

# Event Bridge

# Event Source

# Product Documentation



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Event Source

Overview

Tencent Cloud Service Event Source

Event Structure

## Connector

Overview

Configuring APIGW Connector

Configuring CKafka Connector

Configuring a DTS Connector

Configuring TDMQ Connector

# Event Source

## Overview

Last updated : 2024-07-23 15:08:07

Event sources publish produced events complying with CloudEvents - Version 1.0 to EventBridge. For more information, see [Event Structure](#) .

EventBridge supports the following event source connection methods:

## Tencent Cloud services

If you want to use the **Cloud Monitor events** and **CloudAudit events** generated by Tencent Cloud services as event sources, you only need to activate the corresponding Tencent Cloud services, and then the Tencent Cloud services will be automatically connected to EventBridge. By configuring the predefined event pattern and targets, you can publish events to the Tencent Cloud service event bus with ease. Then the events are filtered based on the event pattern and routed to the event targets. Currently, EventBridge supports the delivery and management of Cloud Monitor events and CloudAudit events. For more information, see [Tencent Cloud Service Event](#).

## Custom applications

If you want to connect a custom business as an event source, you need to configure your application for connection to EventBridge through the corresponding API/SDK. By creating a custom event bus and configuring a custom event pattern and event target, you can publish events generated by your application to the custom event bus, use the custom event pattern to filter them, and route them to the custom event target. For more information about connectors, see [Custom Event](#).

## Connector

A connector is used to proactively pull events from event sources such as **message queue TDMQ** and push them to a custom event bus in **standard format**. With it, you don't need to care about the underlying consumption delivery logic. You can bind one or more connectors to a custom event bus to automatically pull event content from message queues and gateways and push the event content to the specified event bus. For more information about connectors, see [Connector Overview](#) .

# Tencent Cloud Service Event Source

Last updated : 2024-07-23 15:08:07

## Note:

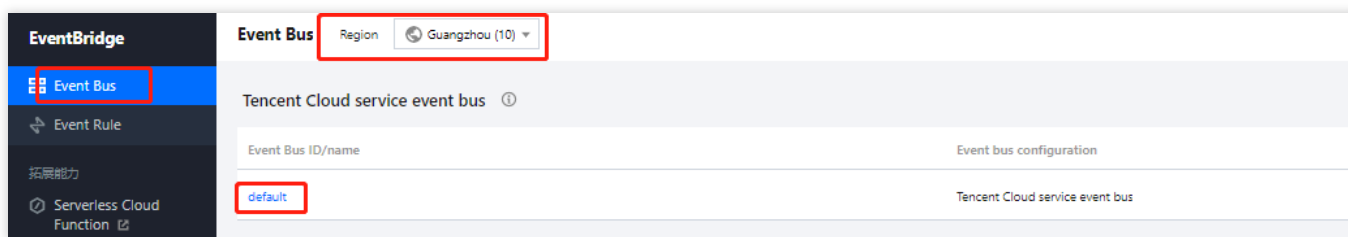
All Ops events such as alarms and auditing events generated by Tencent Cloud services will be delivered to the Tencent Cloud service event bus, which is the default event bus and cannot be modified or deleted. You can go to the EventBridge console to bind rules and targets to the Tencent Cloud service event bus to distribute Tencent Cloud service events.

If you want to use the **Cloud Monitor events** (such as CVM kernel faults and OOM exceptions) and **CloudAudit events** (available soon) that are generated by Tencent Cloud services as event sources, you only need to activate the corresponding Tencent Cloud services, and then the Tencent Cloud services will be automatically connected to EventBridge. The events generated by Tencent Cloud services will be delivered to the **Tencent Cloud service event bus** by default. For more information, see [Tencent Cloud Service Event](#).

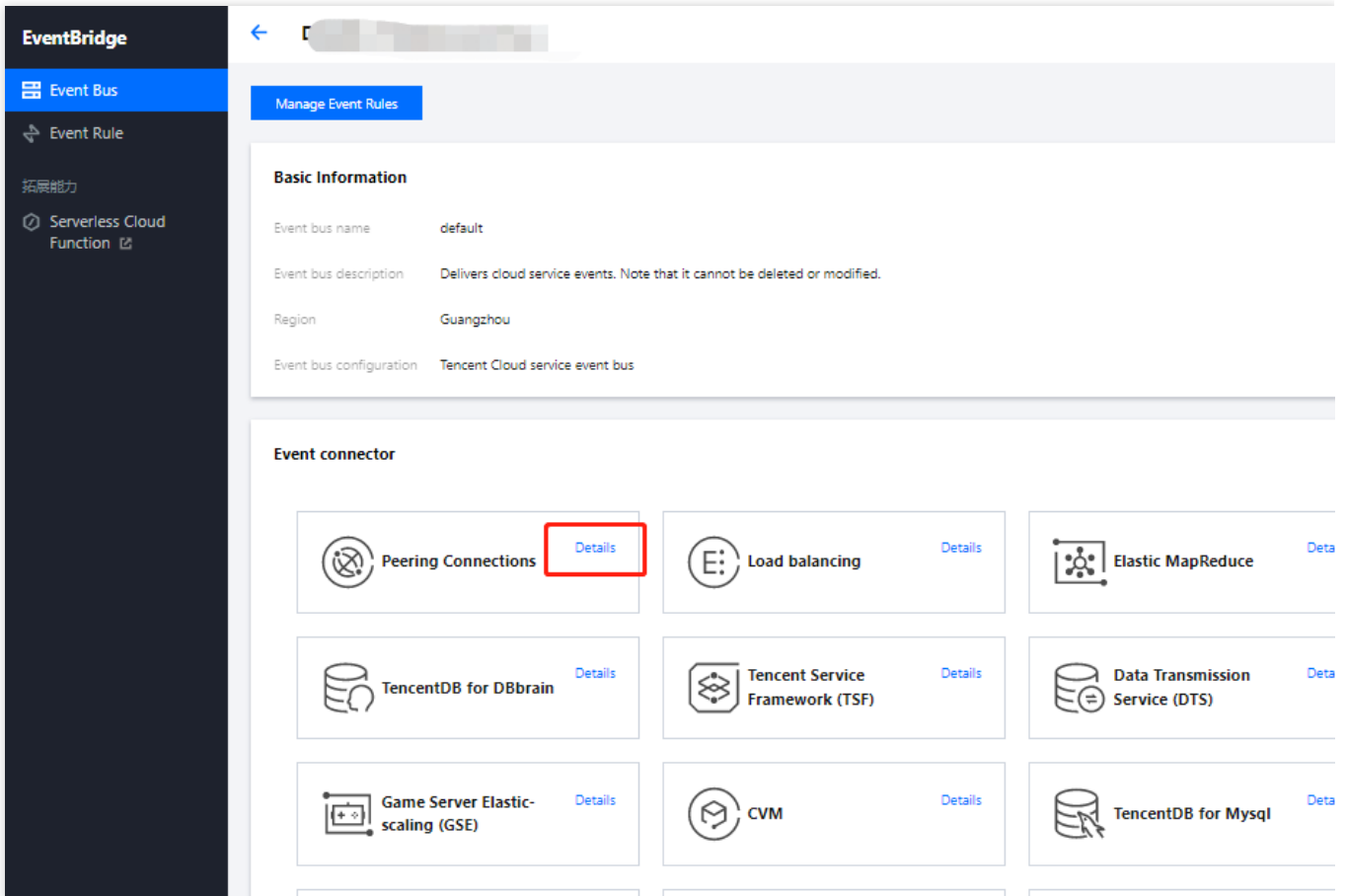
## Directions

The following is the procedure for viewing all Tencent Cloud services (Cloud Monitor) data sources currently supported:

