Function Compute

Advanced Tutorial

Advanced Tutorial

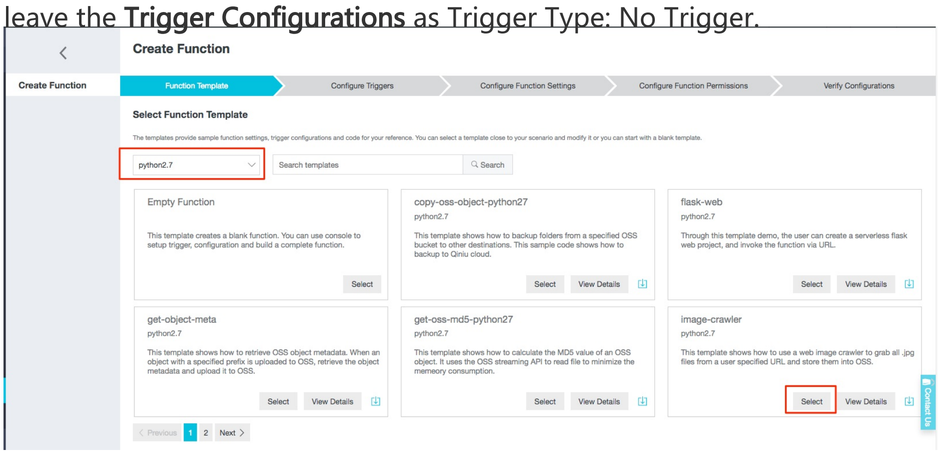
Use template

The Function Compute (FC) console provides function templates to easily create functions such as web crawlers, automatic image classification, and access to Alibaba Cloud Object Storage Service (OSS) or Table Store that can be customized to fulfill your own requirements. The following example demostrates how to use a web crawler template to create a function for web crawling.

Steps

Create a FC service named DEMO in the console.

Create a function inside service DEMO, select the **image-crawler** function template, and leave the **Trigger Configurations** as Trigger Type: No Trigger.



Select Function Template

The templates provide sample function settings, trigger configurations and code for your reference. You can select a template close to your scenario and modify it or you can start with a blank template.

python2.7

Search templates

Empty Function

This template creates a blank function. You can use console to setup trigger, configuration and build a complete function.

copy-oss-object-python27
python2.7

This template shows how to backup folders from a specified OSS bucket to other destinations. This sample code shows how to backup to Qiniu cloud.

flask-web
python2.7

Through this template demo, the user can create a serverless flask web project, and invoke the function via URL.

get-object-meta
python2.7

This template shows how to retrieve OSS object metadata. When an object with a specified prefix is uploaded to OSS, retrieve the object metadata and upload it to OSS.

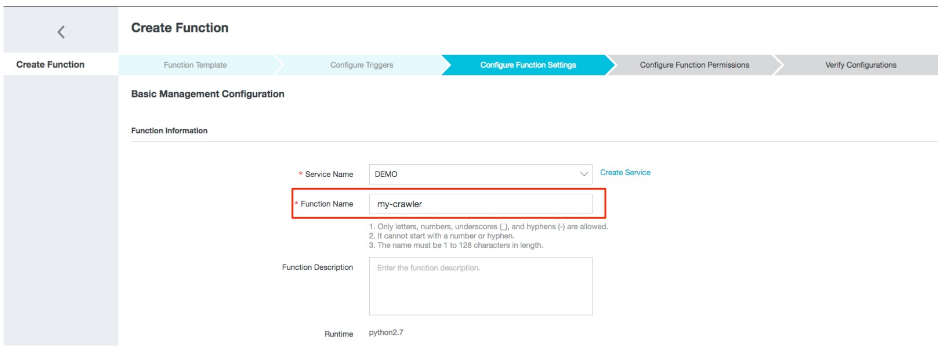
get-oss-md5-python27
python2.7

This template shows how to calculate the MD5 value of an OSS object. It uses the OSS streaming API to read file to minimize the memeory consumption.

image-crawler
python2.7

This template shows how to use a web image crawler to grab all .jpg files from a user specified URL and store them into OSS.

In the **Configure Function Settings** step, provide the Function Name and Function Description (optional).



