Event Bridge

Event Rule

Product Documentation

Contents

Event Rule Overview

Last updated：2024-07-23 15:08:07

Each event bus has several event rules. Event rules are the core feature of EventBridge and implement capabilities such as event triggering, filtering, and extraction. You can specify several event targets for each event rule. When an event hits an event rule, the event will be pushed to the specified event target. Event rule related features and operations are as follows:

The filter feature in an event rule is provided by the event pattern. For more information, please see Event Pattern.

The target triggering feature in an event rule is provided by the event target.

You can view, modify, or delete an event rule. For more information, see Managing Event Rule.

Event Pattern

Last updated：2024-07-23 15:08:07

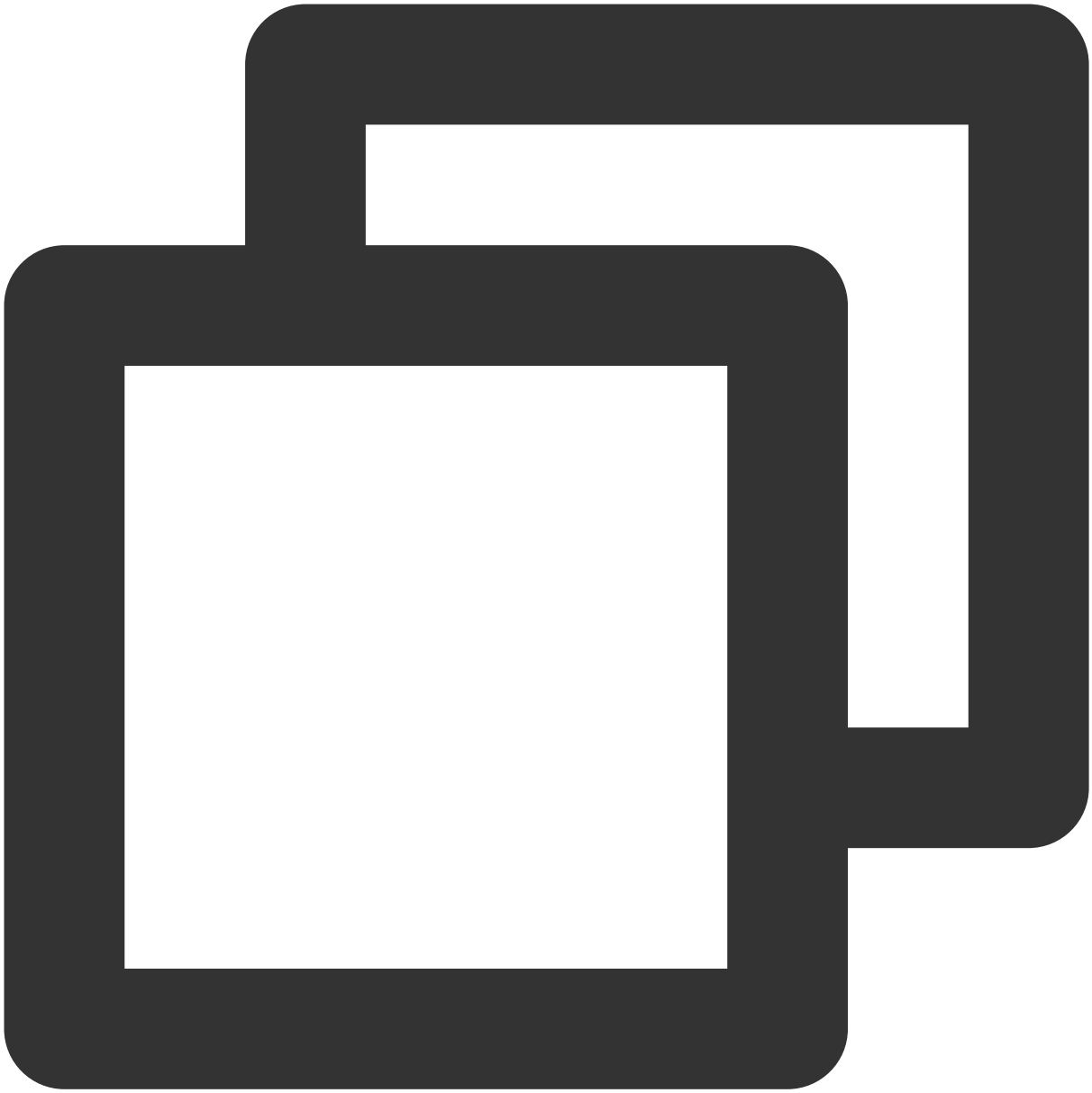
An event pattern defines how EventBridge filters events and routes them to the event target. It must be in the same structure as the matched events. This document describes the common event pattern types:

