

**EventGrid**

# **Best Practices**

**Issue** 01  
**Date** 2025-08-07



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.**

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

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

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

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

## **Huawei Cloud Computing Technologies Co., Ltd.**

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

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

# Contents

<b>1 EG Best Practices.....</b>	<b>1</b>
<b>2 Routing OBS Application Service Messages to DMS for Kafka.....</b>	<b>2</b>
2.1 Introduction.....	2
2.2 Resource and Cost Planning.....	2
2.3 General Procedure.....	3
2.4 Implementation Procedure.....	4
2.4.1 Creating an OBS Bucket.....	4
2.4.2 Buying a Kafka Instance.....	5
2.4.3 Creating a Kafka Connection.....	7
2.4.4 Creating an Event Subscription.....	8
2.4.5 Sending an OBS Event.....	9
2.4.6 Viewing Event Messages.....	9
<b>3 Data Synchronization Between Kafka Instances Based on Serverless Event Stream .....</b>	<b>11</b>
3.1 Overview.....	11
3.2 Implementation and Verification.....	11
<b>4 Synchronizing OBS Bucket Data Based on Event Subscription and FunctionGraph .....</b>	<b>17</b>
4.1 Overview.....	17
4.2 Procedure.....	17
4.3 Implementation Procedure.....	18
4.3.1 Creating an OBS Source Bucket and a Target Bucket.....	19
4.3.2 Creating a Function.....	19
4.3.3 Creating an Event Subscription.....	20
4.3.4 Sending an OBS Event.....	21
4.3.5 Viewing Event Delivery Details.....	22
4.3.6 Viewing Code Execution Details in FunctionGraph.....	23
4.3.7 Checking the Synchronization Result.....	23
<b>5 Routing Events from RocketMQ to FunctionGraph Based on Serverless Event Stream.....</b>	<b>24</b>
<b>6 Routing Messages from OBS Application Service to Custom Connections Based on Event Subscriptions.....</b>	<b>28</b>

6.1 Overview..... 28

6.2 Procedure..... 28

6.3 Implementation Procedure..... 29

6.3.1 Creating a VPC, Subnet, and Security Group..... 30

6.3.2 Purchasing an ECS..... 30

6.3.3 Creating an OBS Bucket..... 32

6.3.4 Creating a Connection..... 32

6.3.5 Creating an Event Subscription..... 32

6.3.6 Sending an OBS Event..... 33

6.3.7 Viewing Event Messages on the ECS..... 33

# 1 EG Best Practices

This document summarizes practices in common application scenarios of EventGrid (EG). Each practice case is given detailed solution description and operation guidance, helping you easily build your services based on EG.

**Table 1-1** EG best practices

Practice	Description
<a href="#">Routing OBS Application Service Messages to DMS for Kafka</a>	This chapter describes how to use EventGrid (EG) to route the application service events of Object Storage Service (OBS) to Distributed Message Service (DMS) for Kafka.
<a href="#">Data Synchronization Between Kafka Instances Based on Serverless Event Stream</a>	This chapter describes how to use the event stream function of EG to synchronize data between DMS for Kafka instances.
<a href="#">Synchronizing OBS Bucket Data Based on Event Subscription and FunctionGraph</a>	This chapter describes how to synchronize OBS bucket data based on EG event subscription and FunctionGraph.
<a href="#">Routing Events from RocketMQ to FunctionGraph Based on Serverless Event Stream</a>	This chapter describes how to use EG to route the RocketMQ events to FunctionGraph.
<a href="#">Routing Messages from OBS Application Service to Custom Connections Based on Event Subscriptions</a>	This section describes how to use EG event subscription to send events to the connection when files in an OBS bucket are updated or added.

# 2 Routing OBS Application Service Messages to DMS for Kafka

## 2.1 Introduction

This chapter describes how to use EventGrid (EG) to route the application service events of Object Storage Service (OBS) to Distributed Message Service (DMS) for Kafka.

### Architecture

The following figure shows the event subscription architecture.



## 2.2 Resource and Cost Planning

The following table shows the resource and cost planning of this practice.

**Table 2-1** Resource and cost planning

Resource	Description	Quantity	Fee (USD)
Virtual Private Cloud (VPC)	Create a VPC.	1	00.00
VPC subnet	Create a subnet in VPC.	1	00.00

Resource	Description	Quantity	Fee (USD)
Security group	Create a security group.	1	00.00
Object Storage Service (OBS)	Create an OBS bucket. <b>NOTE</b> Creating OBS buckets is free of charge. For details, see <a href="#">OBS Pricing Details</a> .	1	00.00
Distributed Message Service (DMS) for Kafka	Buy a pay-per-use Kafka instance.	1	Example: kafka.2u4g.c luster USD0.88/ hour
EventGrid (EG)	<ul style="list-style-type: none"><li>Create an event subscription with OBS application service as the source and DMS for Kafka as the target.</li><li>Create a DMS for Kafka connection.</li></ul>	1	00.00

**NOTICE**

The fees listed here are estimates. The actual fees will be displayed on the Huawei Cloud console.

## 2.3 General Procedure

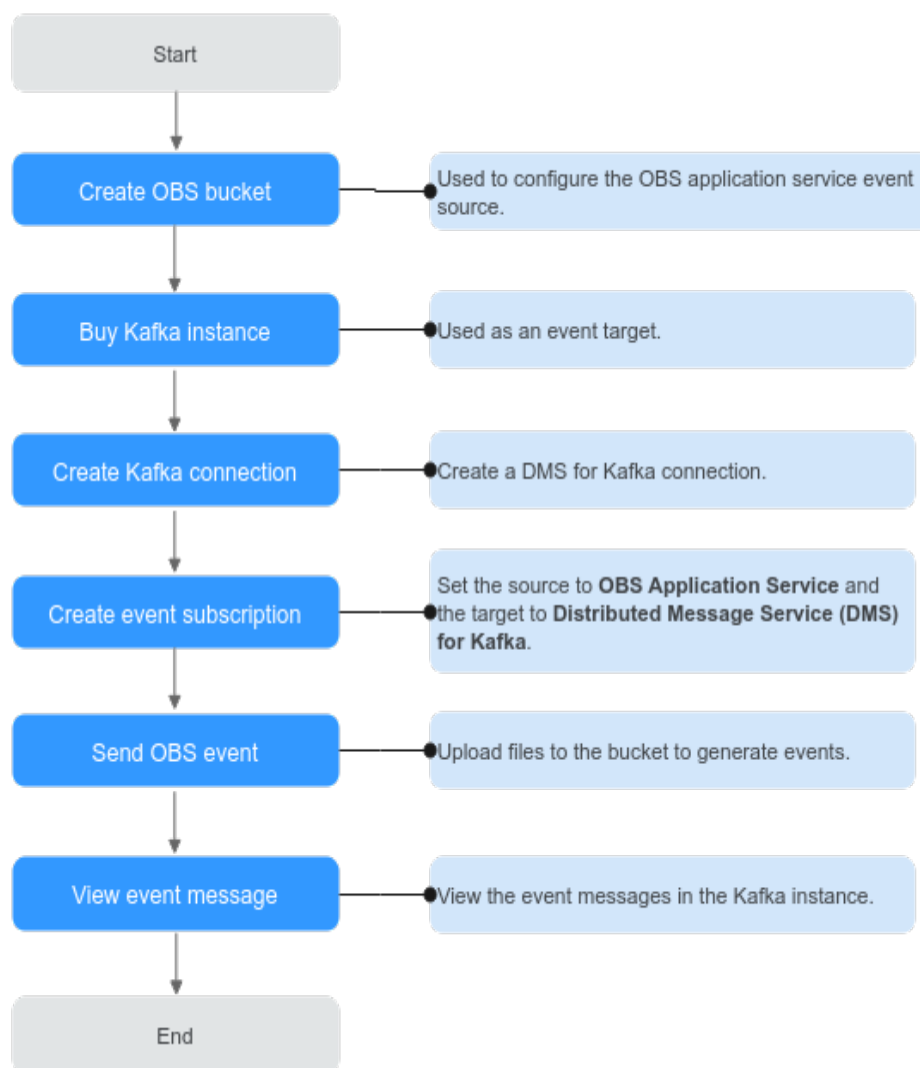
### Prerequisites

Perform the following operations before this practice:

- [Enabling EG and Authorizing Permissions](#).  
To use OBS as the event source of event subscription, you must have configured the **Tenant Administrator** permission.
- [Creating a VPC and a Subnet](#).
- You have [created a security group](#). The default security group must be configured with the ICMP protocol for the Kafka VPC's CIDR block in the inbound rule, while outbound rules must permit all ICMP traffic to destination **0.0.0.0/0**.