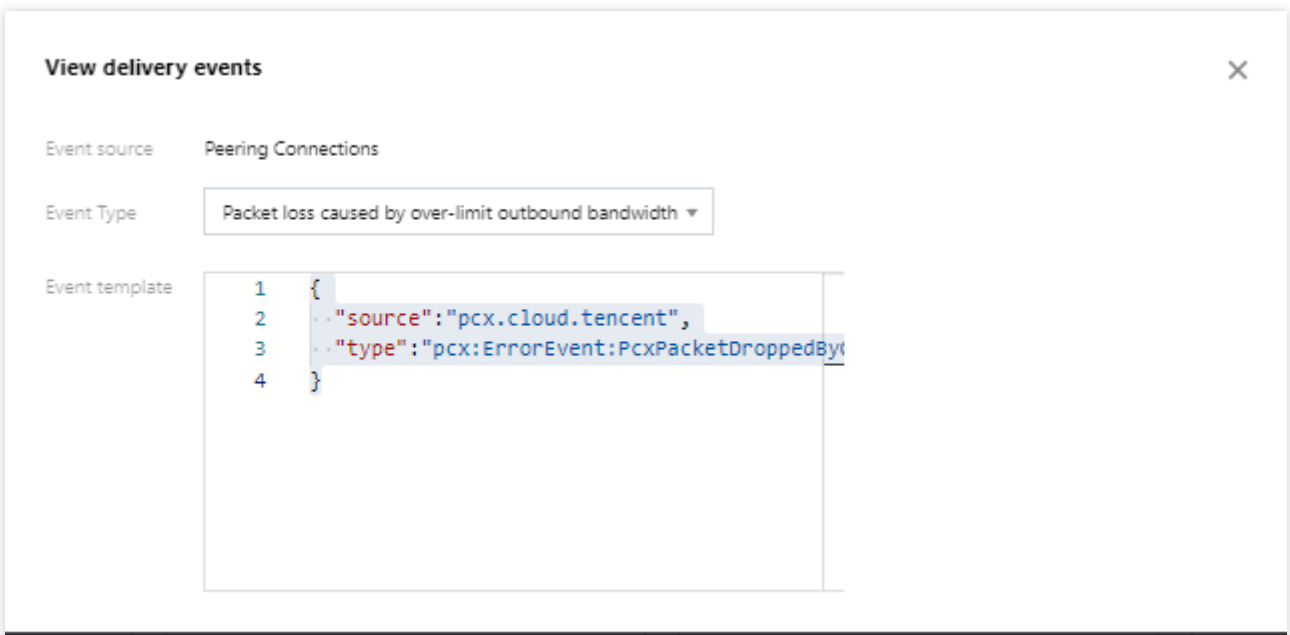
1. Log in to the [EventBridge console](#) and choose **Tencent Cloud service event bus** > **default** to view the Tencent Cloud service events that are currently connected to the Tencent Cloud service event bus.



2. In the **Event Connector** area, you can view all Tencent Cloud services that support alarm event push.



3. You can click **Details** to view all alarm event types that are currently supported.



## Supported Event Sources

[Cloud Monitor Event](#)

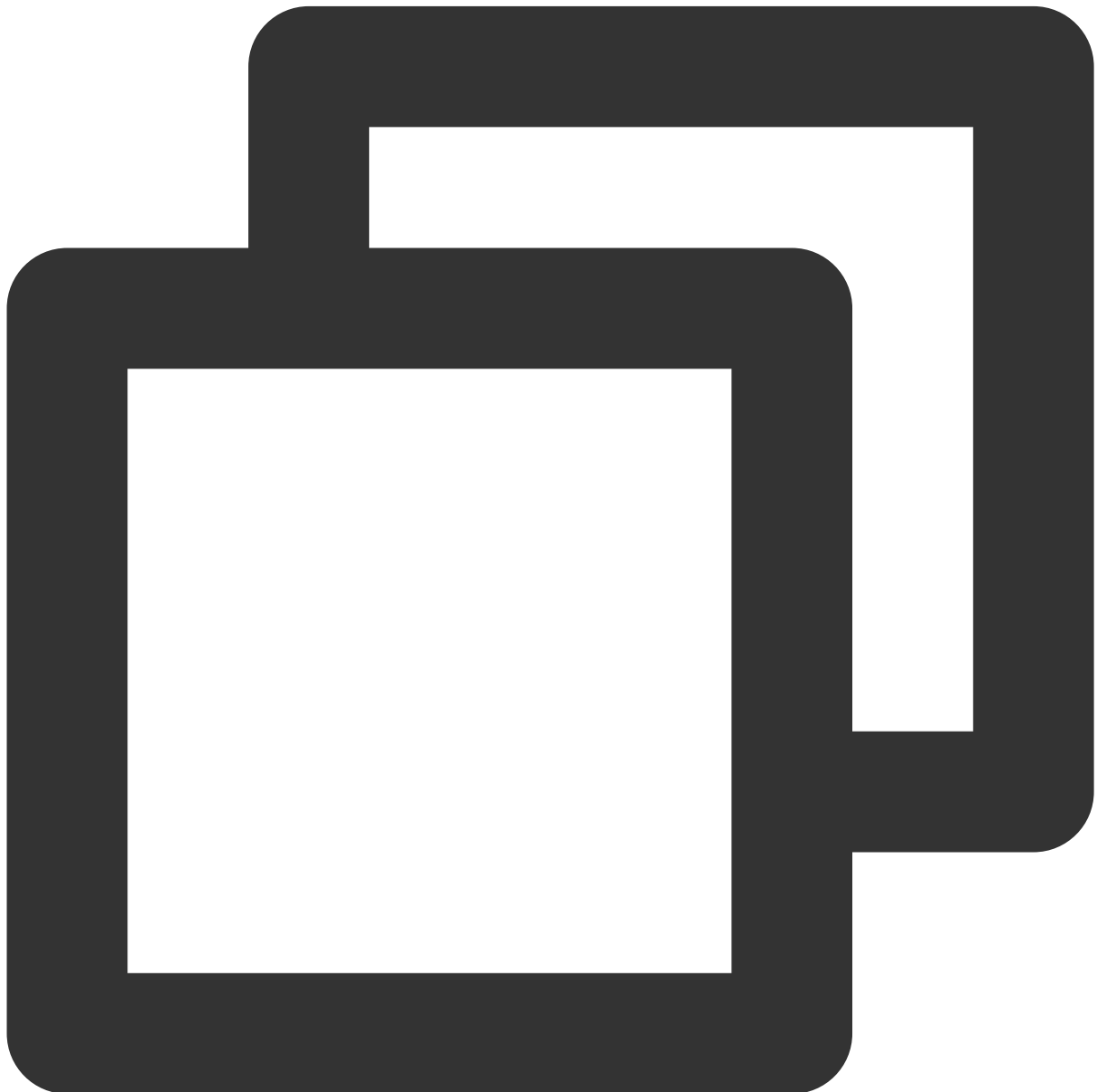
[CloudAudit Event](#)

# Event Structure

Last updated : 2024-07-23 15:08:07

An event is a data record of a status change. This document describes the details of event parameters in EventBridge. Event release from an event source to EventBridge needs to comply with CloudEvents specifications. For more information, please see [CloudEvents - Version 1.0](#).

Below is a sample structure of event release from an event source to EventBridge:



```
{
```



```

"specversion":"0",
"id":"13a3f42d-7258-4ada-da6d-023a333b4662",
"type":"cos:created:object",
"source":"cos.cloud.tencent",
"subject":"qcs::cos:ap-guangzhou:uid1250000000:bucketname",
"time":"1615430559146",
"region":"ap-guangzhou",
"datacontenttype": "application/json;charset=utf-8",
"data":{
  $data_value
}
}

```

The event parameters are as detailed below:

| Field           | Description  | Data Type |
|-----------------|--|-----------|
| specversion     | Event structure version (CloudEvents version, which is 1.0.2 currently).   | String    |
| id              | ID returned by <code>PUT Event</code> .  | String    |
| type            | Type of the event input through <code>PUT Event</code> . The value is <code>COS:Created:PostObject</code> by default for a Tencent Cloud service. Different types are separated with colons.             | String    |
| source          | Event source (which is required for a Tencent Cloud service event and is the abbreviation of <code>subject</code> ). The value is <code>xxx.cloud.tencent</code> by default for a Tencent Cloud service. | String    |
| subject         | Event source details, which can be customized. QCS description such as <code>qcs::dts:ap-guangzhou:appid/uin:xxx</code> is used for a Tencent Cloud service by default.                                  | String    |
| time            | Event time, which is a GMT+0 timestamp in milliseconds such as <code>1615430559146</code> .  | Timestamp |
| datacontenttype | Data structure declaration.  | String    |
| region          | Region information.  | String    |
| data            | Details of the event input through <code>PUT Event</code> .  | String    |

There are two types of events published from event sources to EventBridge:

### Tencent Cloud service event

Tencent Cloud services are automatically connected to EventBridge as event sources.

