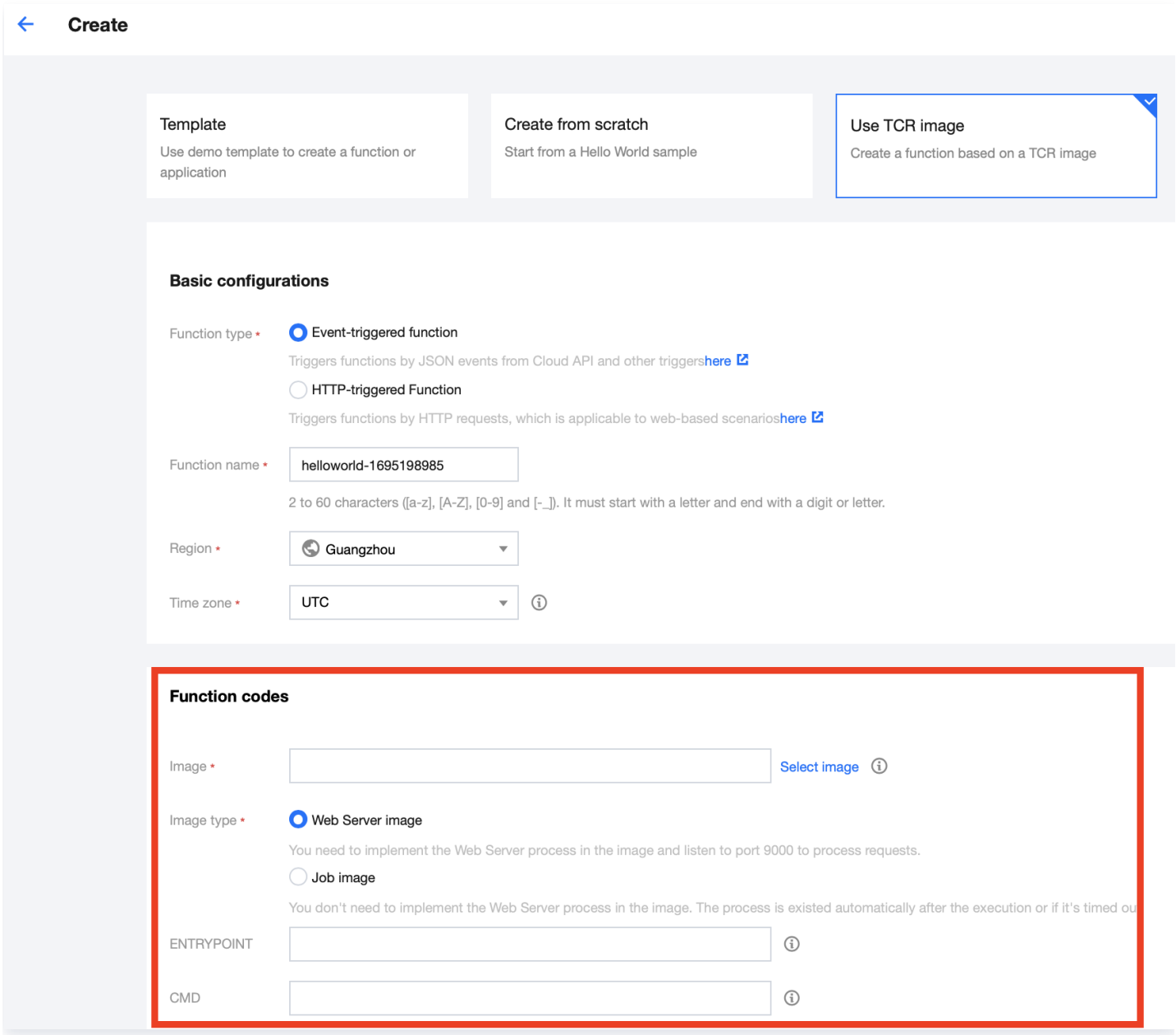
Explanation

The aforementioned three code deployment methods are all subject to the restriction that the total size of the code package and associated layer versions must be less than or equal to 500MB (before compression). If it exceeds 500MB, it is recommended to streamline the code package size, or upload some dependent files to Object Storage COS or File System CFS, and reference them in the function code. If dependent files

need to be referenced during the code initialization stage, it is recommended to use CFS.