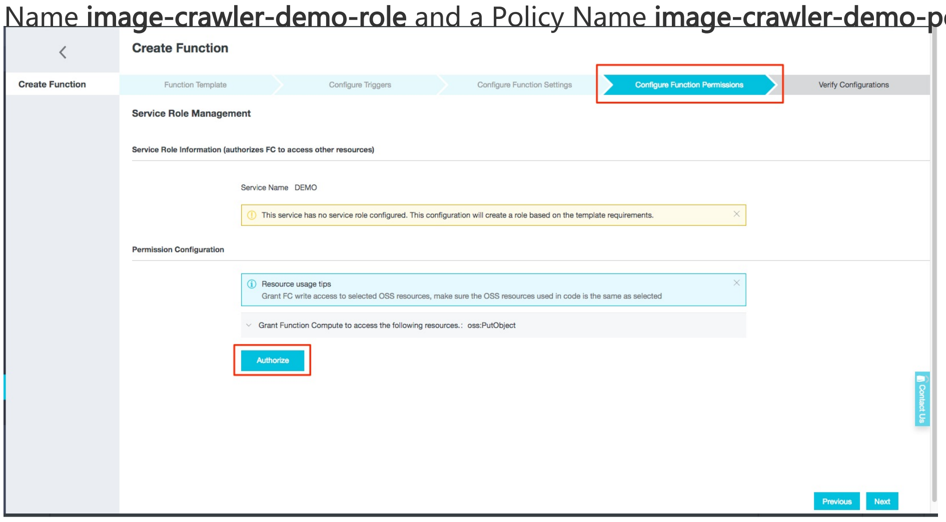
Basic Management Configuration

Function Information

* Service Name	DEMO
* Function Name	my-crawler
Function Description	Enter the function description.
Runtime	python2.7

1. Only letters, numbers, underscores (_), and hyphens (-) are allowed.
2. It cannot start with a number or hyphen.
3. The name must be 1 to 128 characters in length.

In the **Configure Function Permissions** step, click Authorize, after re-directed, give a Role Name **image-crawler-demo-role** and a Policy Name **image-crawler-demo-policy**



Service Role Management

Service Role Information (authorizes FC to access other resources)

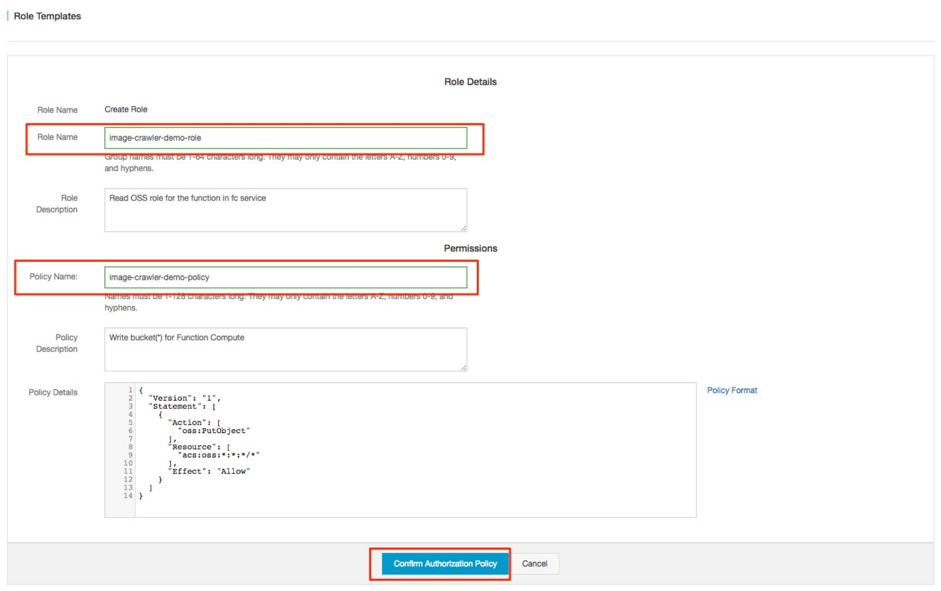
Service Name DEMO

This service has no service role configured. This configuration will create a role based on the template requirements.