Must-Knows

The event pattern match rules are as detailed below:
Matched events must contain all field names listed in the event pattern in the same nesting structure.
Match between events and an event pattern is exact match down to the character and case-sensitive. During match, no standardized operations will be performed on the strings.
Values to be matched must be in JSON format, which include strings and numeric values enclosed in quotation marks as well as keywords not enclosed in quotation marks (`true` , `false` , and `null`).

Specifying Condition and Operator

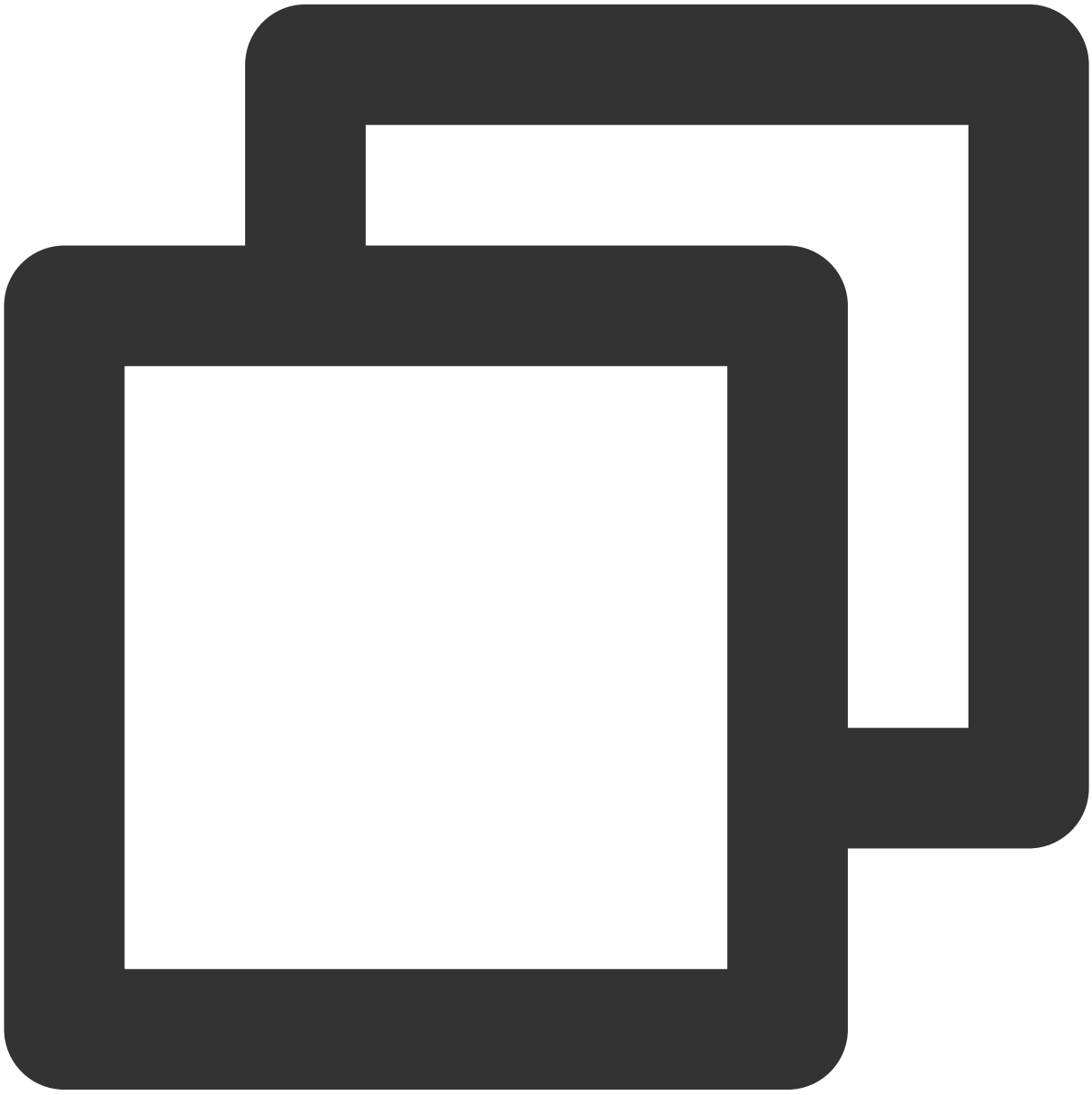
You can specify a field value as a match condition. Values to be matched are in a JSON array and enclosed in `[ ]` . Values in `[ ]` are in OR relationship, and keys are in AND relationship.
Assumes that the following COS event is received:



```
{
    "specversion": "1.0",
    "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",
    "resource": [
        "qcs::eb:ap-guangzhou:uid1250000000:eventbusid/eventruleid"
```

```
    ],
    "data": {
        "name": "testname",
        "scope": 100
    }
}
```