Image deployment

The image deployment capability of SCF is supported by Tencent Cloud Container Registry. Tencent Cloud Container Registry is a cloud-based container image hosting service provided by Tencent Cloud, supporting Docker image, Helm Chart storage distribution, and image security scanning. By using the container image service, you no longer need to build and maintain your own image hosting service, and you can enjoy secure and efficient image hosting and distribution services in the cloud.
With the SCF image deployment feature, you simply need to push the locally built image to the TCR image repository and select the specified image when creating a function.



Function Type Selection

Last updated: 2024-06-14 14:57:28

To cater to the diverse needs of users in various scenarios, SCF offers two types of functions: Event Functions and HTTP-triggered Functions. This document provides a comparison of these function types and their applicable scenarios, serving as a reference for developers when making their selection.

> **Description**
> The method of function deployment is not directly related to the choice of function type. Both code deployment and image deployment can be used to create Event Functions or HTTP-triggered Functions.

Selection Analysis

The following is a comparison of the characteristics of the two different types of SCF functions from various perspectives:

Item	Event Function	HTTP-triggered Function
Design Philosophy	The new Event-Driven Architecture (EDA) development model triggers function execution through events. As a purely managed FaaS, it naturally integrates with cloud products, lowering the development threshold and shortening the research and development cycle.	Addressing the mainstream Serverlessful multi-threaded development model, it solves the issue of high transformation costs of traditional Web frameworks to FaaS. Users can directly send HTTP requests to the URL to trigger function execution.
Programming Paradigm	The function parameters are fixed as JSON format event and context, where context is platform information and does not support user customization, and event is user-defined information or trigger input information with a fixed format.	The function accepts native HTTP or WebSocket requests. The function writing method is close to native web services, and the event structure can be obtained through request headers and request bodies.
Deployment Method	Code, Image	Code, Image

Trigger Support	Scheduled Trigger, CKafka Trigger, COS Trigger, API Gateway Trigger, CLB Trigger, MPS Trigger, TDMQ Trigger, CLS Trigger, Cloud API.	API Gateway, supporting HTTP or WebSocket protocols.
Synchronous/Asynchronous Execution	Synchronous/Asynchronous	Synchronize
Multiple Concurrent Requests	Not supported	Supported
Custom Startup File	Not supported	Supported
Setting the Listening Port	Not required	Fixed port 9000 needs to be set for listening.
Application Scenarios	Business closely integrated with Tencent Cloud Service events, appropriately granulated business, and compute-intensive scenarios.	Website construction and API service site hosting.

Function type

Event-triggered function

The core concept of event-triggered functions is event-driven. Users select an event source (such as Cloud API, API Gateway, EventBridge, etc.) and define the event content. The event source will send the user-defined event content to the function in the format agreed with the

SCF platform according to the trigger rules selected by the user. The function is triggered by the event to run and returns the execution result to the user.
There are three basic concepts in SCF event functions: execution method, function input parameters, and function return. These concepts correspond to the following in typical project development:

- **Execution method**: corresponds to the main function of the project and is the starting point of program execution.
- **Function input parameter**: refers to function input parameters in a normal sense. However, in the SCF environment, the input parameters of an entry function are fixed values. For more information, please see Function Input Parameters.
- **Function return**: corresponds to the returned value of the main function in the project. After the function returns, the code execution ends.

Execution Method

When the SCF platform invokes a cloud function, it first seeks the execution method as the entry point to execute the user's code. At this point, users need to set it in the format of `filename.execution method name`.
For instance, if the user sets the execution method as `index.handler`, the SCF platform will first seek the `index` file in the code package and start executing the `handler` method in that file.
Within the execution method, users can process the input parameters of the entry function and call any other methods in the code. The execution of an SCF function ends when the entry function has finished executing or if there is an exception during execution.