Permission Configuration

Resource usage tips
Grant FC write access to selected OSS resources, make sure the OSS resources used in code is the same as selected

Grant Function Compute to access the following resources.: oss:PutObject



Role Details

Role Name	Create Role
Role Name	image-crawler-demo-role
Role Description	Read OSS role for the function in fc service

Group names must be 1-64 characters long. They may only contain the letters A-Z, numbers 0-9, and hyphens.

Permissions

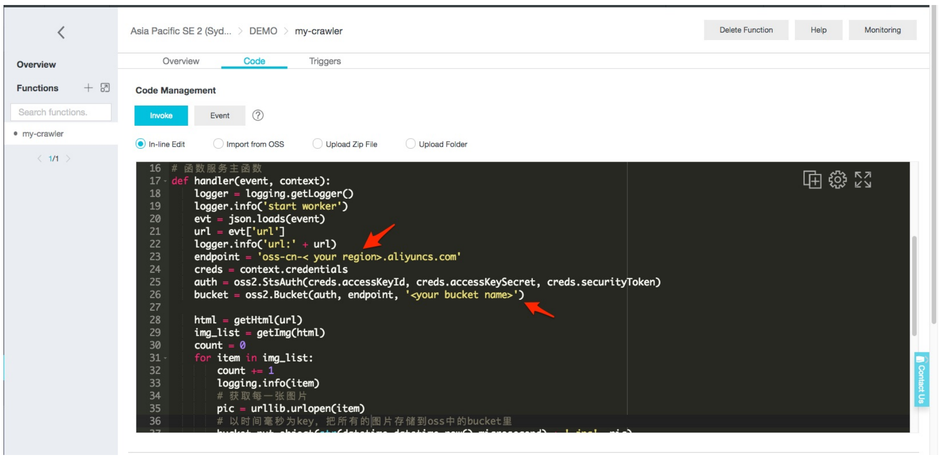
Policy Name:	image-crawler-demo-policy
Policy Description	Write bucket(*) for Function Compute

Names must be 1-128 characters long. They may only contain the letters A-Z, numbers 0-9, and hyphens.

Policy Details

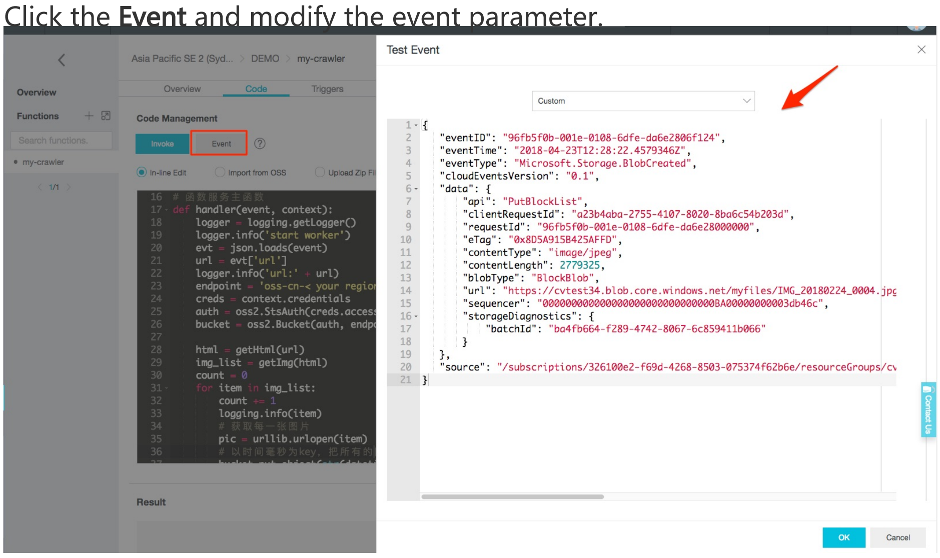
```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:PutObject"
      ],
      "Resource": [
        "acs:oss:*:*:*/*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

In **Code Management**, edit and customize the code online, replace **your region** and **your bucket name** in the code with real values.



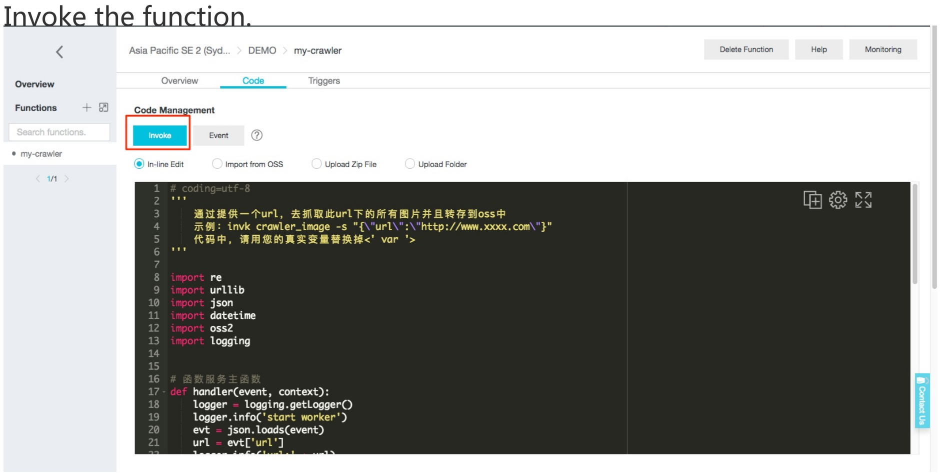
```python
# 函数服务主函数
def handler(event, context):
    logger = logging.getLogger()
    logger.info('start worker')
    evt = json.loads(event)
    url = evt['url']
    logger.info('url:' + url)
    endpoint = 'oss-cn-< your region>.aliyuncs.com'
    creds = context.credentials
    auth = oss2.StsAuth(creds.accessKeyId, creds.accessKeySecret, creds.securityToken)
    bucket = oss2.Bucket(auth, endpoint, '<your bucket name>')