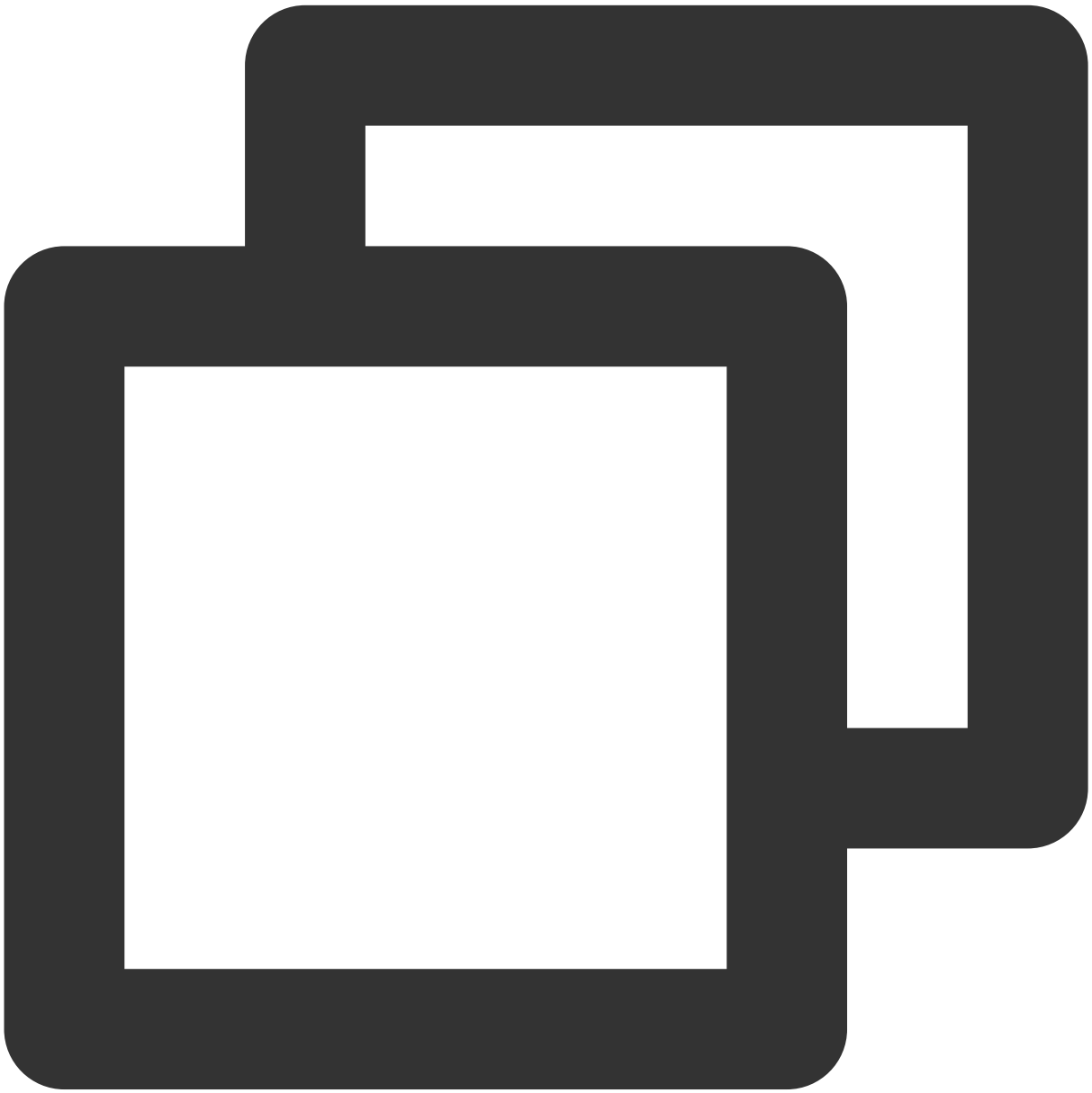
To match this event with the specified `name` in the `data` field, the rule should be:



```
{
    "data": {
```

```
            "name": [
                "testname"
            ]
        }
}
```