### General Procedure

The following figure shows the process of creating an event subscription to route events.



## 2.4 Implementation Procedure

### 2.4.1 Creating an OBS Bucket

**Step 1** Log in to the [OBS console](#).

**Step 2** Click **Create Bucket**.

**Step 3** Set bucket parameters by referring to [Figure 2-1](#). For details about these parameters, see [Creating a Bucket](#).

- **Region**: Select the region of the EG service.
- **Bucket Name**: Enter **eg-obs**.
- **Default Storage Class**: Select **Standard**.
- **Bucket Policy**: Select **Private**.
- **Direct Reading**: Select **Disable**.
- **Enterprise Project**: Select **default**.

Figure 2-1 Creating a bucket

Region: EU-Dublin

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region. Once a bucket is created, the region cannot be changed.

Bucket Name: Enter a bucket name.

Cannot be the same as that of the current user's existing buckets. Cannot be the same as that of any other user's existing buckets. Cannot be edited after creation.

Default Storage Class: Standard (High performance, reliability, and availability), Infrequent Access (High reliability, low cost, and few access), Archive (For data accessed once a year).

Bucket Policy: Private, Public Read, Public Read and Write.

Direct Reading: Enable, Disable.

Server-Side Encryption: Disable, SSE-KMS.

Enterprise Project: --Select--.

Tags: Tag key, Tag value.

**Step 4 Click Create Now.**

**----End**

## 2.4.2 Buying a Kafka Instance

### Buying an Instance

**Step 1** Log in to the [DMS for Kafka console](#) and click **Buy Kafka Instance** in the upper right corner of the page.

**Step 2** Configure the instance parameters. For details about the parameters for purchasing a Kafka instance, see [Buying an Instance](#).

- **Billing Mode:** Select **Pay-per-use**.
- **Region:** Select the region of your EG service.
- **Project:** Select the default project.
- **AZ:** Retain the default value.
- **Instance Name:** Enter **eg-kafka**.
- **Enterprise Project:** Select **default**.
- **Specifications:** Retain the default value.
- **Version:** Select a 3.x version.
- **CPU Architecture:** Select **x86**.
- **Broker Flavor:** Select **kafka.2u4g.cluster.small**.
- **Brokers:** Enter **3**.
- **Storage Space:** Select **High I/O** and set to **100 GB**.
- **Capacity Threshold Policy:** Select **Automatically delete**.

- **VPC:** Select **vpc-default** and **subnet-default**.
- **Security Group:** Select a security group from the drop-down list.  
Configure the following parameters in the **Advanced Settings** area.
- **Kafka SASL\_SSL:** Enable this function.
- **Security Protocol:** Select **SASL\_SSL**.
- **SASL/PLAIN:** Enable this function.
- **Username:** Enter **kafka-name**.
- **Password:** Enter a password.
- **Confirm Password:** Confirm new password.

**Step 3** Click **Buy**.

----End

## Creating a Topic

- Step 1** Log in to the [DMS for Kafka console](#) and select the region where the Kafka instance is located.
- Step 2** On the **DMS for Kafka** page, click the Kafka instance name to go to the details page.
- Step 3** On the **Topics** tab, click **Create Topic**.
- Step 4** Configure topic parameters by referring to [Figure 2-2](#). Set the topic name to **topic-eg**, and retain the default values for other parameters. For details about parameters for creating a topic, see [Creating a Topic](#).

**Figure 2-2** Creating a topic

**Create Topic** X

---

Topic Name

Partitions ?  Value range: 1 to 100

Replicas  Value range: 1 to 3  
Number of message copies.

Aging Time (h)  Value range: 1 to 168  
Time after which data in the topic expires.

Synchronous Replication ? ☐

Synchronous Flushing ? ☐

**Step 5** Click **OK**.

-----End

## 2.4.3 Creating a Kafka Connection

**Step 1** Log in to the [EG console](#).

**Step 2** In the navigation pane, choose **Network Management > Connections**.

**Step 3** Click **Create Connection**.

### NOTE

When you create your first connection, your authorization will be required and an agency will be automatically created. For details, see [Authorization](#).

**Step 4** Configure the connection parameters. For details about the parameters for creating a DMS for Kafka connection, see [Creating a Connection](#).

- **Type:** Select **DMS for Kafka**.
- **Name:** Enter **kafka-connect**.
- **Instance:** Select [eg-kafka](#).
- **SASL\_SSL Authentication:** Select **SCRAM-SHA-512**.

- **Username:** Enter the username **kafka-name** of this instance.
- **Password:** Enter the password of this instance.
- **Acknowledgments:** Select **Leader only**.

**Step 5** Click **OK**. If the connection status is **Normal**, the connection is successfully created.

----End

## 2.4.4 Creating an Event Subscription

### Constraints


- If you specify an object name prefix or suffix, only events with the prefix or suffix will be processed. If not specified, events of any object will be processed.
- If the selected OBS bucket is invoked by other event subscriptions, ensure that the object name prefix and suffix are different from those configured in other event subscriptions. Otherwise, an error message is displayed.

### Procedure

**Step 1** Log in to the [EG console](#).

**Step 2** In the navigation pane, choose **Event Subscriptions**.

**Step 3** Click **Create Event Subscription**.

**Step 4** Click  next to the default subscription name.

**Step 5** Enter a new subscription name and description, and click **OK**.

**Step 6** Configure an event source.

Set the following parameters:

- **Provider:** Select **Huawei Cloud**.
- **Event Source:** Select **OBS Application Service**.
- **Bucket:** Select an OBS bucket.
- **Event Type:** Select the desired event types.
- **Object Name Prefix:** Only events with this specified object prefix will be processed. By default, this field is left blank, indicating full match.
- **Object Name Suffix:** Only events with this specified object suffix will be processed. By default, this field is left blank, indicating full match.
- **Object Name Encoding:** Enable this function.
- **Filter Rule:** Retain the default value. For details about how to configure a filtering rule, see [Filter Rule Parameters](#).

**Step 7** Configure an event target.

Set the following parameters:

- **Provider:** Select **Huawei Cloud**.

- **Event Target:** Select **Distributed Message Service (DMS)** for Kafka.
- **Connection:** Select [kafka-connect](#).
- **Topic:** Select [topic-eg](#).
- **Enable:** Disable message key.
- **Transform Type:** Select **Pass-through**. For details about how to configure the transformation rule, see [Event Content Transformation](#).