### **Custom application event**

When connecting your application as an event source, you need to configure the corresponding API/SDK for connection.

# Connector

## Overview

Last updated : 2024-07-23 15:08:07

A connector is mainly used to proactively pull events from event sources and push them to a custom event bus in **standard format**. With it, you don't need to care about the underlying consumer delivery logic. You can bind one or more connectors in the event bus to automatically pull event content from message queues and gateways and push the content to the specified event bus.

EventBridge connectors support the following event sources:

| Event Source             | Configuration Method             |
|--------------------------|----------------------------------|
| API gateway (APIGW) HTTP | <a href="#">APIGW connector</a>  |
| CKafka                   | <a href="#">CKafka connector</a> |
| DTS data subscription    | <a href="#">DTS connector</a>    |

EventBridge provides the following connector management capabilities:

Creating connectors

Viewing connectors

Editing connectors

Starting connectors

Stopping connectors

Deleting connectors

# Configuring APIGW Connector

Last updated : 2024-07-23 15:08:07

## Overview

You can configure API Gateway to process events delivered through webhook so as to use a third-party webhook to receive events generated by other systems. An API Gateway connector is an ideal cross-terminal event receiving connector in HTTP scenarios.

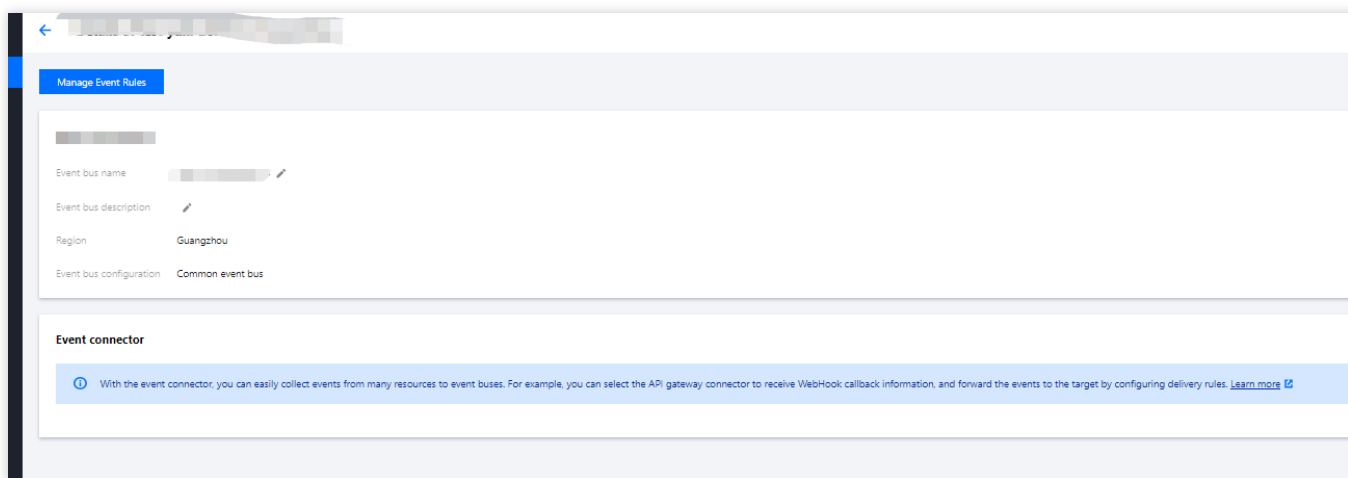
An API Gateway connector is implemented in the **push pattern**. API Gateway monitors requests, generates the corresponding call events, delivers them to the event bus, and routes them to relevant services through event rules.

## Prerequisites

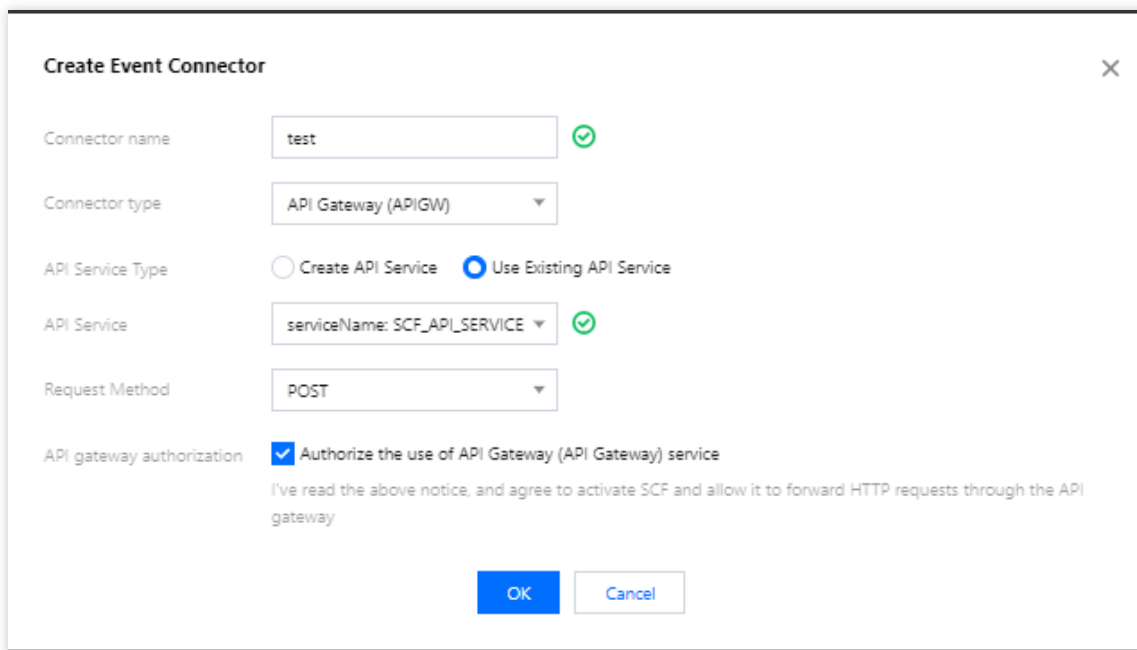
You have [created an event bus](#).

## Directions

1. Log in to the [EventBridge console](#) and select **Event Bus** on the left sidebar.
2. In the **Event Bus** list, select the event bus for which you want to configure an APIGW connector.
3. On the **Event Bus Details** page, click **Add** in the connector configuration section.



4. Enter the relevant information as prompted as shown below:



**Create Event Connector**

Connector name: test ✓

Connector type: API Gateway (APIGW)

API Service Type:  Create API Service  Use Existing API Service

API Service: serviceName: SCF\_API\_SERVICE ✓

Request Method: POST

API gateway authorization:  Authorize the use of API Gateway (API Gateway) service  
I've read the above notice, and agree to activate SCF and allow it to forward HTTP requests through the API gateway

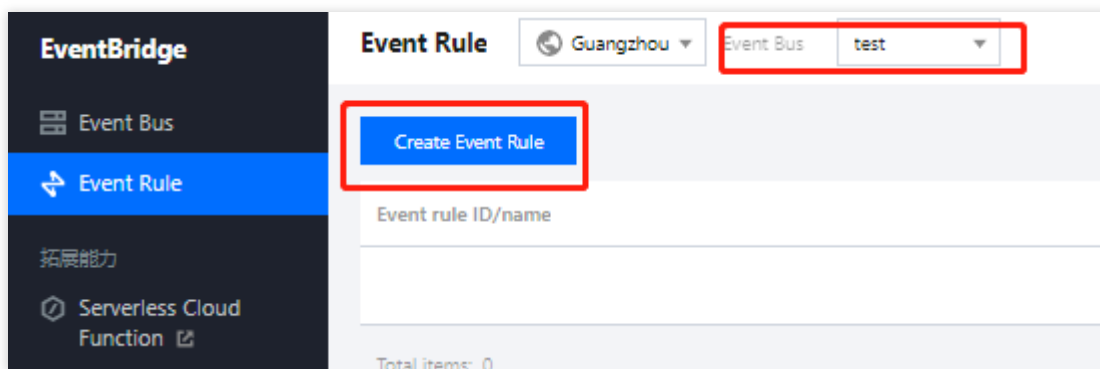
OK Cancel

Here, select **API Gateway (APIGW)** for **Connector type**.