Function input parameters

Function input parameters refer to the content passed to the function when it is triggered and called. Typically, function input parameters include two parts: **event** and **context**.
The `event` parameter is of `dict` type and contains the basic information that triggers the function. It can be in a platform-defined or custom format. After the function is triggered, the event can be processed inside the code.
The `context` is an input parameter provided by the SCF platform. By passing the `context` input parameter to the execution method, the code can parse the `context` input parameter object to obtain information about the runtime environment and the current request.

Function Return

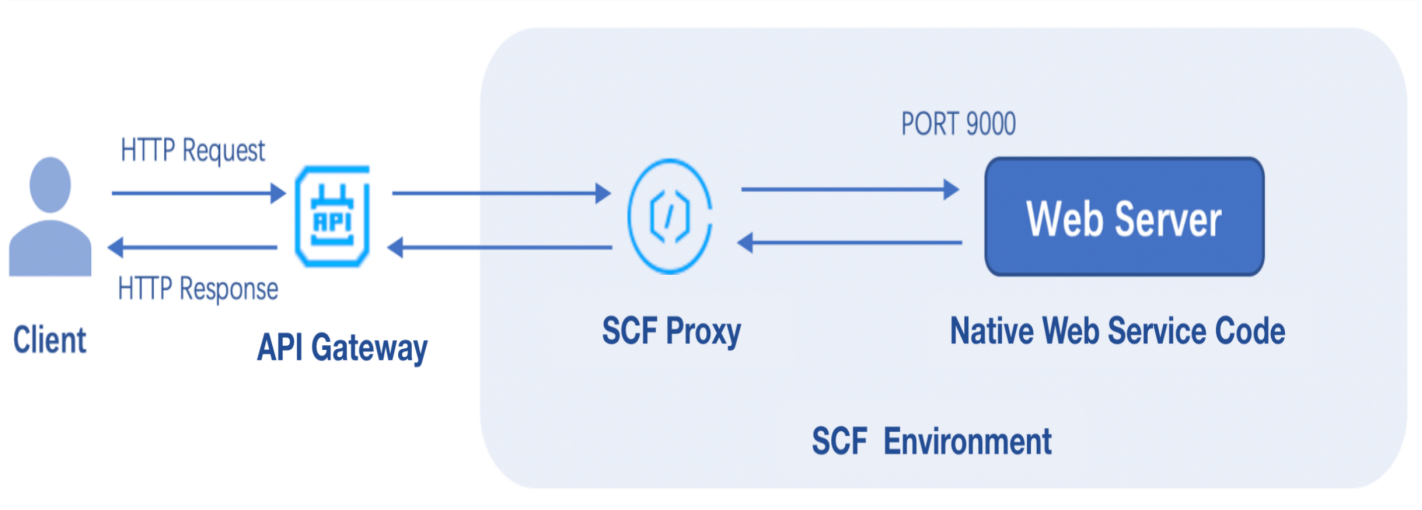
The SCF platform will obtain the return value after the cloud function execution and process it according to different triggering methods.

- **Synchronous Trigger:** When a function is triggered synchronously through the API Gateway or cloud API, the SCF platform does not return the trigger result during the

function execution. After the function execution is completed, the SCF platform will encapsulate the function return value into a JSON format and return it to the caller.

- **Asynchronous Trigger:** When a function is triggered asynchronously through a timer trigger, COS trigger, or other asynchronous triggers, or through a cloud API, the SCF platform will return a trigger request ID upon receiving the trigger event. After the function execution is completed, the function's return value will be encapsulated in JSON format and stored in the logs. Users can retrieve the return value of the asynchronously triggered function from the logs using the returned request ID after the function execution is completed.

HTTP-triggered function



Unlike event functions that are restricted by event formats, SCF HTTP-triggered Functions are specifically optimized for web service scenarios, allowing users to directly send HTTP requests to a URL to trigger function execution.
After the user's HTTP request passes through the API Gateway, the gateway directly transmits the original request and adds platform information such as the function name and function region required when the gateway triggers the function to the request header. This information is then passed to the function environment, triggering the backend function execution.
In the function environment, the built-in proxy is used to implement Nginx-based forwarding, remove the request information not required by the service specification from the header, and send the native HTTP request to your web server service through the specified port.
After being configured with the specified listening port `9000` and service bootstrap file, your web server will be deployed in the cloud and use this port to get HTTP requests for processing.