**Step 8** Click **OK**.

----End

## 2.4.5 Sending an OBS Event

**Step 1** Go to the OBS console, and choose **Buckets** in the navigation pane.

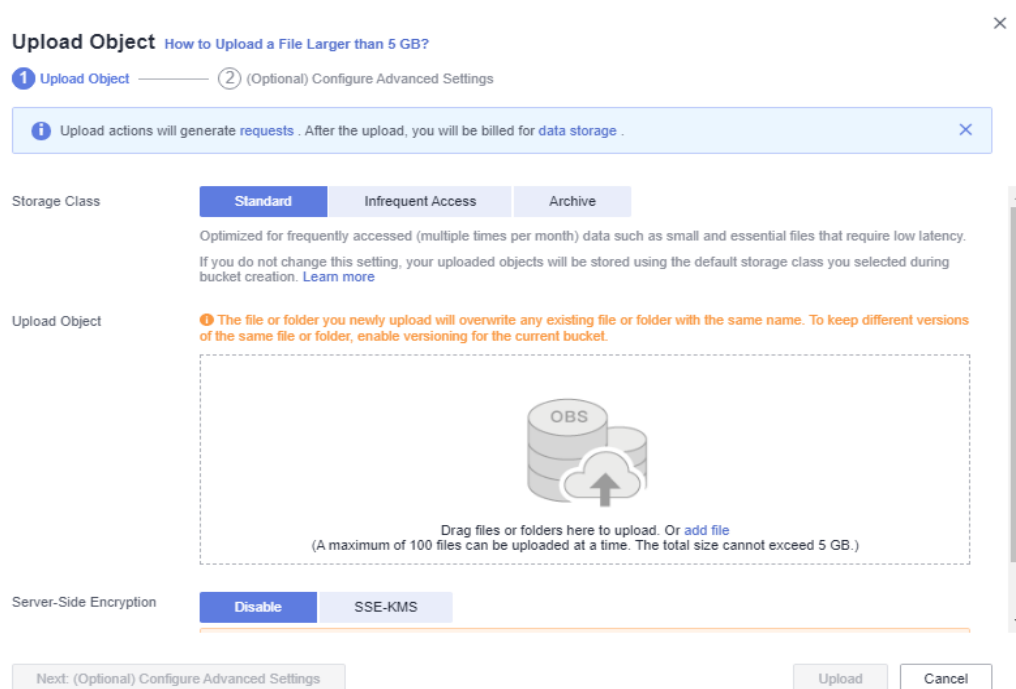
**Step 2** Click bucket [eg-obs](#).

**Step 3** On the **Objects** tab, go to the target folder, and click **Upload Object**.

**Step 4** Upload a local file and click **Upload**.

For more information about how to upload files, see [Uploading an Object](#).

**Figure 2-3** Uploading an object



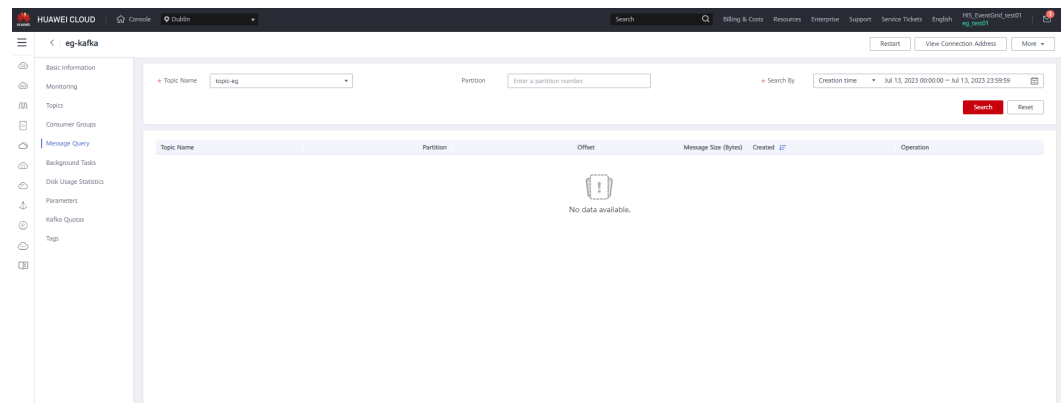
----End

## 2.4.6 Viewing Event Messages

**Step 1** Log in to the [DMS for Kafka console](#) and select the region where the Kafka instance is located.

- Step 2** On the **DMS for Kafka** page, click instance **eg-kafka** to go to the details page.
- Step 3** Click the **Message Query** tab. Then specify the topic name (**topic-eg**), partition, and search method.
- Step 4** Click **Search** to query messages.
- Step 5** Click **View Message Body** to view message details.

**Figure 2-4** Event message details



----End

# 3 Data Synchronization Between Kafka Instances Based on Serverless Event Stream

## 3.1 Overview

Event streams pull, filter, and transform events generated by event sources in real time, and route them to event targets for lightweight and efficient stream processing.

Event streams can route messages from source DMS for Kafka to target DMS for Kafka. For details, see [Event Stream Overview](#).

This chapter describes how to use the serverless event stream function of EG to synchronize data between DMS for Kafka instances.

### Prerequisites

- [EG is enabled and permissions are authorized.](#)
- [A Kafka instance is purchased.](#)
- A DMS for Kafka connection is created. For details, see [Connections](#).

#### NOTE

- The size of a single message cannot exceed 256 KB.
- No Service Level Agreement (SLA) is committed during OBT.
- Currently, cross-region deployment and Kafka-OBS (being planned) are not supported.
- On-cloud, off-cloud, and cross-cloud vendors require private lines on the user side.
- Related monitoring metrics are available since September 30, 2023.

## 3.2 Implementation and Verification

### Constraints


- Custom certificates are not supported in the event target DMS for Kafka.

- The size of a single message must be less than 2 MB.

## Configuring the Source

**Step 1** Log in to the [EG console](#).

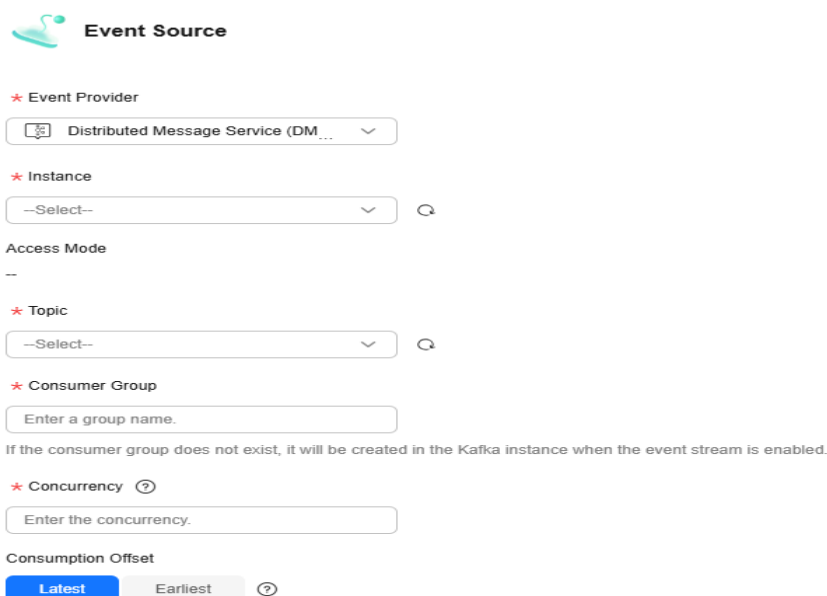
**Step 2** In the navigation pane, choose **Event Streams > Serverless Event Streams**. Click **Create Serverless Event Stream**.

**Step 3** Click  in the upper left corner to configure the event stream name and description, and click **OK**.

**Step 4** Configure a Kafka event source.

1. Click **Event Source**.
2. Select **Distributed Message Service (DMS) for Kafka** for **Event Provider**.
3. Set event source parameters.