5. Click **OK**.

6. Select **Event Rule** on the left sidebar.

7. In the drop-down lists at the top of the **Event Rule** page, select the same connector information as that set during connector creation and click **Create Event Rule** as shown below:



**EventBridge**

Event Bus

Event Rule

拓展能力

Serverless Cloud Function

**Event Rule** Guangzhou Event Bus test

Create Event Rule

Event rule ID/name

Total items: 0

8. Enter the relevant information as prompted as shown below:

### ← Create Event Rule

1 Rule Pattern > 2 Delivery Target

#### Basic Information

Region: Guangzhou  
Event Bus: eb-my0aecoe(default)  
Rule Name \*  ✓  
Rule Description:

#### Event Matching

Rule Pattern:   
Tencent Cloud service:   
Event Type \*   
Rule Pattern Preview \* 

```
1 {  
2   "source": "apigw.cloud.tencent"  
3 }  
4
```

[Edit](#) [Examples](#)

▶ Test Event Matching

**Rule Pattern** > **2 Delivery Target**

### Delivery Target

Trigger \*

Function source \*  Existing function  New function

Namespace \*  [Create Namespace](#)

Function resource \*  [Learn More](#)

Version and alias \*

Batch delivery  Enable

Add

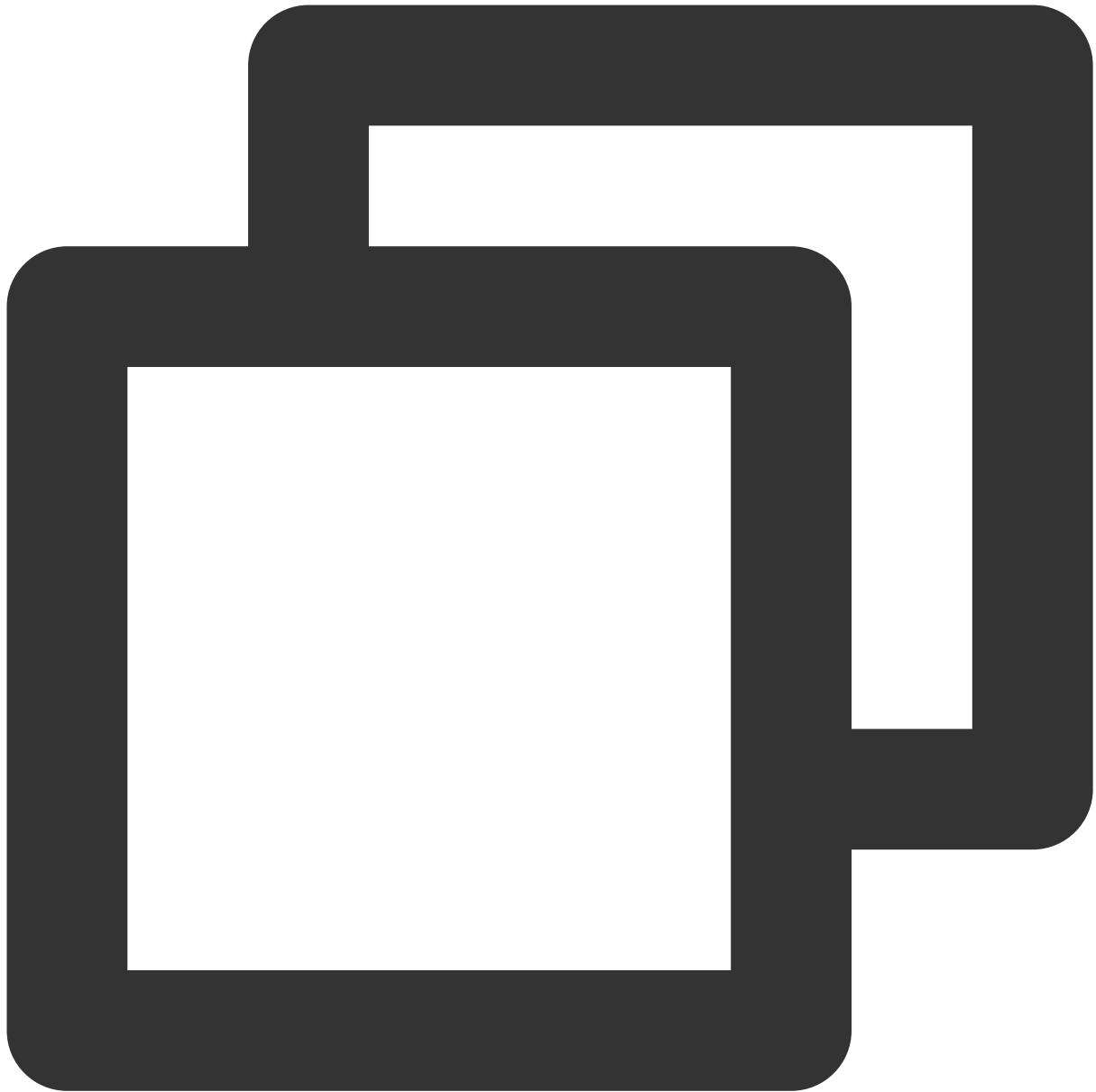
Enable event rules now

[Previous](#) [Complete](#)

9. Select **API Gateway (APIGW)** for **Tencent Cloud service** and configure the delivery target.

10. Click **OK**.

### API Gateway connector data structure description



```
{
  "specversion": "1.0",
  "id": "13a3f42d-7258-4ada-da6d-023a333b4662",
  "type": "connector:apigw",
  "source": "apigw.cloud.tencent",
  "subject": "qcs::apigw:ap-guangzhou:uid1250000000/appidxxx:Serverid/Appid",
  "time": "1615430559146",
  "region": "ap-guangzhou",
  "datacontenttype": "application/json;charset=utf-8",
  "data": {
    "headers": {
```



```
    "Accept-Language": "en-US,en,cn",
    "Accept": "text/html,application/xml,application/json",
    "Host": "service-3ei3tii4-251000691.ap-guangzhou.apigateway.myqcloud.com",
    "User-Agent": "User Agent String"
  },
  "body": "{\\"test\\":\\"body\\"}",
  "stageVariables": {
    "stage": "release"
  },
  "path": "/test/value",
  "queryString": {
    "foo": "bar",
    "bob": "alice"
  },
  "httpMethod": "POST"
}
}
```

The parameters are described as follows:

| Parameter   | Description  |
|-------------|--|
| path        | Records the complete path of the actual request.   |
| httpMethod  | Records the HTTP method of the actual request.   |
| queryString | Records the complete query content of the actual request.                                    |
| body        | Records the content of the actual request after being converted into a <code>String</code> . |
| headers     | Records the complete headers of the actual request.  |