Function Storage Selection

Last updated: 2023-09-27 14:38:38

The SCF provides an elastic, pay-as-you-go FaaS service, serving as a workload for a multitude of Serverless applications. During operation, developers may need to access various external data due to business requirements, such as importing third-party libraries, unstructured data, and function computation outputs. This necessitates a rich variety of storage support types provided by the SCF to cater to the developers' business needs.
The SCF currently supports a variety of storage types, including integration with other cloud products such as COS and CFS, as well as local temporary storage and layers. This document outlines the differences between these storage types, serving as a reference for developers when choosing a storage type.

Selection Analysis

The table below compares the characteristics of the four different data storage options provided by the SCF from various perspectives:

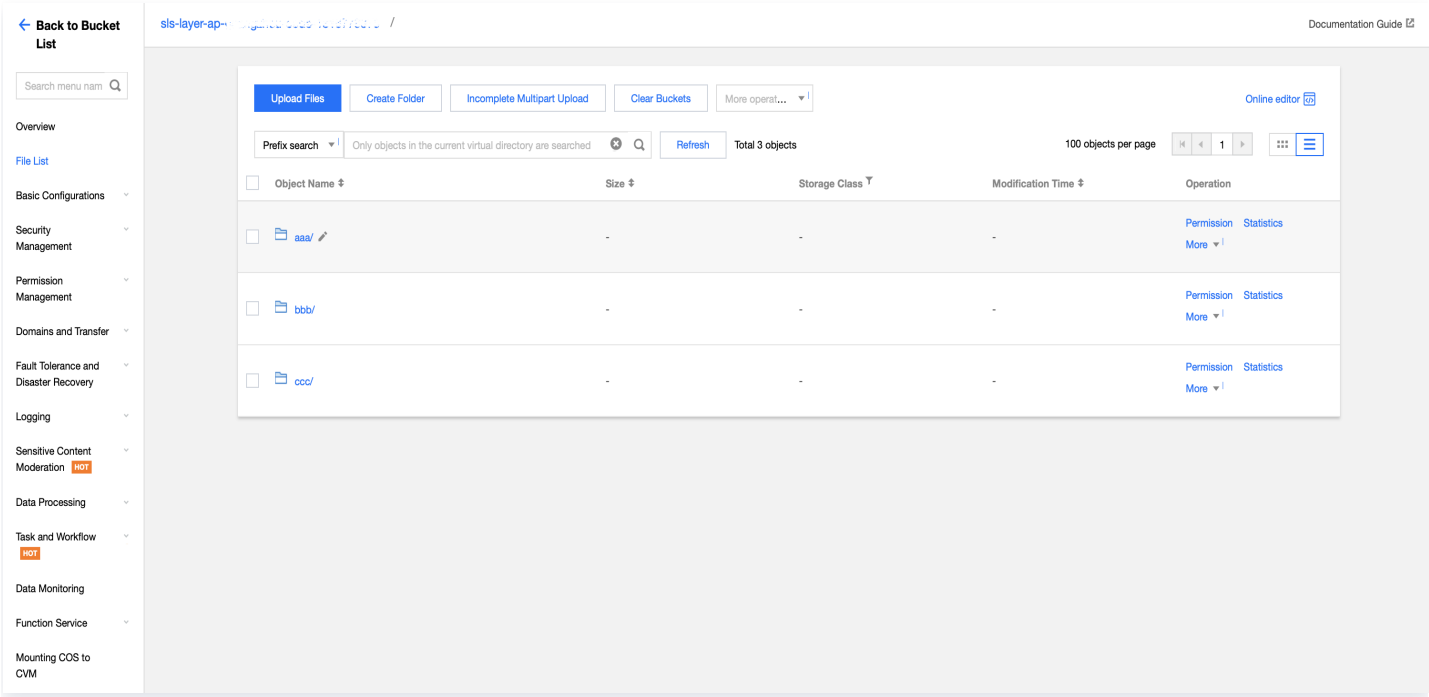
Entry	COS	Cloud File System (CFS)	Temporary Storage /tmp	Layer
Maximum Space	elastic	elastic	512M	The combined size of the layer version code and the code of the function versions associated with the layer is 500 MB (before compression).
Persistence	Persistence	Persistence	Temporary Storage	Persistence
Storage Content	Writable	Writable	Writable	Non-writable
Storage type	Object	File system	File system	Code Dependency Archiving
Event Source Integration	Yes	No	No	No
Operation	COS	File Read/Writ	File Read/Write	Write operation is not permitted during runtime.