To match the event with any of the specified `name` values in the `data` field, the statement should be:



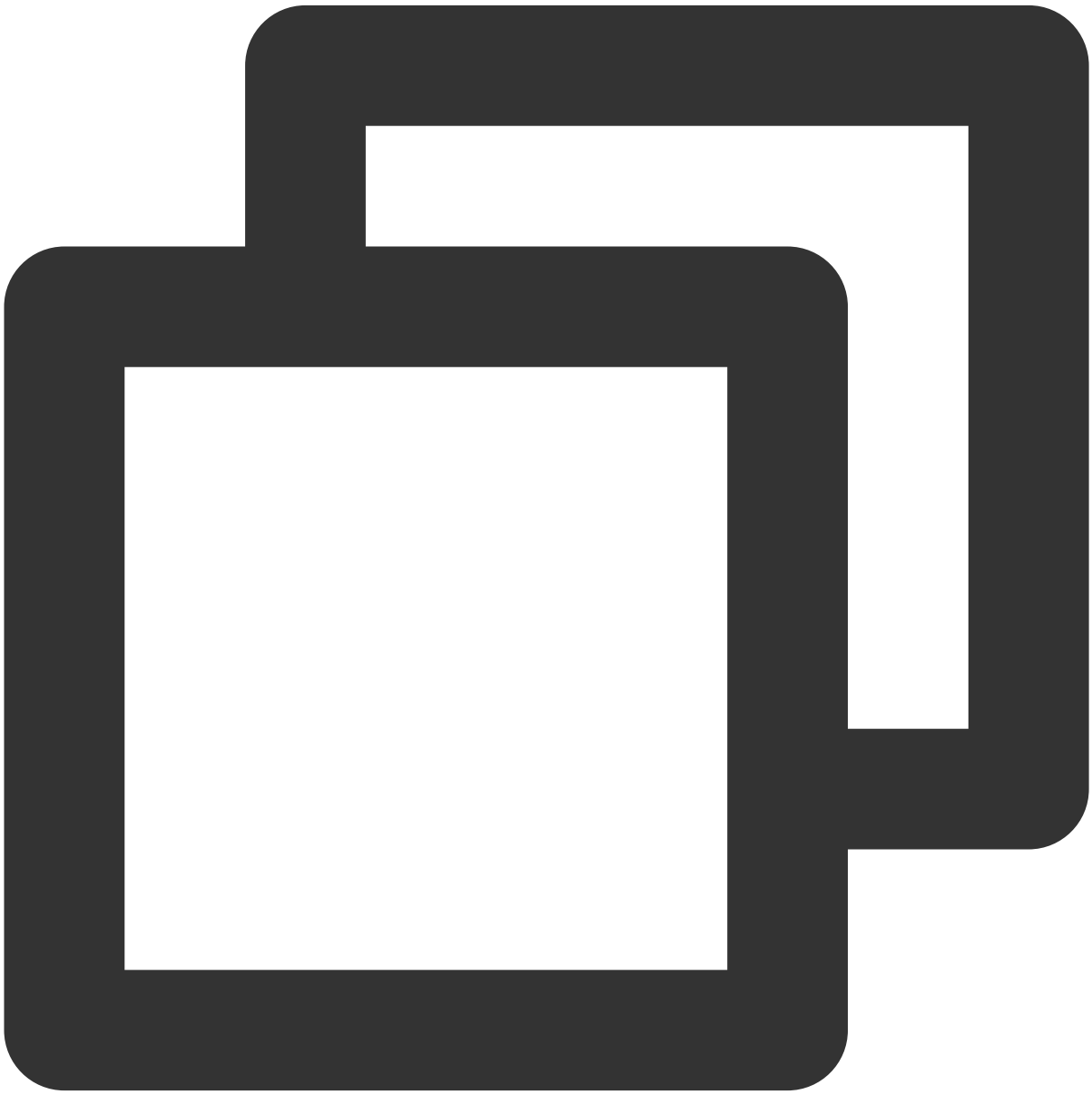
```
{
    "data": {
        "name": [
```

```
            "testname","test"
        ]
    }
}
```

Prefix Matching

You can match events with the specified prefix by using `{ "prefix": "2021-10-02" }`.