**Figure 3-1** Kafka event source



The screenshot shows the 'Event Source' configuration page. It includes the following fields and options:

- Event Provider:** A dropdown menu with 'Distributed Message Service (DM...)' selected.
- Instance:** A dropdown menu with '--Select--' and a search icon.
- Access Mode:** A dropdown menu with '--' selected.
- Topic:** A dropdown menu with '--Select--' and a search icon.
- Consumer Group:** A text input field with the placeholder 'Enter a group name.' Below it, a note states: 'If the consumer group does not exist, it will be created in the Kafka instance when the event stream is enabled.'
- Concurrency:** A text input field with the placeholder 'Enter the concurrency.' and a help icon.
- Consumption Offset:** Two radio buttons, 'Latest' (selected) and 'Earliest', with a help icon.

**Table 3-1** Parameter description

Parameter	Description
Instance	Select a Kafka instance.
Topic	Select a topic.
Consumer Group	Enter a group name.
Concurrency	Enter the number of concurrent messages. Range: 1–1000. This parameter is autofilled with the number of partitions for the selected topic. Recommended: retain this default number for better message retrieval.

Parameter	Description
Consumption Offset	Select a consumption offset. <ul style="list-style-type: none"><li>– <b>Latest</b>: Consumption starts from the latest message in the queue.</li><li>– <b>Earliest</b>: Consumption starts from the earliest message in the queue.</li></ul>
SASL Mechanism	This parameter is available when SASL SSL authentication is enabled for the Kafka instance. Select an SASL authentication mechanism. <ul style="list-style-type: none"><li>– <b>PLAIN</b>: a simple username and password verification mechanism.</li><li>– <b>SCRAM-SHA-512</b>: uses the hash algorithm to generate credentials for usernames and passwords to verify identities. SCRAM-SHA-512 is more secure than PLAIN.</li></ul>
SASL Certificate URL	This parameter is available when SASL SSL authentication is enabled for the Kafka instance. Enter an SASL certificate URL. For details about the URL, see <a href="#">How Do I Obtain the SASL Certificate Address of a DMS for Kafka Instance?</a> <ul style="list-style-type: none"><li>– The package must be in ZIP format. The number of files in the package cannot exceed two. The size of the package and file cannot exceed 1 MB.</li><li>– The certificate name must be fixed to <b>client.jks</b>.</li></ul>
SASL Certificate Key	This parameter is available when SASL SSL authentication is enabled for the Kafka instance. Enter an SASL certificate key.
Username	This parameter is available when SASL SSL authentication is enabled for the Kafka instance. Enter a username.
Password	This parameter is available when SASL SSL authentication is enabled for the Kafka instance. Enter a password.

**Step 5** Click **Save**.

----End

## Configuring the Target

**Step 1** Configure an event target.

1. Click **Event Target**.
2. Select **Distributed Message Service (DMS) for Kafka** for **Target**.
3. Set event target parameters.

**Figure 3-2** Distributed Message Service (DMS) for Kafka

The screenshot shows the 'Event Target' configuration page for 'Distributed Message Service (DMS) for Kafka'. The interface includes several sections:

- Target:** A dropdown menu showing 'Distributed Message Service (DM ...)'.
- Connection:** A dropdown menu showing '--Select--' and a 'Create Connection' link. Below it, a note states 'Only Kafka connections displayed here.'
- Topic:** A dropdown menu showing '--Select--' and a search icon. Below it, a note states 'Select a connection first.'
- Message Key:** A section with an 'Enable' toggle switch currently turned off.
- Rule:** A section with a 'Transform Type' dropdown showing 'Pass-through' (selected), 'Variables', and 'Constants'. Below it, a note states 'Route all content of events to the target.'
- Message Push:** A section with a 'Batch Push' toggle switch currently turned off.

At the bottom right, there are 'Previous' and 'OK' buttons.

**Table 3-2** Distributed Message Service (DMS) for Kafka parameters

Parameter	Description
Connection	Select a connection. If no connection is available, <a href="#">create one with DMS for Kafka</a> .
Topic	First select a connection, and then select a topic.
<b>Message Key</b>	
Disable	<b>Message Key</b> is not enabled.
Enable	<b>Variable:</b> The key is a variable value from CloudEvents-compliant events. <b>Constant:</b> The key is a specified constant. All messages will be sent to the same partition.
<b>Rule</b>	

Parameter	Description
Transform Type	<b>Variables:</b> <i>data.value</i> For more information about transformation types, see <a href="#">Event Content Transformation</a> .
<b>Message Push</b>	
Batch Push	Specify whether to enable <b>Batch Push</b> to aggregate multiple events.
Messages Interval (s)	The maximum number of aggregated records that can be pushed at a time. Default: <b>100</b> . Range: 1–10,000. This parameter is available only when <b>Batch Push</b> is enabled.  The interval between batch pushes, in seconds. Default: <b>1</b> . Range: 0–15. This parameter is available only when <b>Batch Push</b> is enabled.

**Table 3-3** Batch push example

Messages	Interval (s)	Effect
100	1s	100 messages pushed every second.
200	5s	500 messages pushed every 5 seconds.

**Step 2** After the configuration is complete, click **OK**.

**Step 3** Click **Enable** in the operation column of **Event Streams**. If the **Status** changes to **Running**, the link is successfully created.

----End

## Performing Verification

**Step 1** Log in to the [DMS for Kafka console](#) and select the region where the Kafka instance is located.

**Step 2** On the **DMS for Kafka** page, click the Kafka instance name to go to the details page.

**Step 3** Choose **Message Query** and query the **Message Body** of the source topic and target topic.

Figure 3-3 Target message

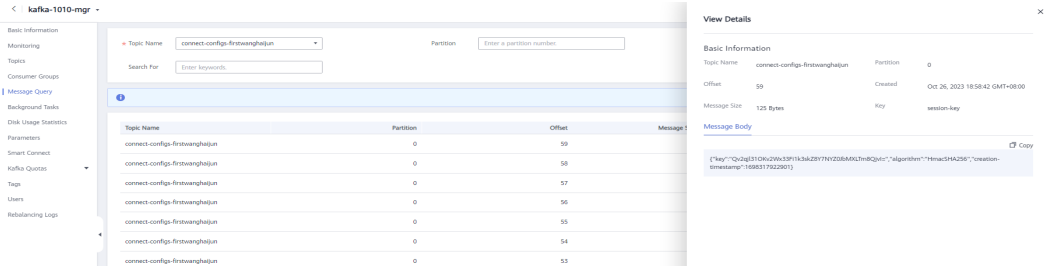
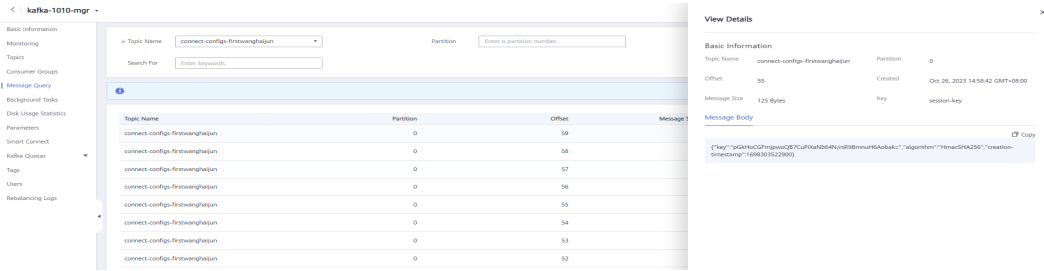


Figure 3-4 Source message



**Step 4** Check whether the messages are consistent. If yes, data synchronization between the Kafka instances is successful.