		e		
Cost	Storage, Request, Management	Storage	No additional charges incurred.	No additional charges incurred.
Shared among calls	Yes	Yes	No	Yes
Function's Access Speed	Swift	Extremely swift	Fastest	Swift
Scenario	Log and Business File Storage	Log and Business File Storage	Temporary files generated by business operations	Minimize code package references or reuse of code libraries and tools between functions

Storage type

COS

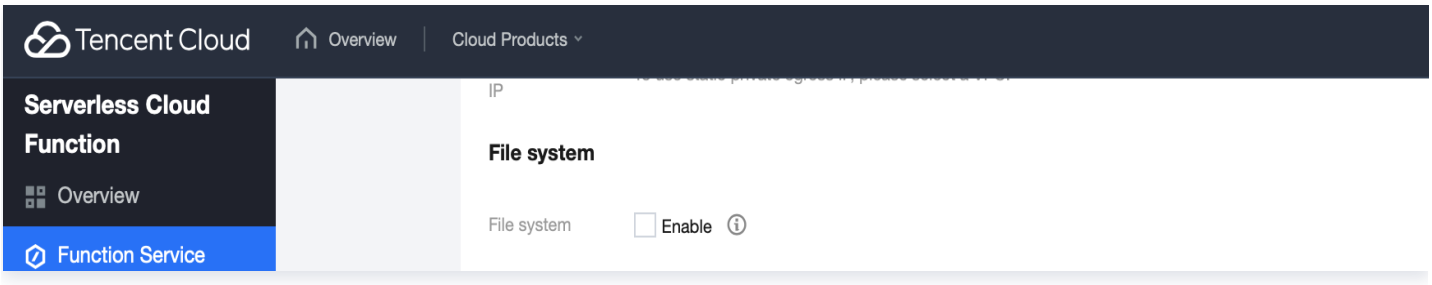
Tencent Cloud's Cloud Object Storage (COS) is a distributed storage service designed to handle a massive amount of files. It offers high scalability, low cost, and reliable security. COS is particularly suitable for storing unstructured data, including binary data such as images or media, log files, and sensor data.



Developers can utilize the SDK provided by SCF to quickly implement object creation and deletion in COS Bucket using the standard HTTP protocol without mounting, thereby completing the business logic loop. Simultaneously, COS can act as an **event source**, publishing events to SCF and invoking functions with event data, thereby achieving function event source integration.
Implement real-time audio and video transcoding based on SCF + COS to meet the needs of users in different scenarios when the total amount and frequency of uploaded images, audio and video are high. Multiple cloud functions are used to process different clarity levels, adapting to the characteristics of mobile networks with smaller and unstable bandwidth.
Object storage is an important repository for data lakes. The Serverless architecture's data lake solution is triggered by cloud function triggers. After data invocation, the cloud function captures and records batch data information, completing related structure conversion, data format conversion, and data compression capabilities within the function.
Additionally, the intelligent tiered storage type of COS provides a hot-cold layering mechanism for data. It can automatically convert the hot and cold levels of data based on the user's data access patterns, thereby reducing the storage cost of user data. It is used for data with unpredictable or variable access patterns, offering users a product experience with low latency and high throughput consistent with standard storage, but at a lower cost.

Cloud File System (CFS)

Tencent Cloud's Cloud File Storage (CFS) provides a scalable shared file storage service. In addition to cloud functions, it can also be used in conjunction with Tencent Cloud servers, container services, or batch processing services.