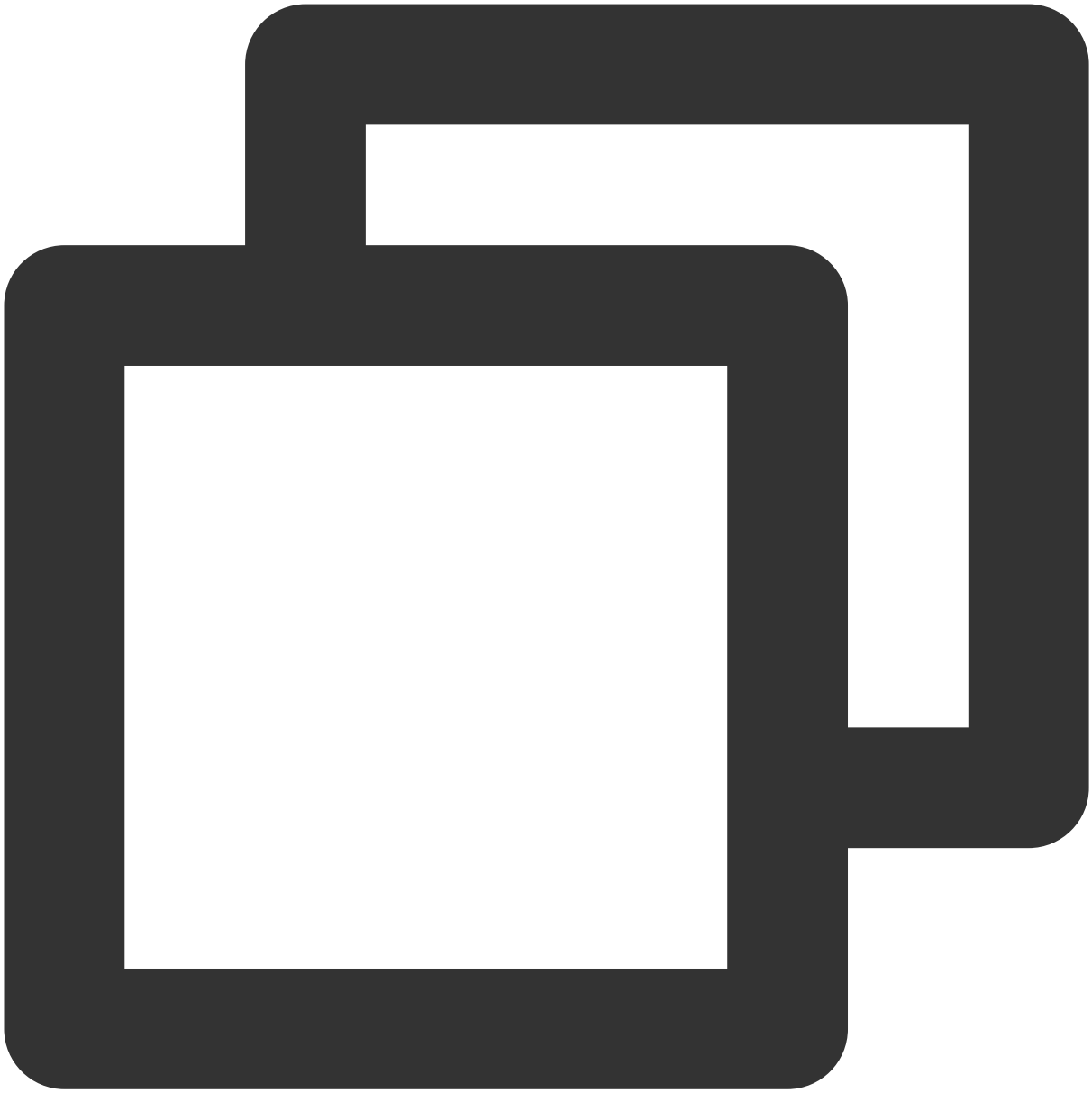
Assumes that the following COS event is received:



```
{
```

```
    "specversion": "1.0",
    "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",
    "resource": [
        "qcs::eb:ap-guangzhou:uid1250000000:eventbusid/eventruleid"
    ],
    "data": {
        "name": "testname",
        "scope": 100
    }
}
```