----End

# 4 Synchronizing OBS Bucket Data Based on Event Subscription and FunctionGraph

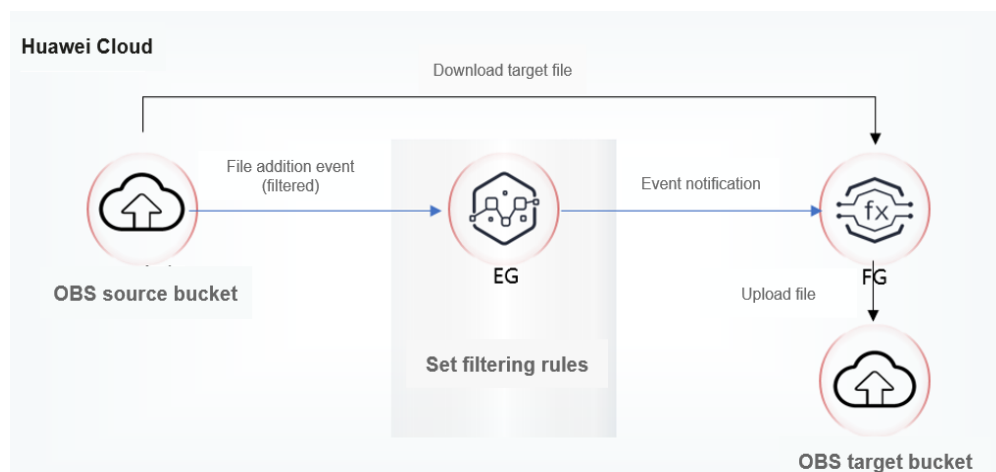
## 4.1 Overview

This chapter describes how to synchronize OBS bucket data based on EG event subscription and FunctionGraph.

### Architecture

The following figure shows the event subscription architecture.

**Figure 4-1** Event subscription architecture



## 4.2 Procedure

### Prerequisites

- **EG is enabled and permissions are authorized.**

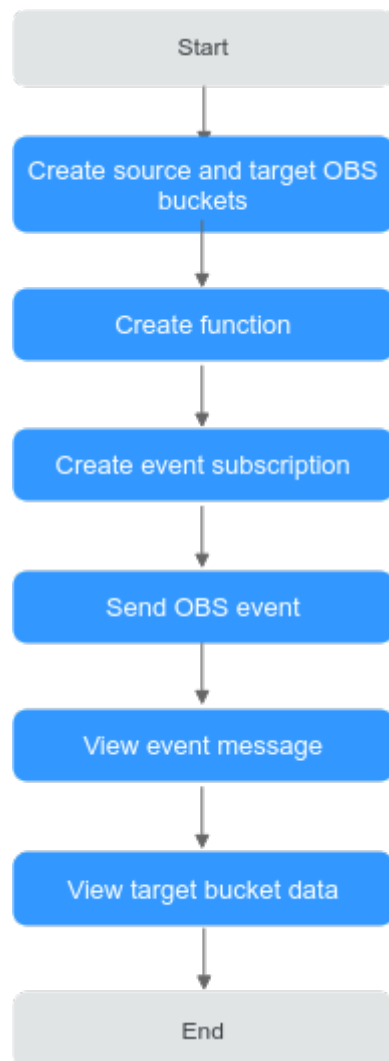
To use OBS as the event source of event subscription, you must have configured the **Tenant Administrator** permission.

- FunctionGraph has been enabled and authorized.
- OBS has been enabled and authorized.
- The account has been added to the OBS whitelist. (Contact the OBS service.)

## General Procedure

The following figure shows the process of creating an event subscription to route events.

**Figure 4-2** Flowchart



## 4.3 Implementation Procedure

## 4.3.1 Creating an OBS Source Bucket and a Target Bucket

**Step 1** Log in to the [OBS console](#).

**Step 2** Click **Create Bucket**.

**Step 3** Set bucket parameters. For details about these parameters, see [Creating a Bucket](#).

**Table 4-1** Bucket parameters

Parameter	Description
Region	Select the region of your EG service.
Bucket Name	For example, <b>eg-test</b> .
Storage Class	Configure as needed.
Bucket Policy	Configure as needed.
Server-Side Encryption	Configure as needed.
Direct Reading	Configure as needed.
Enterprise Project	Configure as needed.

**Step 4** Click **Create Now**.

**Step 5** Create the source and target bucket based on the preceding procedure.

----End

## 4.3.2 Creating a Function

**Step 1** Download the image compression program package. For details, see [Compressing Images Using a Function](#).

**Step 2** Create a function. (Specify an agency with OBS access permissions so that FunctionGraph can invoke the OBS service.)

1. Log in to the [FunctionGraph console](#), and choose **Functions > Function List** in the navigation pane.
2. Click **Create Function**.
3. Set the function information and click **Create Function**. For details, see [Creating a Function](#).
4. Click the created function.
  - On the **Code** tab, choose **Upload > Local ZIP**, upload the program package.
  - Choose **Configuration > Environment Variables**, set environment variables, and click **Save**.

**Table 4-2** Parameter description

Parameter	Description
obs_endpoint	The bucket address parameter defined in <b>index.py</b> . Set the value to <b>obs.region.myhuaweicloud.com</b> .
output_bucket	The output bucket parameter defined in <b>index.py</b> . Set the value to <b>your-bucket-output</b> , the bucket created in <a href="#">Creating OBS Buckets</a> .

**Step 3** Add a dependency.

The sample code depends on the Pillow package, which needs to be imported as a dependency. The procedure is as follows:

1. Go to the details page of the created function, click the **Code** tab, and click **Add** in the **Dependencies** area at the bottom.
2. Add the public dependency **pillow-7.1.2**. The default version is **1**. Click **OK**.


----End

### 4.3.3 Creating an Event Subscription

#### Constraints

- If you specify an object name prefix or suffix, only events with the prefix or suffix will be processed. If not specified, events of any object will be processed.
- If the selected OBS bucket is invoked by other event subscriptions, ensure that the object name prefix and suffix are different from those configured in other event subscriptions. Otherwise, an error message is displayed.

#### Procedure

- Step 1** Log in to the [EG console](#).
- Step 2** In the navigation pane, choose **Event Subscriptions**.
- Step 3** Click **Create Event Subscription**.
- Step 4** Click  next to the default subscription name.
- Step 5** Enter a new subscription name and description, and click **OK**.
- Step 6** Configure an event source.

**Table 4-3** Parameter description

Parameter	Description
Provider	Select <b>Cloud services</b> .
Event Source	Select <b>OBS Application Service</b> .
Bucket	Select an OBS bucket.
Event Type	Select the desired event types.
Object Name Prefix	Only events with this specified object suffix will be processed.
Object Name Suffix	Only events with this specified object suffix will be processed.
Object Name Encoding	Enable it.
Filter Rule	Retain the default. For details about how to configure a filtering rule, see <a href="#">Filter Rule Parameters</a> .

**Step 7** Configure an event target.**Table 4-4** Parameter description

Parameter	Description
Provider	Select <b>Cloud services</b> .
Event Target	Select <b>FunctionGraph (function computing)</b> .
Function	Select the function created for OBS data synchronization.
Version	Configure as needed.
Execute	Configure as needed.
Agency	If no agency is available, click <b>Create Agency</b> to generate one named <b>EG_TARGET_AGENCY</b> .
Transform Type	Configure as needed.