CFS adheres to the standard NFS file system access protocol, providing a shared data source for multiple computing nodes. It supports elastic capacity and performance expansion, and existing applications can be mounted without modification. As a highly available and reliable distributed file system, CFS is suitable for scenarios such as big data analysis, media processing, and content management. CFS is cost-effective, adopting a pay-as-you-go billing model with an hourly billing cycle, where you only pay for the actual storage space used.
Cloud Function SCF supports seamless integration with CFS. With the necessary configurations, your functions can easily access files stored in the CFS file system. The advantages of using CFS are as follows:

- The function execution space is unlimited.
- Multiple functions can share the same file system so as to share files.

Temporary Storage /tmp

The SCF execution environment provides a file system in the `/tmp` directory for developers to use, with a fixed size of 512 MB. This storage space is exclusively used by a single function instance. Multiple invocations of a function can reuse the same execution environment to optimize performance, and the data will be retained for the lifetime of the function instance execution.
Hence, this is intended to serve as a temporary storage space. While a function can store data between invocations, it should only be used for data required by the code within a single invocation. It is not a place for permanent data storage, but rather, it is better suited to support operations required by the code.
In terms of operation and usage, using files in /tmp is the same as using cloud disks. You can directly read and write with code, and it provides fast I/O throughput. Below is an example of file compression using Python.

```
import os, zipfile
os.chdir('/tmp')
with zipfile.ZipFile(yourzipfile, 'r') as zip:
    zip.extractall()
```

Layer

Layers are the solution provided by SCF to handle the storage of common dependency libraries. If SCF has a large number of dependency libraries or common code files, you can manage them using layers in SCF. By managing layers, dependencies can be placed in layers instead of deployment packages, ensuring that the deployment packages remain small in size. Layer management is static. The compressed file created for a layer will be stored according to the layer's version. When a layer is bound to a function, it will be bound according to the specific layer version and function version. Currently, a function can bind up to 5 specific versions of layers, and there is a certain order when binding.

Strengths

Last updated: 2023-09-27 14:38:56

Ease of Use

Minimize Component Expenditure

When utilizing cloud functions, users only need to focus on writing the "core code", eliminating the need to manage peripheral components such as load balancing, auto-scaling, and gateways. This significantly reduces the complexity of setting up service architectures.

Automatic scaling

Without any manual configuration, cloud functions can automatically scale horizontally based on the volume of requests. Whether your application only has a few requests per day (such as regular transactions like log statistics) or thousands to tens of thousands of requests per second (such as the backend of a mobile application), cloud functions can automatically allocate appropriate computing resources to meet business needs.

Efficient and Creative Development

Accelerate Development

Cloud functions do not require a specific framework or dependencies, allowing developers to concentrate on the development of core code. At the same time, developers can form multiple small teams, with the development of individual modules not requiring an understanding of the code details of other teams. The speed of independent development and iteration becomes unprecedentedly fast, helping users seize the golden time for product launch.

Reuse of Third-Party Services

You can use cloud functions to write some business modules with a single purpose and independent logic, thus you can fully reuse mature third-party code implementations, such as using OAuth to implement a login module.

Simplify Ops

Each function operates, deploys, and scales independently. Once the user uploads the code, it can be automatically deployed, eliminating the difficulties associated with deploying and upgrading monolithic applications.

Stability & Reliability

High availability deployment

Cloud functions can automatically select availability zones at random in each region for operation. If a disaster or power failure causes paralysis in a certain availability zone, cloud functions will automatically select the infrastructure of other available zones for operation, eliminating the risk of failure in a single availability zone.

Complements other computing services

Persistent workloads can be handled through Cloud Virtual Machine (CVM) and Tencent Kubernetes Engine (TKE), while event-triggered workloads can utilize cloud functions. Different cloud services cater to various business scenarios and requirements, making your service architecture more robust.

Simplified Management

Simplified Security Configuration

Users no longer need to perform complex configuration and management for OS intrusion, login risks, file system security, network security, and port monitoring. All these are handled by the platform, which ensures the isolation of each user through customized containers.

Visualization Management

Users can directly manage function code and when the function runs (i.e., function triggers) from the console. With no need for complex configuration files, functions can be deployed and tested with a single click.