To match the event with the specified prefix of `name` , the statement should be:

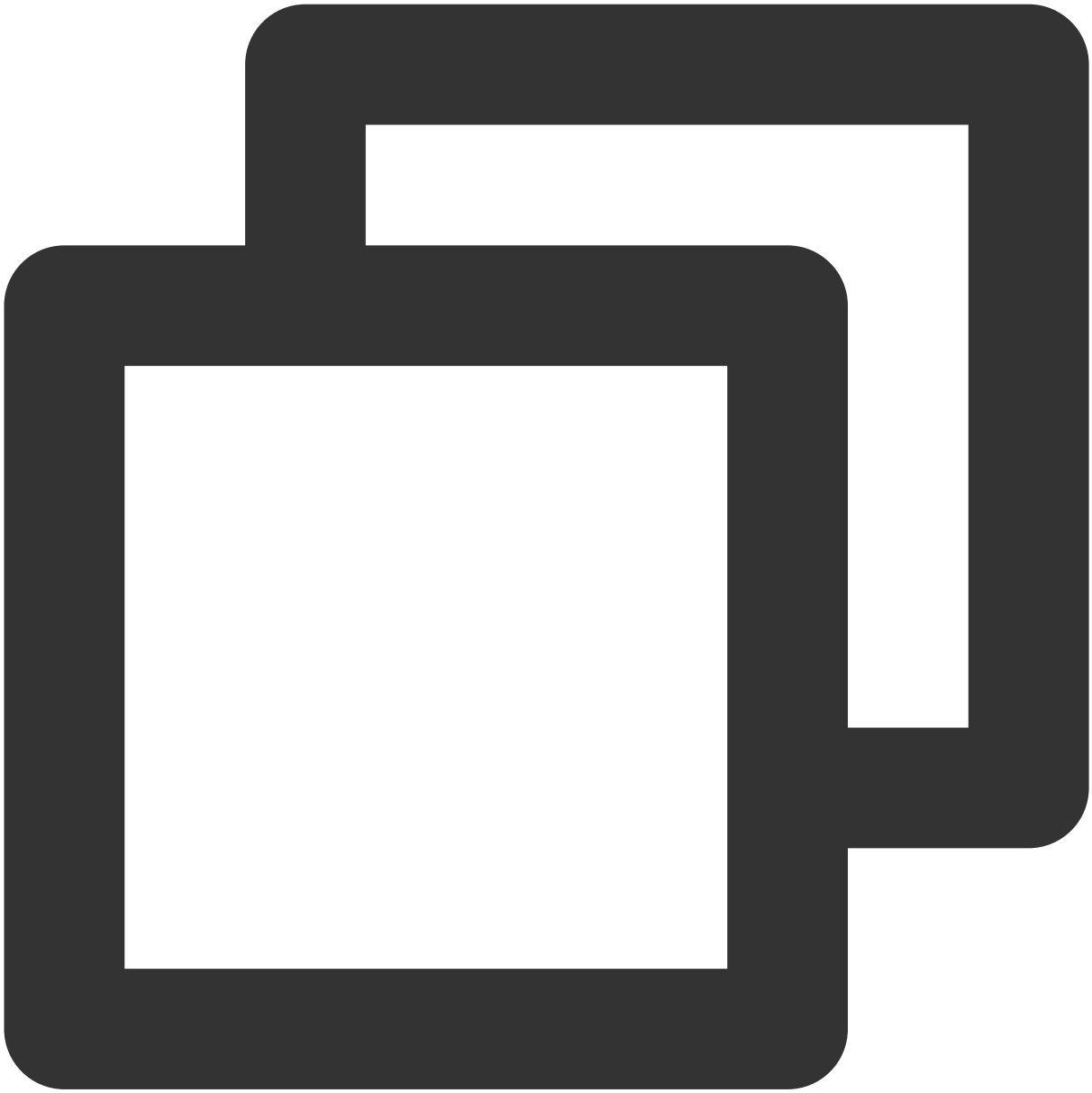


```
{
    "data":{
        "name":[
            {
                "prefix":"te"
            }
        ]
    }
}
```

Suffix Matching

You can match events with the specified suffix by using `{ "suffix": ".txt" }`.