    html = getHtml(url)
    img_list = getImg(html)
    count = 0
    for item in img_list:
        count += 1
        logging.info(item)
        # 获取每一张图片
        pic = urllib.urlopen(item)
        # 以时间毫秒为key, 把所有的图片存储到oss中的bucket里
```

Click the **Event** and modify the event parameter.



```json
{
    "eventID": "96fb5f0b-001e-0108-6dfe-da6e2806f124",
    "eventTime": "2018-04-23T12:28:22.4579346Z",
    "eventType": "Microsoft.Storage.BlobCreated",
    "cloudEventsVersion": "0.1",
    "data": {
        "api": "PutBlockList",
        "clientRequestId": "a23b4aba-2755-4107-8020-8ba6c54b203d",
        "requestId": "96fb5f0b-001e-0108-6dfe-da6e28000000",
        "eTag": "0x8D5A915B425AFFD",
        "contentType": "image/jpeg",
        "contentLength": 2779325,
        "blobType": "BlockBlob",
        "url": "https://cvtest34.blob.core.windows.net/myfiles/IMG_20180224_0004.jpg
        "sequencer": "000000000000000000000000000000BA00000000003db46c",
        "storageDiagnostics": {
            "batchId": "ba4fb664-f289-4742-8067-6c859411b066"
        }
    },
    "source": "/subscriptions/326100e2-f69d-4268-8503-075374f62b6e/resourceGroups/cv
}
```

Invoke the function.



```python
# coding=utf-8
'''
    通过提供一个url, 去抓取此url下的所有图片并且转存到oss中
    示例: invk crawler_image -s "{\"url\":\"http://www.xxxx.com\"}"
    代码中, 请用您的真实变量替换掉<' var '>
'''