----End

## 4.3.4 Sending an OBS Event

**Step 1** Events supported by OBS are listed as follows:

Figure 4-3 Supported event types

☐ **OBS:DWR:ObjectCreated:PUT**  
Create or override bucket objects via UI or Put request

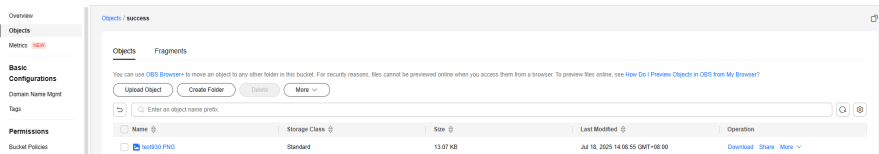
☐ **OBS:DWR:ObjectCreated:POST**  
Create or override bucket objects via Post request

☐ **OBS:DWR:ObjectCreated:COPY**  
Create or override bucket objects via Copy request

☐ **OBS:DWR:ObjectCreated:CompleteMultipartUpload**  
Merge parts via UI or API request

**Step 2** After an event (such as uploading and overwriting) is triggered in OBS, it can be routed based on the event types selected in the event subscription. For example, upload the **test930.png** file to the **success** directory in the **test\_zzy** OBS bucket.

Figure 4-4 OBS bucket list



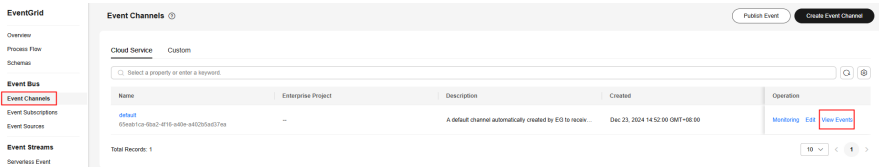
-----End

### 4.3.5 Viewing Event Delivery Details

**Step 1** On the EG console, choose **Event Channels**, and click **View Events** of the corresponding event channel to view the delivery and event details.

- 1. View events

Figure 4-5 Viewing events



- 2. View event details

Figure 4-6 Viewing event details

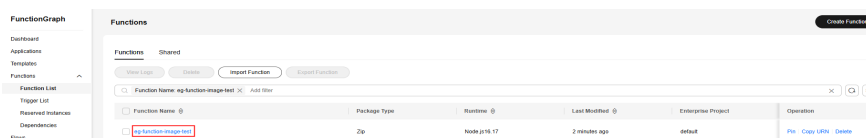
```
1 {
2   "datacontenttype": "application/json",
3   "data": {
4     "obs": {
5       "bucket": {
6         "bucket": "bucket-input-my",
7         "name": "bucket-input-my",
8         "arn": "",
9         "ownerIdentity": {
10          "ID": "f9e40463c23845438ca9efd3a7ec854e"
11        },
12        "Version": "1.0",
13        "configurationId": "483ec214-c662-47f8-9f5b-b43c86820bbb",
14        "object": {
15          "versionId": "G001019242159AB9FFFFD2C701236A9Cnull",
16          "oldpsxpth": "",
17          "size": 10,
18          "eTag": "cf9cc90a0dd26a12b83400704bf3372a",
19          "key": "success/test.txt",
20          "sequencer": "1"
21        }
22      }
23    }
24  }
```

----End

## 4.3.6 Viewing Code Execution Details in FunctionGraph

**Step 1** In the function list, find the function for data synchronization.

Figure 4-7 Function list



**Step 2** Click the corresponding request in the monitoring to view the code log.

----End

## 4.3.7 Checking the Synchronization Result

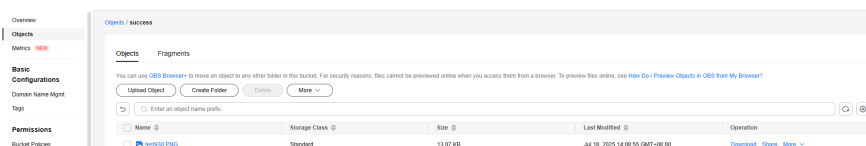
By default, The file path in the source bucket and the synchronized file path in the target bucket are the same. Check whether the file is synced successfully in the target bucket object list.

**Step 1** Log in to the [OBS console](#).

**Step 2** On the **Buckets** page, click the target bucket name.

**Step 3** Check whether the objects have been synchronized.

Figure 4-8 Target bucket directory



----End

# 5 Routing Events from RocketMQ to FunctionGraph Based on Serverless Event Stream

---

This chapter describes how to use EG to route the RocketMQ events to FunctionGraph.


## Prerequisites

Perform the following operations before this practice:

- [Enabling EG and Authorizing Permissions.](#)
- [Creating a VPC.](#)
- [Creating a Security Group.](#)
- [Buying a RocketMQ Instance.](#)
- [Creating an Event Function.](#) Use Python 3.9 as an example.

## Creating a Serverless Event Stream

**Step 1** Log in to the [EG console](#). In the navigation pane, choose **Event Streams > Serverless Event Streams**. In the upper right corner, click **Create Serverless Event Stream**.

**Step 2** Click  in the upper left corner to configure the event stream name and description, and click **OK**.

**Step 3** [Configure a RocketMQ event source.](#)

**Step 4** Configure [event rules](#). In this example, the default configuration is used.

**Step 5** [Configure the FunctionGraph event target.](#)

In the **Rule** area, set **Transform Type** to **Variables**. The following are examples for **Parameters** and **Template**.

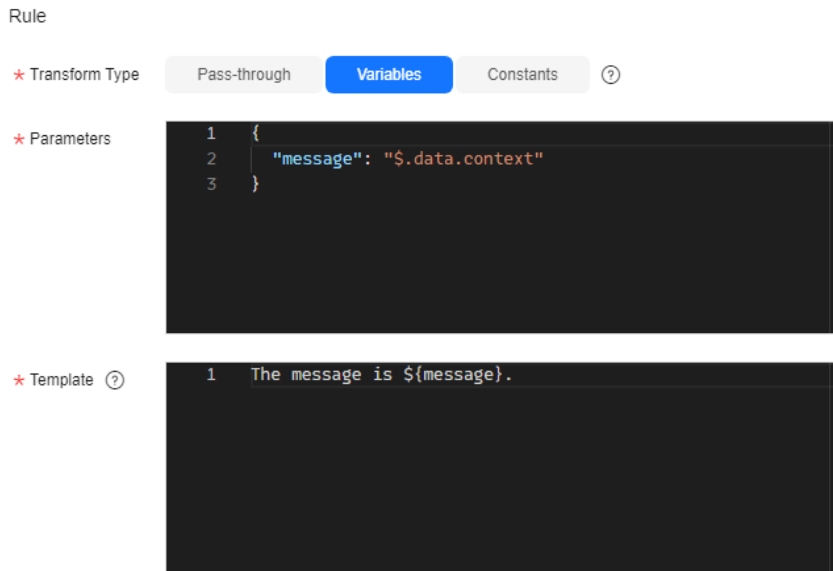
Parameter:

```
{
  "message": "${data.context}"
}
```


Template:

The message is \${message}.

**Figure 5-1** Configure variables



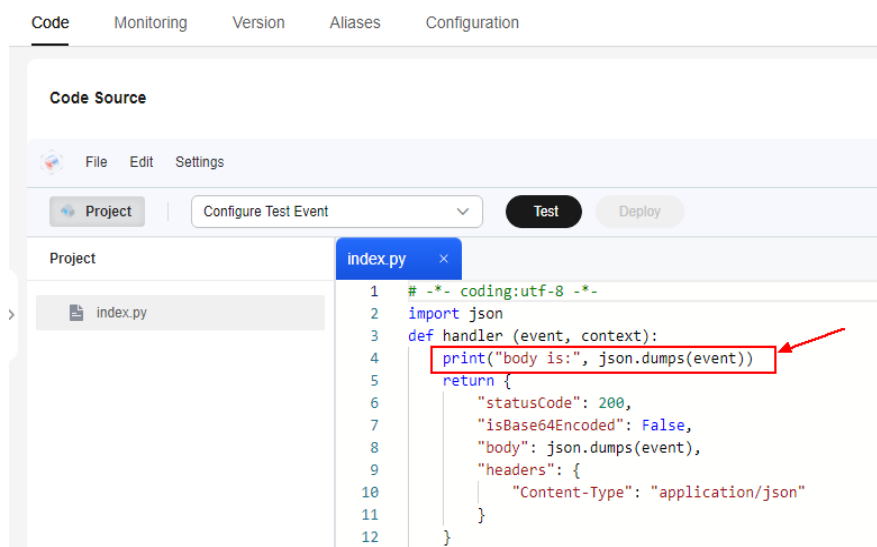
**Step 6** After the event stream is configured, click **Save** in the upper right corner and click **Enable** in the **Operation** column of the event stream list.