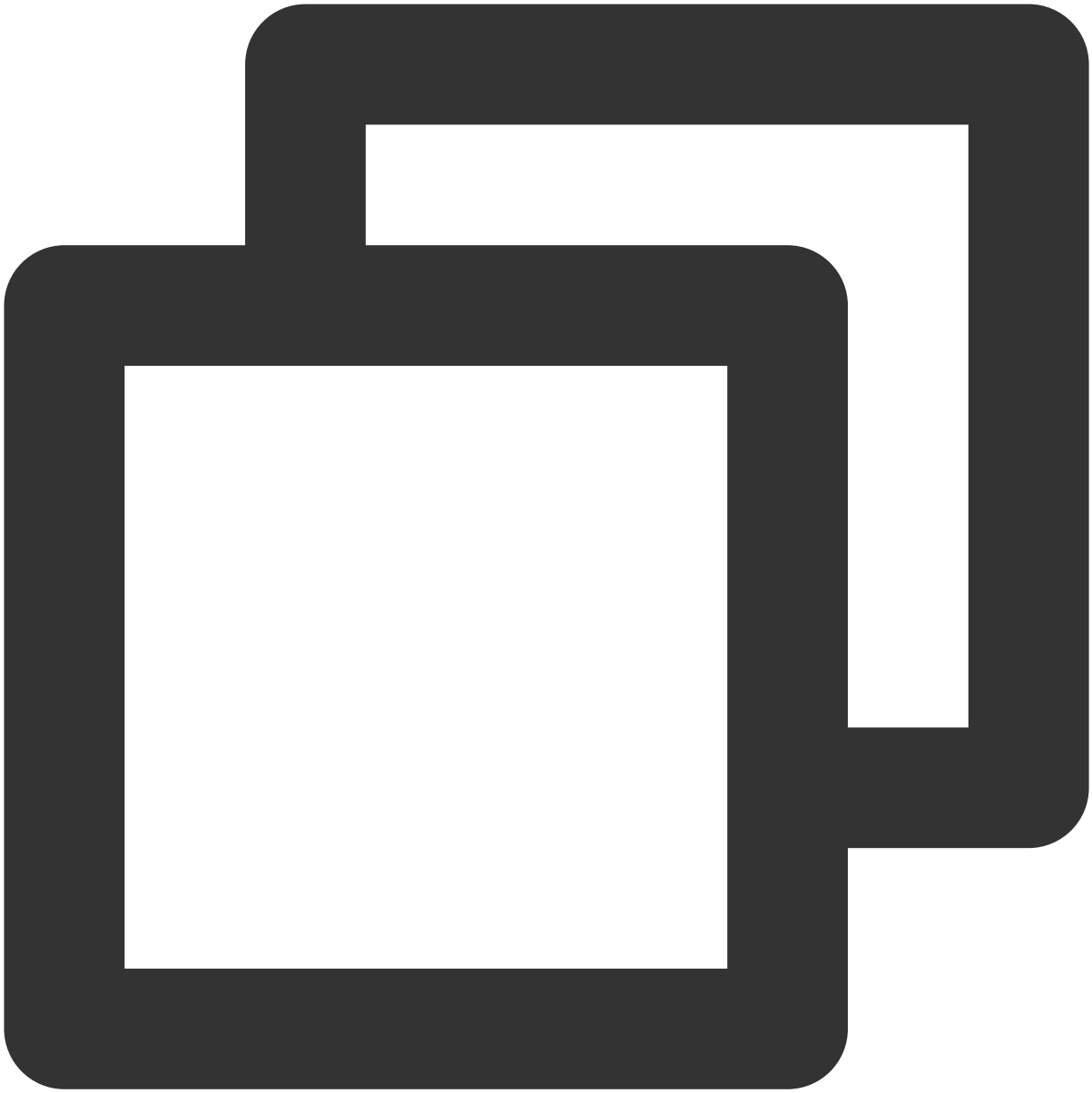
Assumes that the following TDMQ event is received:



```
{
    "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!"
    }
}
```