# Configuring CKafka Connector

Last updated : 2024-07-23 15:08:07

## Overview

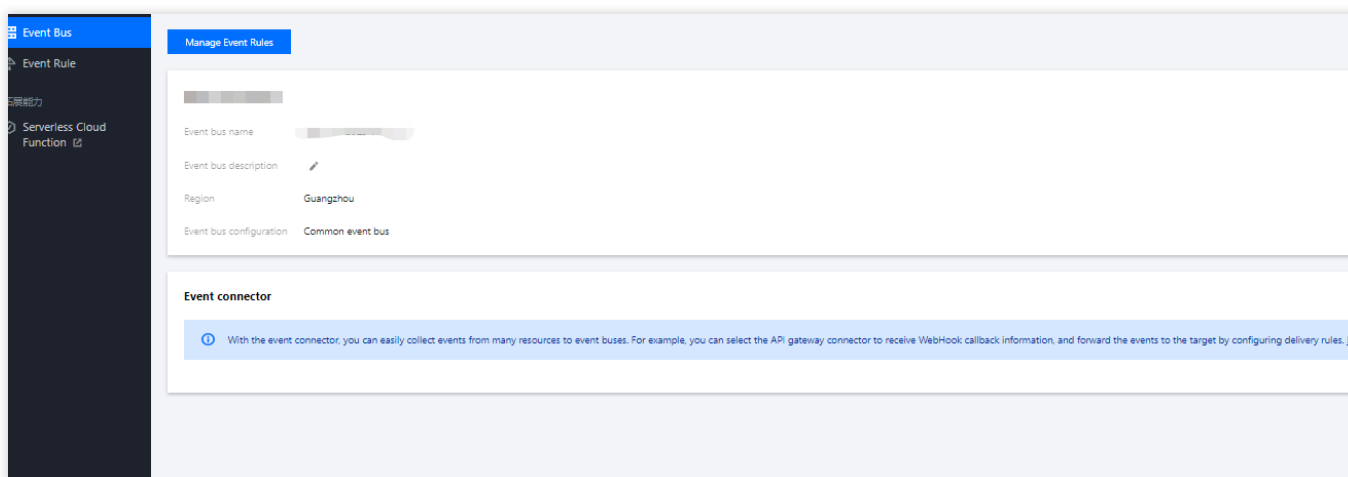
You can configure a CKafka connector to consume content in CKafka message queues. A CKafka connector is implemented in the **pull pattern**. It automatically pulls CKafka content and routes events to relevant services through event rules. This document describes how to create a CKafka connector and the structure of events generated by the CKafka connector.

## Prerequisites

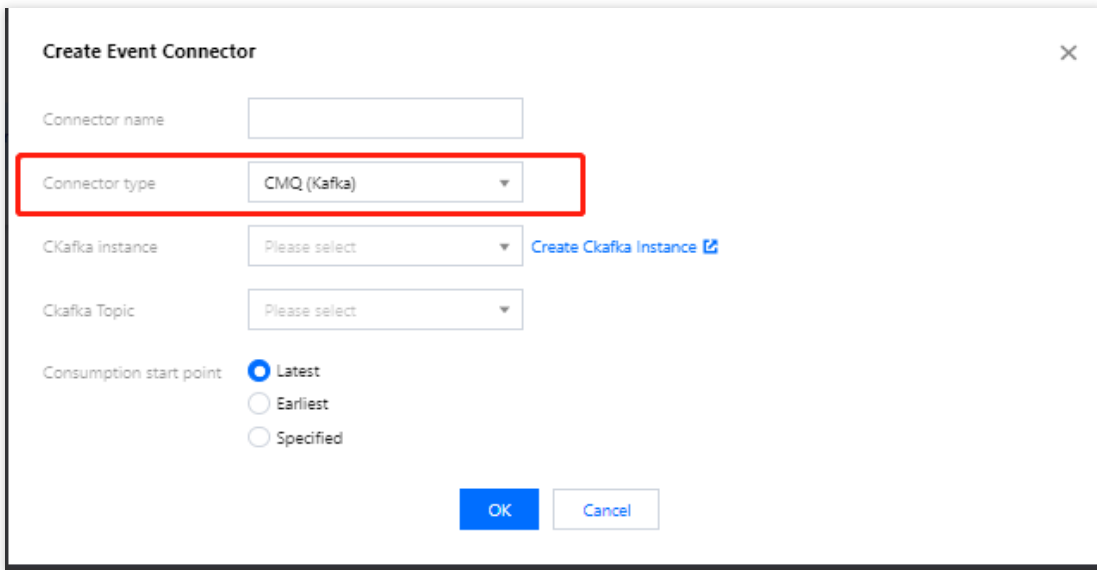
You have [created an event bus](#).

## Directions

1. Log in to the [EventBridge console](#) and select **Event Bus** on the left sidebar.
2. In the **Event Bus** list, select the event bus for which you want to configure a CKafka connector.
3. On the **Event Bus Details** page, click **Add** in the connector configuration section.



4. Enter the relevant information as prompted as shown below:



**Create Event Connector**

Connector name

Connector type **CMQ (Kafka)**

CKafka instance  [Create Ckafka Instance](#)

CKafka Topic

Consumption start point  Latest  
 Earliest  
 Specified

**OK**

Here, select **CMQ (Kafka)** for **Connector type** and enter other configuration items as prompted.

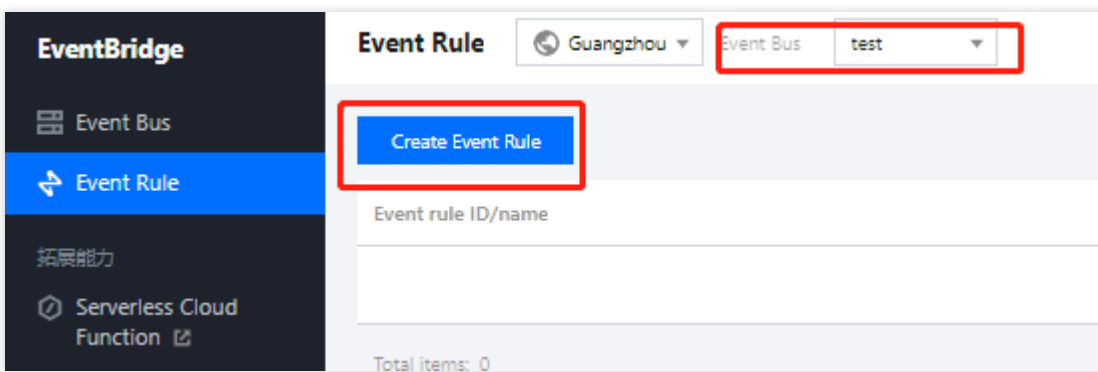
**Note:**

Currently, only delivery for Tencent Cloud CKafka instances is supported. Please confirm that no username and password have been configured for your CKafka instances; otherwise, the connector may fail to get messages.

5. Click **OK**.

6. Select **Event Rule** on the left sidebar.

7. In the drop-down lists at the top of the **Event Rule** page, select the same connector information as that set during connector creation and click **Create Event Rule** as shown below:



**EventBridge**

Event Bus

**Event Rule**

拓展能力

Serverless Cloud Function

**Event Rule**

**Create Event Rule**

Event rule ID/name

Total items: 0

8. Enter the relevant information as prompted as shown below:

The screenshot displays the 'Rule Pattern' configuration page in the Tencent Cloud Event Bridge console. The page is divided into two main sections: 'Basic Information' and 'Event Matching'. In the 'Basic Information' section, the 'Region' is set to 'Guangzhou' and the 'Event Bus' is 'eb-my0aecoe(default)'. There are input fields for 'Rule Name' and 'Rule Description'. The 'Event Matching' section includes dropdown menus for 'Rule Pattern' (set to 'Default'), 'Tencent Cloud service' (set to 'Message Queue CKafka'), and 'Event Type' (set to 'All events'). A 'Rule Pattern Preview' window shows a JSON snippet: 