**Step 7** Click  in the upper left corner, search for **FunctionGraph**, and go to the FunctionGraph console. In the navigation pane on the left, choose **Functions** > **Function List**, and click the name of the created function to go to the function details page.

**Step 8** Click the **Code** tab, add the following code to the editing area, and click **Deploy** to refresh the code.

```
print("body is:", json.dumps(event))
```

Figure 5-2 Editing the code



----End

## Publishing Events from RocketMQ to FunctionGraph


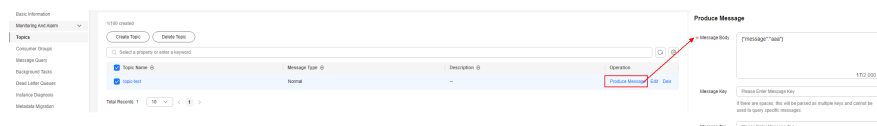
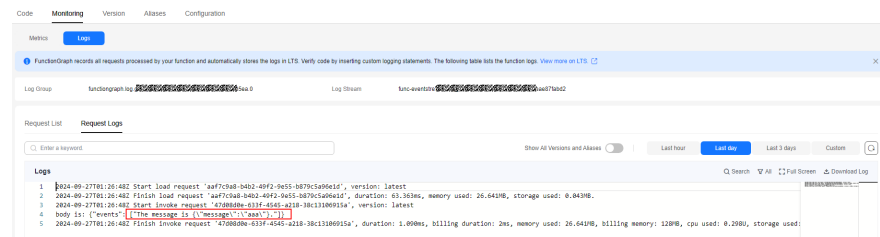
- Step 1** Click  in the upper left corner, search for **rocketmq**, and access the Distributed Message Service (for RocketMQ) console. In the navigation pane on the left, choose **RocketMQ Instances**. Click the name of the created instance to go to its details page.
- Step 2** In the navigation pane on the left, choose **Topics**. Click **Produce Message** of the created topic. In the **Message Body** box, enter **{"message":"aaa"}**. Retain the default values for other parameters and click **OK**.

Figure 5-3 Producing a message



- Step 3** Return to the FunctionGraph console, choose **Functions > Function List** in the navigation pane, and click a function name to go to the function details page.
- Step 4** Choose **Monitoring > Logs > Request Logs** to view the event content pushed from the RocketMQ instance.

### Figure 5-4 Viewing logs



---End

# 6 Routing Messages from OBS Application Service to Custom Connections Based on Event Subscriptions

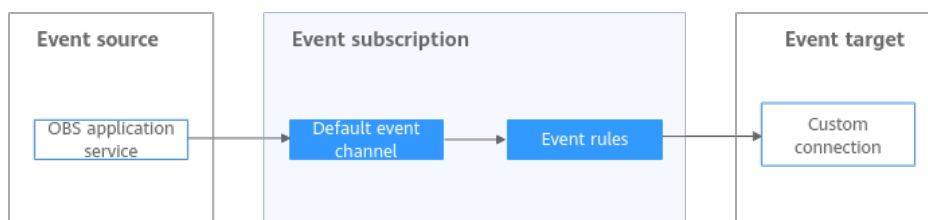
## 6.1 Overview

This section describes how to use EG event subscription to send events to the connection when files in an OBS bucket are updated or added.

### Architecture

The following figure shows the event subscription architecture.

**Figure 6-1** Event subscription architecture



## 6.2 Procedure

### Constraints

When creating service resources, ensure that the resources are in the same region.

### Prerequisites

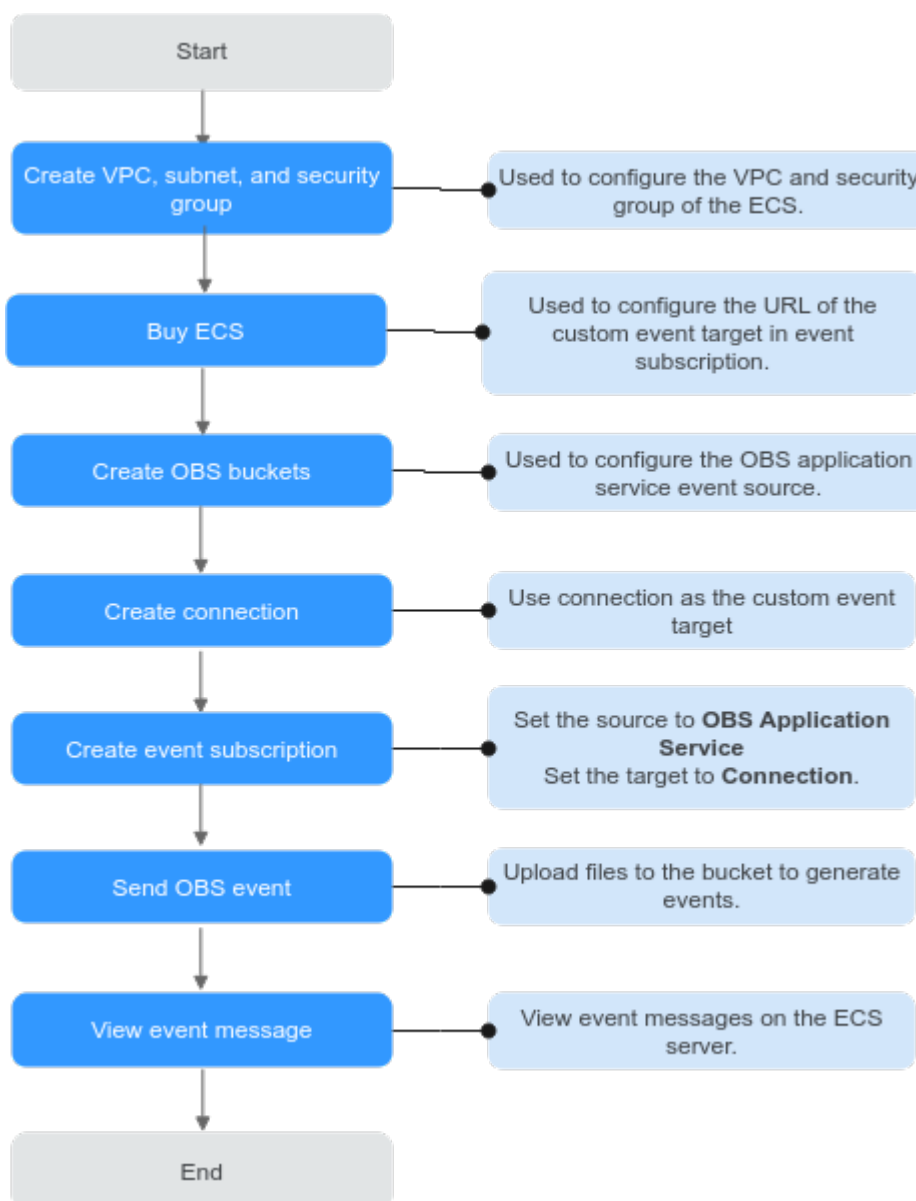
- **EG is enabled and permissions are authorized.**  
To use OBS as the event source of event subscription, you must have configured the **Tenant Administrator** permission.
- OBS has been enabled and authorized.

- ECS has been enabled and authorized.
- VPC has been enabled and authorized.