import re
import urllib
import json
import datetime
import oss2
import logging


# 函数服务主函数
def handler(event, context):
    logger = logging.getLogger()
    logger.info('start worker')
    evt = json.loads(event)
    url = evt['url']
```

Conclusion

Using a function template can make permissions configurations and code development easier. More templates will be added overtime to help developers stay focus on business logic development rather than setups.

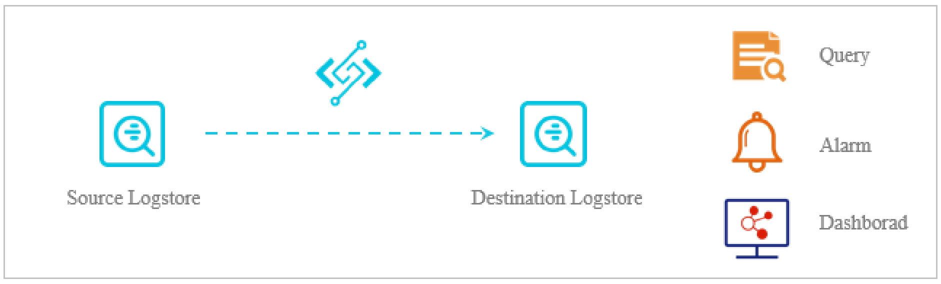
Log Service trigger sample

Demo overview

Function Compute integrated with Log Service can provide a fully-managed processing service for streaming data.

Introduction

You can configure a Log Service trigger. This trigger regularly obtains updated data and triggers function execution. In this way, Function Compute consumes incremental data from the Logstore in Log Service, and completes custom processing tasks, such as data scrubbing and extract, transform, and load (ETL). Then, Function Compute ships data to a third-party service, as shown in the following figure:



The functions used to process data can be templates provided by Log Service or your custom functions.

In this sample, Log Service uses a function from Function Compute to retrieve log files and print them to Function Compute. This example describes how to use the Log Service trigger to integrate Log Service with Function Compute.

Before starting the process that is shown in this example, make sure that you have signed up for Function Compute and Log Service.

Contents

To run a function triggered by the Log Service event source, follow these steps:

Activate Log Service, and create the project and Logstores of Log Service and the Log Service trigger.

Author the function.

Test your function.

References

Developer Guide

1. Set up services

In this guide, you will go through the process of activating Log Service, creating a Log Service project and two Logstores, creating a Function Compute service, creating a Function Compute function, and then configure a Log Service trigger. Skip this guide if you have available Log Service, Functions, and Log Service Trigger.

Preparations

Make sure that Function Compute and Log Service are deployed in the same region. Otherwise, Function Service cannot locate Log Service when you configure the Log Service trigger. For more information, see Regions and zones.

Log on to the Alibaba Cloud console.

Navigate to the AliyunLogETLRole role management page, click the **Confirm Authorization Policy** button to grant Function Compute the AliyunLogETLRole access policy.

Log on to Log Service console, create one Logstore to process log files and data sources, and create another Logstore to store the log files that are generated by Function Compute. For more information, see *Log Service* topic Preparation.