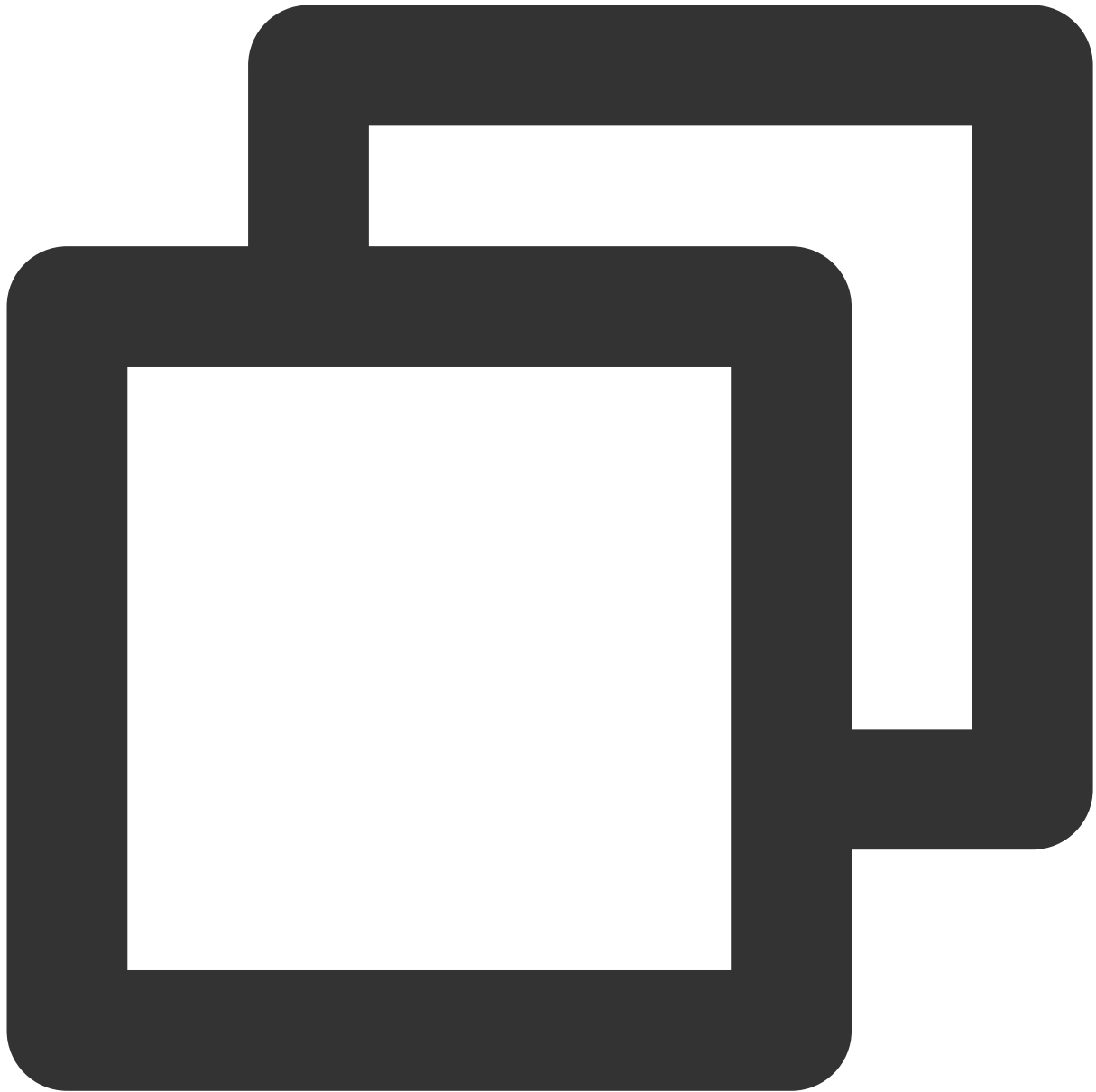
```
1 {
2   "source": "kafka.cloud.tencent"
3 }
4
```

 with 'Edit' and 'Examples' buttons. A 'Test Event Matching' link is also present. At the bottom, there are 'Previous' and 'Next' navigation buttons.

Here, select **Message Queue CKafka** for **Tencent Cloud service** and configure the delivery target.

9. Click **OK**.

### Structure of events generated by the CKafka connector



```
{
  "specversion": "1.0",
  "id": "13a3f42d-7258-4ada-da6d-023a333b4662",
  "type": "connector:kafka",
  "source": "ckafka.cloud.tencent",
  "subject": "qcs::ckafka:ap-guangzhou:uin/1250000000:ckafkaId/uin/1250000000",
  "time": "1615430559146",
  "region": "ap-guangzhou",
  "datacontenttype": "application/json;charset=utf-8",
  "data": {
    "topic": "test-topic",
```

```
    "Partition":1,  
    "offset":37,  
    "msgKey":"test",  
    "msgBody":"Hello from Ckafka again!"  
  }  
}
```

The parameters are described as follows:

| Parameter | Description  |
|-----------|--|
| topic     | CKafka delivery topic.   |
| Partition | Event source partition. One topic can contain one or more partitions. CKafka uses partition as an allocation unit. |
| offset    | Consumer group, which specifies the consumption start point.   |
| msgKey    | CKafka message key.  |
| msgBody   | CKafka message body.   |

# Configuring a DTS Connector

Last updated : 2024-07-23 15:08:07

## Overview

[Data Transmission Service \(DTS\)](#) supports data subscription for MySQL, MariaDB, PostgreSQL, Redis, MongoDB, and other relational databases and NoSQL databases. It gets the data changes of key database businesses and provides them for downstream entities to subscribe to, get, and consume. This facilitates data sync between TencentDB databases and heterogeneous systems, such as cache update, real-time ETL (data warehousing technology) sync, and async business decoupling.

In EventBridge, you can configure a DTS connector to pull incremental BinLog from the source instance in real time based on DTS data subscription to consume and process logs and deliver them to different targets. This document introduces how to create a DTS connector and the structure of events generated by the DTS connector.

## Feature Description

For more information, see [Data Subscription \(for Kafka\) Overview](#).

### Supported databases

| Source Database Type            | Source Database Version                            | Subscribable Data Types                          |
|---------------------------------|--|--|
| TencentDB for MySQL             | TencentDB for MySQL 5.5.x, 5.6.x, 5.7.x, and 8.0.x | Data update<br>Structure update<br>Full instance |
| TencentDB for MariaDB           | TencentDB for MariaDB 10.0.10 and 10.1.9           | Data update<br>Structure update<br>Full instance |
| TencentDB for MariaDB (Percona) | TencentDB for MariaDB (Percona) 5.6.x and 5.7.x    | Data update<br>Structure update<br>Full instance |
| TDSQL for MySQL                 | TDSQL for MySQL 5.6.x, 5.7.x, and 8.0.18           | Data update<br>Structure update<br>Full instance |
| TDSQL-C MySQL                   | TDSQL-C MySQL 5.7.x and 8.0.x                      | Data update                                      |

|                      |                      |                                   |
|----------------------|----------------------|-----------------------------------|
|                      |                      | Structure update<br>Full instance |
| TDSQL for PostgreSQL | TDSQL for PostgreSQL | Data update                       |

## Supported subscription operations

DTS allows you to subscribe to databases and tables. Specifically, the following three subscription types are supported:

Data update: Subscription to DML operations.

Structure update: Subscription to DDL operations.

Full instance: Subscription to the DML and DDL operations of all tables.

## Use limits

EventBridge has a limit on the event size. An upstream log greater than **1 MB** cannot be successfully delivered to EventBridge for consumption. Pay attention to your log size.

Under the current scheme, EventBridge can consume data in multiple partitions concurrently and cannot guarantee the consumption sequence.

To ensure normal data consumption, you need to exercise caution when updating the consumer group account password after creating a connector. If the password is updated, you need to bind the connector again. Otherwise, data consumption may fail.

DTS supports batch delivery. Batch changes will be delivered in array format in the same event.

## Prerequisites

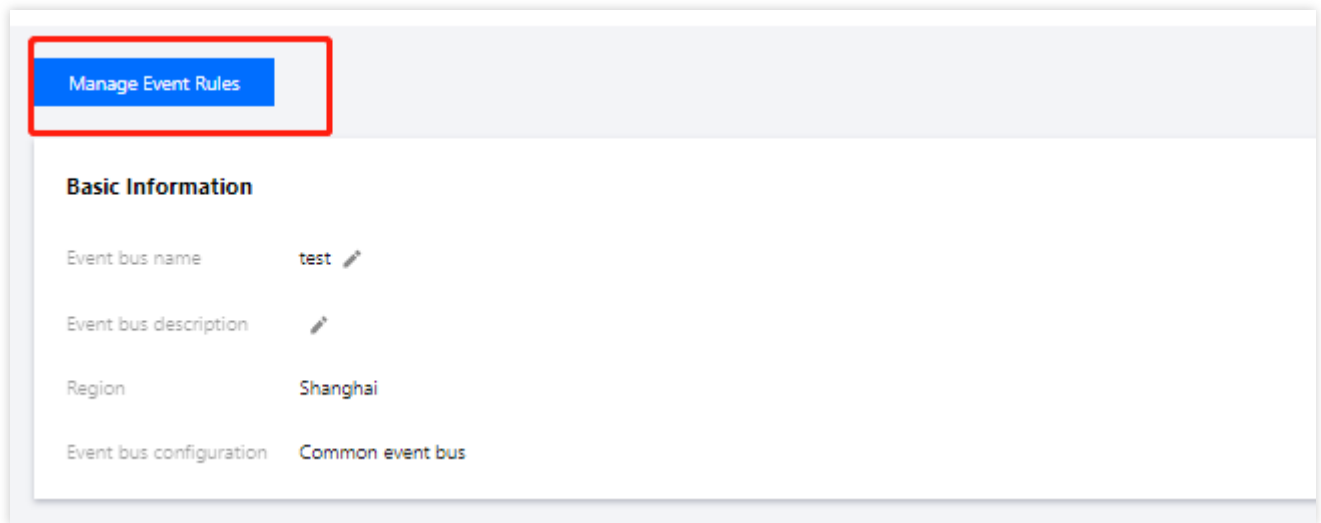
1. You have [activated the DTS data subscription service](#) and created an instance.
2. Your **sub-account** has obtained EventBridge and DTS related operation permissions by using the root account.