## General Procedure

The following figure shows the process of creating an event subscription to route events.

**Figure 6-2** Flowchart

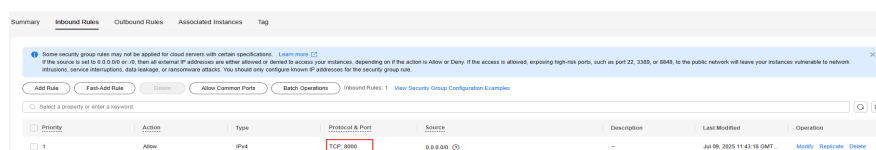


## 6.3 Implementation Procedure

## 6.3.1 Creating a VPC, Subnet, and Security Group

- Step 1** Log in to the [VPC console](#).
- Step 2** In the navigation pane on the left, choose **My VPCs**. Click **Create VPC** in the upper right corner. For details, see [Creating a VPC and Subnet](#).
- Step 3** In the navigation pane on the left, choose **Security Groups**. Click **Create Security Group** in the upper right corner. For details, see [Creating a Security Group](#). In addition, configure a rule to allow access to port **8000** in the inbound rule, as shown in [Figure 6-3](#).

**Figure 6-3** Allowing access to port 8000



----End

## 6.3.2 Purchasing an ECS

- Step 1** Log in to the [ECS console](#).
- Step 2** In the navigation pane on the left, choose **Elastic Cloud Server**. In the upper right corner, click **Buy ECS**. For details, see [Buying an ECS](#). Select the VPC, subnet, and security group created in [Creating a VPC, Subnet, and Security Group](#).
- Step 3** Return to the ECS list and click **Remote Login** in the **Operation** column.

**Figure 6-4** Remote login



- Step 4** In the displayed dialog box, click **Log In** and enter the password to access the CloudShell terminal console.

Ensure that the security group allows access from the port (**22** by default) used for CloudShell logins.

- Step 5** In the **usr** directory, run the **mkdir** and **openssl** commands to create the **local/test** directory and generate the key certificate in the **test** directory.

- Run the following command to create the **local/test** directory in the **usr** directory:  

```
mkdir -p local/test
```
- Run the following command to go to the **test** directory and generate the key certificate:  

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
```
- After the command in step **2** is executed, set **PEM PASS** to **privatenetwork**. After the setting is complete, enter the related information as prompted.

Figure 6-5 Generating the key certificate

```
[root@ecs-xxxxxxxx test]# openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365
Generating a RSA private key
.....+++++
writing new private key to 'key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:qu
State or Province Name (full name) [Some-State]:qu
Locality Name (eg, city) []:qu
Organization Name (eg, company) [Internet Widgits Pty Ltd]:qu
Organizational Unit Name (eg, section) []:qu
Common Name (e.g. server FQDN or YOUR name) []:qu
Email Address []:qu
[root@ecs-xxxxxxxx test]# ll
total 12
-rw-r--r-- 1 root root 1322 Oct 14 15:54 cert.pem
-rw----- 1 root root 1854 Oct 14 15:54 key.pem
[root@ecs-xxxxxxxx test]#
```

**Step 6** In the **test** directory, use Python 3 to set up a Python server.

Run the following command to create a python file in the **test** directory:

```
vi httpserver.py
```

Enter the following content into the python file:

```
from http.server import HTTPServer, BaseHTTPRequestHandler

from io import BytesIO
import ssl

class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):

    def do_GET(self):
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b'Hello, world!')

    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        body = self.rfile.read(content_length)
        self.send_response(200)
        self.end_headers()
        response = BytesIO()
        response.write(b'This is POST request. ')
        response.write(b'Received: ')
        response.write(body)
        print(body)
        self.wfile.write(response.getvalue())

httpd = HTTPServer(('0.0.0.0', 8000), SimpleHTTPRequestHandler)
httpd.socket = ssl.wrap_socket(httpd.socket,
                               keyfile="key.pem",
                               certfile='cert.pem', server_side=True)
httpd.serve_forever()
```

**Step 7** Install Python 3 and configure environment variables. Run the following command to start the **httpserver**.

```
python3 httpserver.py
```

----End

### 6.3.3 Creating an OBS Bucket

**Step 1** Log in to the [OBS console](#).

**Step 2** In the navigation pane on the left, choose **Buckets**. In the upper right corner, click **Create Bucket**. For details, see [Creating a Bucket](#).

**Step 3** Click **Create Now**.

----End

### 6.3.4 Creating a Connection

**Step 1** Log in to the [EG console](#).

**Step 2** In the navigation pane on the left, choose **Network Management > Connections**. In the upper right corner, click **Create Connection**. For more information about connections, see [Connections](#).

- **Type:** Select **WEBHOOK**.
- **Name:** Enter a custom name.
- **Description:** Enter a description.
- **VPC:** Select the VPC created in [Creating a VPC, Subnet, and Security Group](#).
- **Subnet:** Select the subnet created in [Creating a VPC, Subnet, and Security Group](#).

**Step 3** Click **OK**. The connection is created.

----End

### 6.3.5 Creating an Event Subscription


#### Constraints

- If you specify an object name prefix or suffix, only events with the prefix or suffix will be processed. If not specified, events of any object will be processed.
- If the selected OBS bucket is invoked by other event subscriptions, ensure that the object name prefix and suffix are different from those configured in other event subscriptions. Otherwise, an error message is displayed.

#### Procedure

**Step 1** Log in to the [EG console](#).

**Step 2** In the navigation pane on the left, choose **Event Bus > Event Subscriptions**. Then, click **Create Event Subscription** in the upper right corner.

**Step 3** On the event subscription editing page, click  in the upper left corner, enter the subscription name and description, and click **OK**.

**Step 4** Configure the event source. The event source parameters are as follows. For more information about event subscriptions, see [Creating an Event Subscription](#).

1. **Provider:** Select **Cloud services**.

2. **Event Source:** Select **OBS Application Service**.
3. **Bucket:** Select the bucket created in [Creating an OBS Bucket](#).
4. **Event Type:** Select the desired event types.
5. **Object Name Prefix:** If this parameter is set, an event can be triggered only when the name matches the specified prefix. By default, this parameter is left blank, indicating that all objects are matched.
6. **Object Name Suffix:** If this parameter is set, an event can be triggered only when the name matches the specified suffix. By default, this parameter is left blank, indicating that all objects are matched.
7. **Object Name Encoding:** This function is enabled by default.
8. **Filter Rule:** For details about how to configure a filtering rule, see [Filter Rule Parameters](#).

**Step 5** Configure the event target. The parameters are as follows:

1. **Provider:** Select **Custom**.
2. **URL:** Enter the IP address of the ECS instance created in [Purchasing an ECS](#). The URL format is **https://x.x.x.x:8000**.
3. **Connection:** Select the connection created in [Creating a Connection](#).

Retain the default values for other parameters.

**Step 6** After all parameters are configured, click **Save** in the upper right corner. The event subscription is created.

----End

## 6.3.6 Sending an OBS Event

**Step 1** Log in to the [OBS console](#).

**Step 2** In the navigation pane on the left, choose **Buckets**. Click the name of the bucket created in [Creating an OBS Bucket](#). The object details page is displayed.

**Step 3** On the object details page, choose **Objects** in the navigation pane and click **Upload Object**. The **Upload Object** dialog box is displayed.

**Step 4** Upload a local file and click **Upload**. For details about how to upload an object file, see [Uploading an Object](#).

----End

## 6.3.7 Viewing Event Messages on the ECS

**Step 1** Return to the CloudShell terminal console page opened in [Purchasing an ECS](#).

**Step 2** Since **httpsserver.py** has been started, CloudShell will print event information.

[illegible]

**----End**