Log on to the Function Compute console and create a Service. In the **Create Service** dialog box that appears:

Select a region, in this sample, we use the **China (shanghai)** region.

Click **Create Service**.

In the Create Service dialog box that appears, enter the service name, in this sample, the service name is **log-com**.

Enable the **Advanced Settings** option.

In the **Log Configs**, set your available Log Project and Logstore.

In the **Role Config**, select **Create new role** from the Role Operation drop-down list, select AliyunLogFullAccess and AliyunLogReadOnlyAccess from the System Policies drop-down list.

Click **Authorize** and **OK** to confirm your action.

In the left-side navigation pane, select the new service you created.

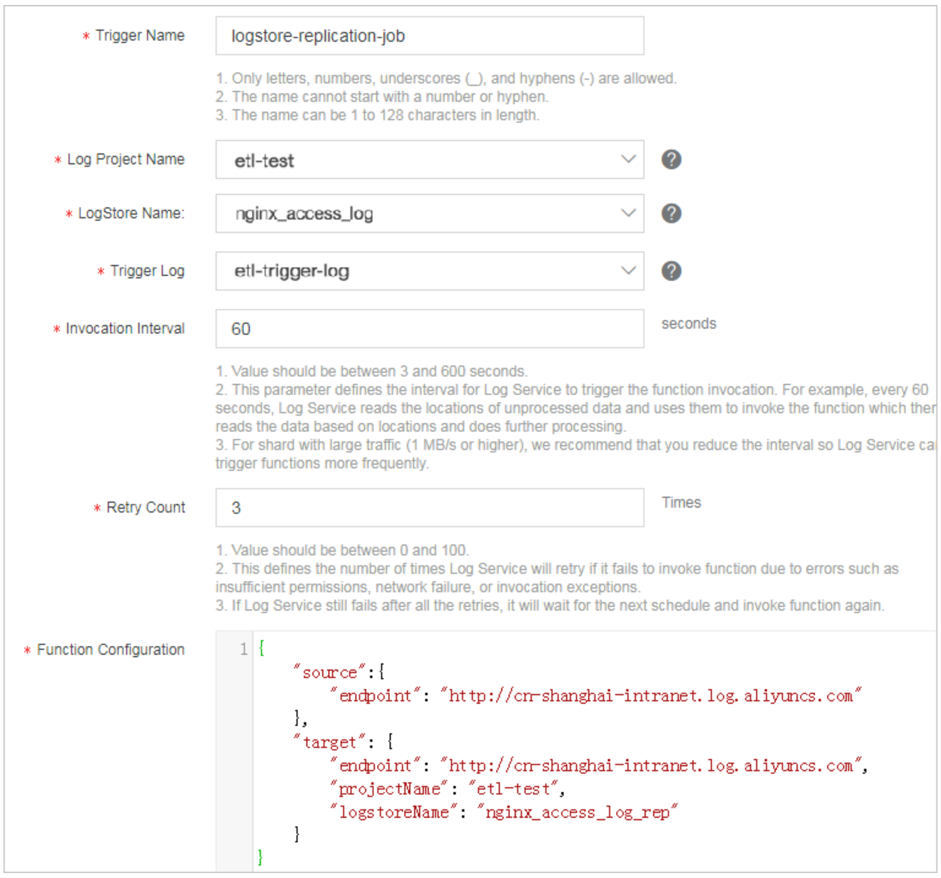
Click **Create Function** to go to the **Create Function** page.

i. Click **Select All**, and select **python2.7** from the drop-down list. This sample takes Python code execution for example.

Click **Select** in **Empty Function**.

Note: You can create the trigger during or after function creation. Then, you can configure the trigger. For more information, see Basic operations.

Select **Log Service (Log)** from the Trigger Type drop-down list, set the **Trigger Name**, **Log Project Name**, **Trigger Log**, **Invocation Interval**, **Retry Count**, and **Function Configuration** configuration. In this sample, we set the trigger as follows:



* Trigger Name: logstore-replication-job

1. Only letters, numbers, underscores (_), and hyphens (-) are allowed.
2. The name cannot start with a number or hyphen.
3. The name can be 1 to 128 characters in length.

* Log Project Name: etl-test

* LogStore Name: nginx_access_log

* Trigger Log: etl-trigger-log

* Invocation Interval: 60 seconds

1. Value should be between 3 and 600 seconds.
2. This parameter defines the interval for Log Service to trigger the function invocation. For example, every 60 seconds, Log Service reads the locations of unprocessed data and uses them to invoke the function which ther reads the data based on locations and does further processing.
3. For shard with large traffic (1 MB/s or higher), we recommend that you reduce the interval so Log Service ca trigger functions more frequently.

* Retry Count: 3 Times

1. Value should be between 0 and 100.
2. This defines the number of times Log Service will retry if it fails to invoke function due to errors such as insufficient permissions, network failure, or invocation exceptions.
3. If Log Service still fails after all the retries, it will wait for the next schedule and invoke function again.

* Function Configuration