## Directions

1. Log in to the [EventBridge console](#) and select **Event Bus** on the left sidebar.
2. In the **Event Bus** list, select the event bus for which you want to configure a DTS connector.
3. On the event bus details page, click **Add** in the event connector section, and set the connector type to **DTS data subscriptions**.
4. Select the data subscription instance to consume as instructed, and enter information such as the consumer group name, account, and password as needed. If no consumer object is bound, complete related configuration in the DTS console.
5. Click **OK**. Then you can view the information of the bound connector on the event bus page.



6. On the event bus details page, click **Manage Event Rules**.



7. Click **Create Event Rule** and enter the relevant information as prompted.

You need to set **Tencent Cloud service** to **DTS data subscriptions** and configure [data conversion](#) and complete trigger target binding as needed.

8. Click **OK**.

## Event Format

The format of processed events received by a DTS connector is as follows:

### DDL operation example



```
{
  "id": "38cecd93-a9c2-11ec-b952-*****d8da53:16",
  "type": "dts:MYSQL:INSERT",
  "specversion": "1.0",
  "source": "dts.cloud.tencent",
  "subject": "cdb-xxxx",
  "time": 1648109734,
  "region": "ap-guangzhou",
  "datacontenttype": "application/json;charset=utf-8",
  "tags": null,
  "data": {
```

```
"topic": "topic-sub-xxxx-cdb-xxxx",
"partition": 3,
"offset": 16005,
"partition_seq": 16006,
"event": {
  "dmlEvent": {
    "columns": [
      {
        "name": "string",
        "originalType": "text"
      },
      {
        "name": "int",
        "originalType": "tinyint (4) "
      },
      {
        "name": "time",
        "originalType": "time"
      },
      {
        "name": "double",
        "originalType": "double"
      },
      {
        "name": "id",
        "originalType": "int (11) ",
        "isKey": true
      },
      {
        "name": "float",
        "originalType": "float"
      },
      {
        "name": "longtext",
        "originalType": "longtext"
      }
    ],
    "rows": [
      {
        "newColumns": [
          {
            "dataType": 13,
            "charset": "utf8",
            "bv": "dG1w"
          },
          {
```

```
        },
        {
            },
            {
                "dataType":10,
                "sv":"1"
            },
            {
                "dataType":3,
                "sv":"3"
            },
            {
            },
            {
            }
        ]
    }
]
}
},
"header":{
    "sourceType":1,
    "messageType":2,
    "timestamp":1648109734,
    "serverId":109741,
    "fileName":"mysql-bin.000005",
    "position":2234587,
    "gtid":"38cecd93-a9c2-11ec-b952-*****d8da53:16",
    "schemaName":"dts",
    "tableName":"dts_mysql",
    "seqId":16017,
    "isLast":true
},
"eb_consumer_time":"2022-03-24 16:15:34.287359965 +0800 CST m=+1120.357657669
"eb_connector":""
}
}
```

## DML operation example



```
{
  "id": "38cecd93-a9c2-11ec-b952-*****8da53:19",
  "type": "dts:MYSQL:DDL",
  "specversion": "1.0",
  "source": "dts.cloud.tencent",
  "subject": "cdb-xxxx",
  "time": 1648110060,
  "region": "ap-guangzhou",
  "datacontenttype": "application/json;charset=utf-8",
  "tags": null,
  "data": {
```

```

"topic": "topic-sub-aniwxeeem4-cdb-xxxx",
"partition": 0,
"offset": 16065,
"partition_seq": 16066,
"event": {
  "ddlEvent": {
    "schemaName": "dts",
    "sql": "ALTER TABLE `dts_mysql` ADD COLUMN `t` tinyint (0) NULL , ADD UN
  }
},
"header": {
  "sourceType": 1,
  "messageType": 3,
  "timestamp": 1648110060,
  "serverId": 109741,
  "fileName": "mysql-bin.000005",
  "position": 2235430,
  "gtid": "38cecd93-a9c2-11ec-b952-*****d8da53:19",
  "seqId": 16087,
  "isLast": true
},
"eb_consumer_time": "2022-03-24 16:21:01.19682088 +0800 CST m=+1447.267118604"
"eb_connector": ""
}
}

```

## Parameter description

| Parameter   | Description  |
|-------------|--|
| id          | Event ID, which is automatically generated by EventBridge. It is the unique ID of an event in EventBridge.   |
| type        | Event type in three-segment format. For a DTS connector, the format is <code>dts:\${Database type}:\${Operation type}</code> .                               |
| specversion | CloudEvents version, which is 1.0 by default. The value of this parameter is automatically generated by EventBridge.   |
| source      | Event source. For a DTS connector, the value is <code>dts.cloud.tencent</code> .   |
| subject     | Instance on which an event is generated. For a DTS connector, the value of this parameter is the ID of the database instance bound to the data subscription. |
| time        | Time when an event is delivered to EventBridge.  |
| region      | Region where an event is generated.  |

|                |   |
|----------------|---|
| tags           | Resource tags.  |
| data           | Binlog content of the database.   |
| data.topic     | Information of the data subscription instance.  |
| data.partition | Consumer partition.   |
| data.offset    | Consumer offset.  |
| data.event     | There are two types: <code>dmlEvent</code> and <code>ddlEvent</code> . <code>dmlEvent</code> describes the data table schema format and updates. <code>ddlEvent</code> describes specific SQL operations. |
| data.header    | Operation log header information, including the database name, table name, and update timestamp.  |

# Configuring TDMQ Connector

Last updated : 2024-07-23 15:08:07

## Overview

You can configure a TDMQ connector to consume content in TDMQ message queues. A TDMQ connector is implemented in the **pull pattern**. It automatically pulls TDMQ content and routes events to relevant services through event rules. This document describes how to create a TDMQ connector and the structure of events generated by the TDMQ connector.

### Note

Currently, a TDMQ connector can consume only content in TDMQ-Pulsar message queues.