Significantly Reduced Overhead

Never pay for idle time

Functions do not incur any costs when not executing, significantly reducing overhead for non-permanent business processes. When functions are executed, charges are based on the number of requests and the runtime of computing resources. This pricing model offers a clear advantage and is particularly beneficial for developers in their start-up phase.

Scenarios

Last updated：2023-09-27 14:39:13

Tencent Cloud's Serverless Cloud Function (SCF) is currently undergoing continuous iteration and development. As the product capabilities and integrated products continue to grow, the application scenarios adapted by SCF will also increase. The following video will introduce you to the application scenarios of SCF:
Watch video

File Processing and Notifications

By using COS as a function trigger, event notifications can be sent when a file in a COS bucket changes, which enables the prompt processing of the changed file and business notifications. For example, once an image is uploaded to a COS bucket, the cloud function is notified immediately, and the image can be immediately obtained for automatic processing, such as cropping, thumbnailing, and watermarking. In addition, the processed image can be written to a database for future use.

Data ETL Processing

Some data processing systems need to process huge amounts of data frequently, either periodically or planned.
For example, a securities company needs to collect statistics on its transactions every 12 hours and determine the top 5 transaction volumes. Another example is that a flash sale website needs to process its transaction flow logs once a day to obtain errors caused by the resources being sold out and thereby analyze the website buzz and trends. Possessing great scalability, SCF facilitates the computation of large-volume data. SCF also enables the concurrent execution of multiple mapper and reducer functions on the source data, and finishes the tasks in a short period of time. Compared with traditional functions, SCF reduces costs by preventing resources from being idleness and wasting resources.

Mobile and Web Applications

Cloud functions let you run mobile and web application backend code to implement the server-side application logic and provide services through APIs. By using SCF with Cloud Cache, TencentDB, COS, and other products, developers can build mobile and web applications with elastic scalability and create various serverless backends. These applications can run in multiple IDCs with high availability. You don’t need to manage the scalability and backup redundancy.

AI Inference

After an AI model is trained and ready to provide inference services, you can use SCF to encapsulate the model into a function. The code is executed upon request reception. In this way, you only need to pay on demand without investing in servers and GPUs to get the automatic scaling capability featuring high request concurrency.

Mini program

Tencent CloudBase is a native Serverless cloud service developed jointly by the WeChat team and Tencent Cloud, integrated into the Mini Program console. Its core features include: Cloud Functions, Cloud Database, and Cloud Storage. Cloud Functions allow developers to run code in the cloud, with developers only needing to write their own business logic code. Combined with WeChat's private default authentication, the platform ensures security and isolation, and automatically scales according to requests.

Message Archiving

CMQ and CKafka can be used as function triggers. A received message triggers the execution of the cloud function, and the message is passed to the cloud function as the event content. For example, when CKafka receives a log of a business system, the cloud function can write the log content as a file to COS for log archive and storage.

Business Transition

As the intermediate channel of business event flow, CMQ connects multiple functions for business status flow and assignment. According to the messages, business logic judgment and processing in the functions can be performed to implement channel assignment, status flow, and event distribution, which connect the complex business processes.

Relevant Products

Last updated: 2023-09-27 14:39:31

The Tencent Cloud Serverless Cloud Function (SCF) may be associated with and utilize the following products during its operation:

Product Name	Relationship with Cloud Functions
Virtual Private Cloud (VPC)	By configuring the Cloud Function into a VPC, it is possible to access resources within the VPC.
Cloud Object Storage (COS)	By configuring the Object Storage trigger, the Cloud Function can be activated when an event occurs in the corresponding bucket.
Cloud Log Service (CLS)	By configuring the integration with the log service, the execution logs of the Cloud Function can be written into the log service.
Cloud Message Queue (CMQ)	By configuring the Message Queue trigger, the Cloud Function can be activated when a message is received in the corresponding queue.
Cloud Kafka (CKafka)	By configuring the CKafka trigger, the Cloud Function can be activated when a message is received in the corresponding Kafka topic.
API Gateway	By configuring the API Gateway trigger, the Cloud Function can be activated when an HTTP request is received on the API URL.
Cloud Access Management (CAM)	By configuring roles in Cloud Access Management, you can grant permissions to access authorized resources when the Cloud Function executes the code.