```
{
    "source":{
        "endpoint": "http://cn-shanghai-intranet.log.aliyuncs.com"
    },
    "target": {
        "endpoint": "http://cn-shanghai-intranet.log.aliyuncs.com",
        "projectName": "etl-test",
        "logstoreName": "nginx_access_log_rep"
    }
}
```

You can set the **Function Configuration** field according to the event parameter in your function code. In this sample, we set the **Function Configuration** as follows:

```
{
"source":{
"endpoint": "http://cn-shanghai-intranet.log.aliyuncs.com"
},
"target": {
"endpoint": "http://cn-shanghai-intranet.log.aliyuncs.com",
"projectName": "etl-test",
"logstoreName": "nginx_access_log_rep"
}
}
```

Configure the **Service Name**, **Function Name**, **Function Description**, **Runtime**, and **Runtime Environment** parameters.

v. Click **Next**.

vi. Make sure that all the settings are correct, and then click **Create**.

Next step

2. Author your function.

References

For more information about creating services in the console, see Create a service.

Function logs are saved to the Logstore and used in debugging. For more information about logs, see Function logs.

Roles that you configure in Function Compute automatically have access permissions for other cloud services. For more information about permissions, see Introduction.

2. Author your function

After the previous step and your Service is ready, you can create new functions in the service. You can use the Function Compute console or the fcli tool to submit your functions. This guide provides sample code that uses the specified programming language to the code edit field in the Function Compute console.

In this guide, incremental log entries collected by Log Service trigger executing this function. Then, the function retrieves log entries, and prints them to Function Compute. For more information about programming, see *Log Service* topic Development guide for ETL function.

Sample function for Python 2.7

You can use this code example as the template for retrieving all logical-log files. The accessKeyId and accessKey must be either manually specified or obtained from context and creds objects.

```
# -*- coding: utf-8 -*-
import logging
import json
from aliyun.log import LogClient
from time import time

def logClient(endpoint, creds):
logger = logging.getLogger()
logger.info('creds info')
logger.info(creds.access_key_id)
logger,info(creds.access_key_secret)
logger,info(creds.security_token)
accessKeyId = 'your accessKeyId'
accessKey = 'your accessKeyId scr'
```

```
client = LogClient(endpoint, accessKeyId, accessKey)
return client

def handler(event, context):
logger = logging.getLogger()
logger.info('start deal SLS data')
logger.info(event.decode().encode())
info_arr = json.loads(event.decode())
fetchdata(info_arr['source'])
return 'hello world'

def fetchdata(event):
logger = logging.getLogger()
endpoint = event['endpoint']
creds = context.credentials
client = logClient(endpoint, creds)
if client == None :
logger.info("client creat failed")
return False
project = event['projectName']
logstore = event['logstoreName']
start_cursor = event['beginCursor']
end_cursor = event['endCursor']
loggroup_count = 10
shard_id = event['shardId']
while True:
res = client.pull_logs(project, logstore, shard_id, start_cursor, loggroup_count, end_cursor)
res.log_print()
next_cursor = res.get_next_cursor()
if next_cursor == start_cursor :
break
start_cursor = next_cursor

#log_data = res.get_loggroup_json_list()
return True
```

Request parameters

The event parameter is the Function Compute object. The format of event is as follows:

```
{
"parameter": {},
"source": {
"endpoint": "http://cn-shanghai-intranet.log.aliyuncs.com",
"projectName": "log-com",
"logstoreName": "log-en",
"shardId": 0,
"beginCursor": "MTUyOTQ4MDIwOTY1NTk3ODQ2Mw==",
"endCursor": "MTUyOTQ4MDIwOTY1NTk3ODQ2NA=="
},
"jobName": "1f7043ced683de1a4e3d8d70b5a412843d817a39",
"taskId": "c2691505-38da-4d1b-998a-f1d4bb8c9994",
"cursorTime": 1529486425
}
```

parameter: Contains the function configuration of the trigger.

source: The information of the log block that the triggered function reads from Log Service.

- endpoint: The region of the Log Service project.
- projectName: The Log Service project name.
- logstoreName: The Logstore name.
- shardId: The specified shard in the Logstore.
- beginCursor: The location on the shard where the triggered function starts consuming data.
- endCursor: The location on the shard where the triggered function stops consuming data.

jobName: The name of the Extract, Transform, and Load (ETL) job in Log Service. A Log Service trigger in Function Compute corresponds to an ETL job in Log Service.

taskId: The identifier of the execution of the specified function. This execution corresponds to an ETL job.

cursorTime: The unix_timestamp of the last log entry that reaches Log Service. The function reads this data when retrieving log entries.

Next step

3. Test your function.

3. Debug your function

In this guide, we use Function Compute console for debugging. You can write an incremental log entry to Log Service, and check whether Function Compute prints this log entry or not.

Create and debug a HELLO WORLD function

Create the function that prints Hello World.

> **Note**: When you create this function, enable **Advanced Settings**, and set LogStore to the Logstore that the demo uses in execution.

Click **Invoke**.

Check whether or not the Logstore that you have configured for the Log Service trigger has collected the incremental log entry.

Now, you have created the program for executing the function triggered by Log Service. For more information, see *Log Service* topic Development guide for ETL function.