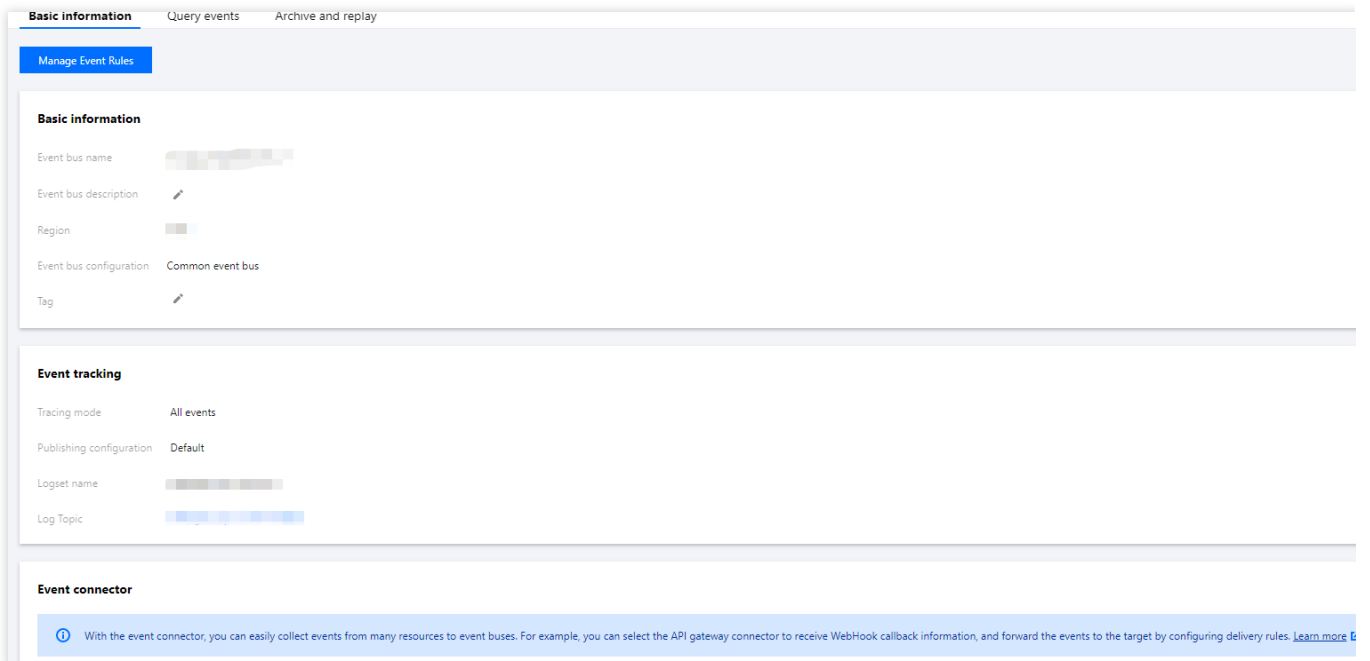
## Prerequisites

You have [created an event bus](#).

## Directions

1. Log in to the [EventBridge console](#) and click **Event Bus** in the left sidebar.
2. In the **Event Bus** list, select the event bus for which you want to configure a TDMQ connector.
3. On the **Event bus details** page, click **Add** in the **Event connector** section.





4. In the **Create event connector window, set parameters as prompted.**

**Here, select Message queue (TDMQ) for Connector type\*\* and set other parameters as prompted.**

5. Click **OK**.

6. Click **Event rule** in the left sidebar.

7. In the drop-down lists at the top of the **Event rule** page, select the same connector information as that set during connector creation and click **Create event rule**, as shown below:



8. Set parameters as prompted, as shown below:

**Create Event Rule**

1 Rule Pattern > 2 Delivery Target

### Delivery Target

Trigger \* Serverless Cloud Function (SCF) ▾

Function source \*  Existing function  New function

Namespace \* Please select ▾ [Create Namespace](#)

Function resource \* Please select ▾ [Learn More](#)

Version and alias \* Please select ▾

Batch delivery  Enable  
After enabling batch delivery, events will be delivered to the function in array form.

Max Waiting Time ⓘ  seconds

Maximum messages ⓘ  rules

Add

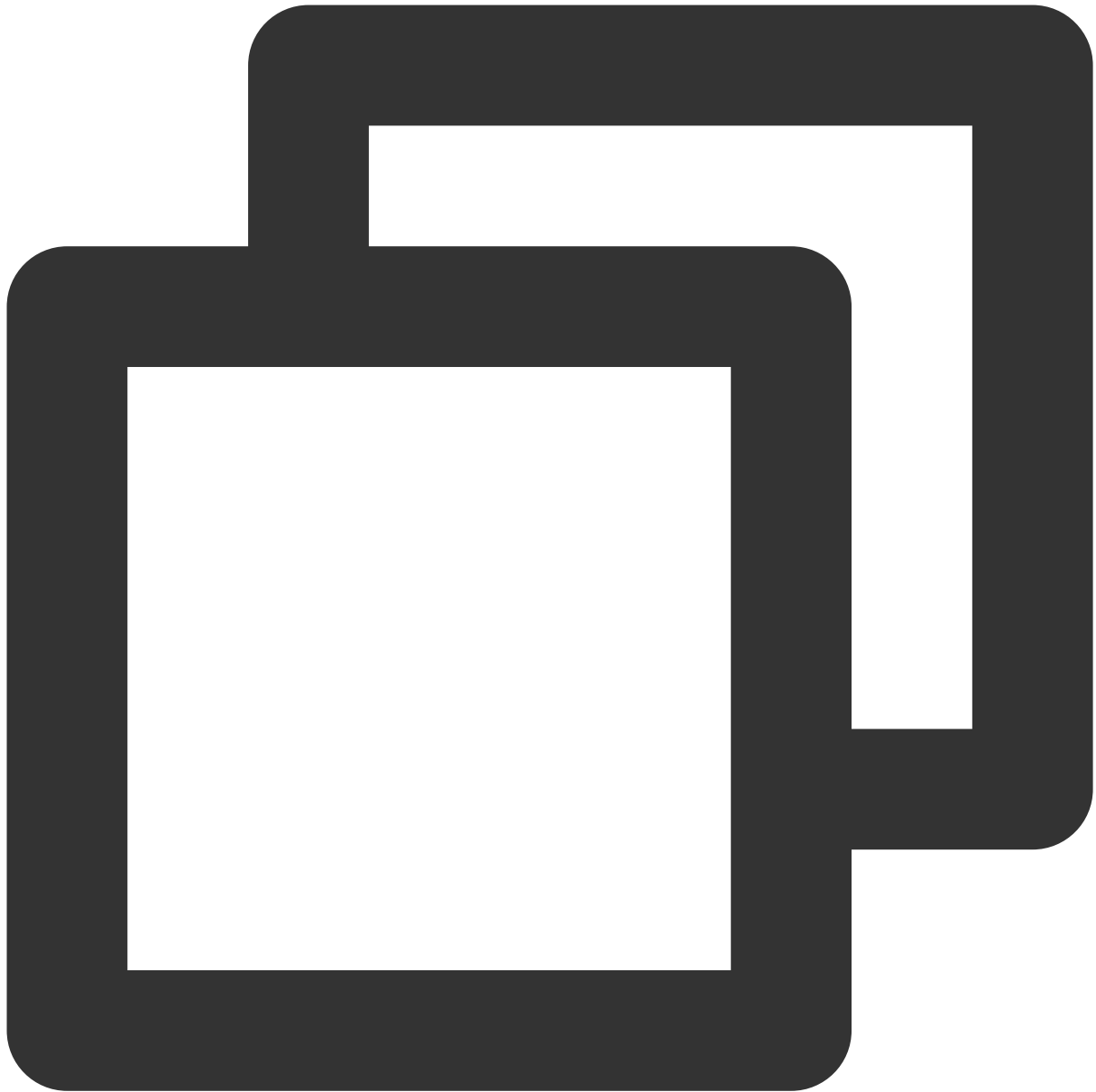
Enable event rules now

[Previous](#) [Complete](#)

Here, select **Message queue (TDMQ)** for **Tencent Cloud service** and configure the event target.

9. Click **OK**.

### TDMQ connector data structure description



```
{
  "specversion": "1.0",
  "id": "13a3f42d-7258-4ada-da6d-023a333b4662",
  "type": "connector:tdmq",
  "source": "tdmq.cloud.tencent",
  "subject": "qcs::tdmq:$region:$account:topicName/$topicSets.clusterId/$topicSet",
  "time": "1615430559146",
  "region": "ap-guangzhou",
  "datacontenttype": "application/json;charset=utf-8",
  "data": {
    "topic": "persistent://appid/namespace/topic-1",
```

```

    "tags": "testtopic",
      "TopicType": "0",
      "subscriptionName": "xxxxxx",
      "toTimestamp": "1603352765001",
    "partitions": "0",
      "msgId": "123345346",
      "msgBody": "Hello from TDMQ!"
  }
}

```

The parameters are described as follows:

| Parameter        | Description   |
|------------------|---|
| topic            | Complete topic path, such as <code>persistent://appid/namespace/topic-1</code> .  |
| tags             | TDMQ tags.  |
| topicType        | Topic type. Valid values:<br>0: general message<br>1: globally sequential message<br>2: partitional sequential message<br>3: retry letter topic<br>4: dead letter topic |
| subscriptionName | Subscription name.  |
| timestamp        | Timestamp, which is accurate to milliseconds.   |
| partitions       | Partitions consumed by a TDMQ message queue.  |
| msgId            | TDMQ message ID.  |
| msgBody          | TDMQ message body.  |