To match the event with the specified topic suffix, the statement should be:

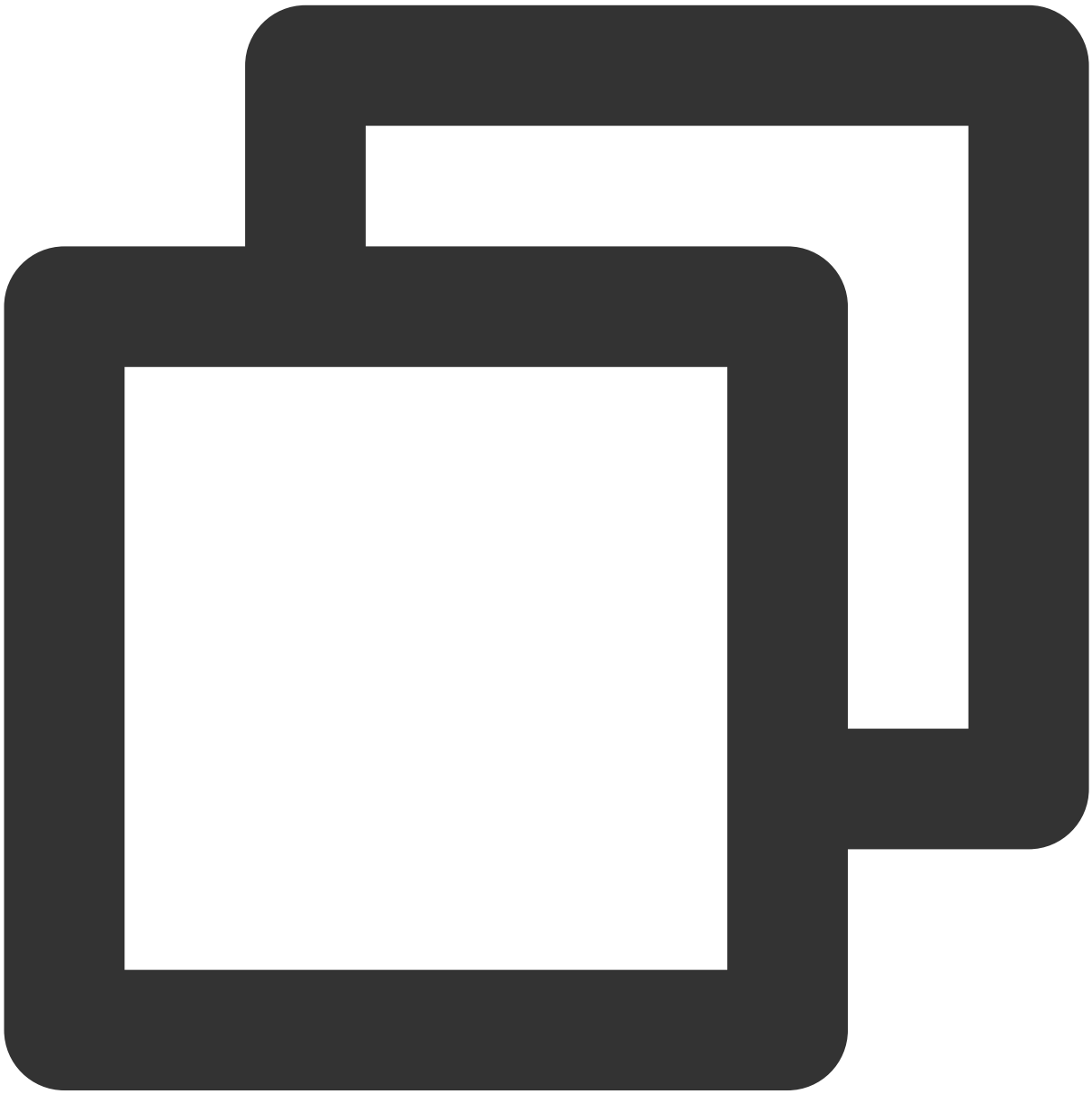


```
{
    "data": {
        "topic": [{
            "suffix":"/topic-1"
        }]
    }
}
```

Exclusion Matching

You can match events that contain anything but the specified value in the specified field with a statement like `{ "anything-but": "initializing" }`.

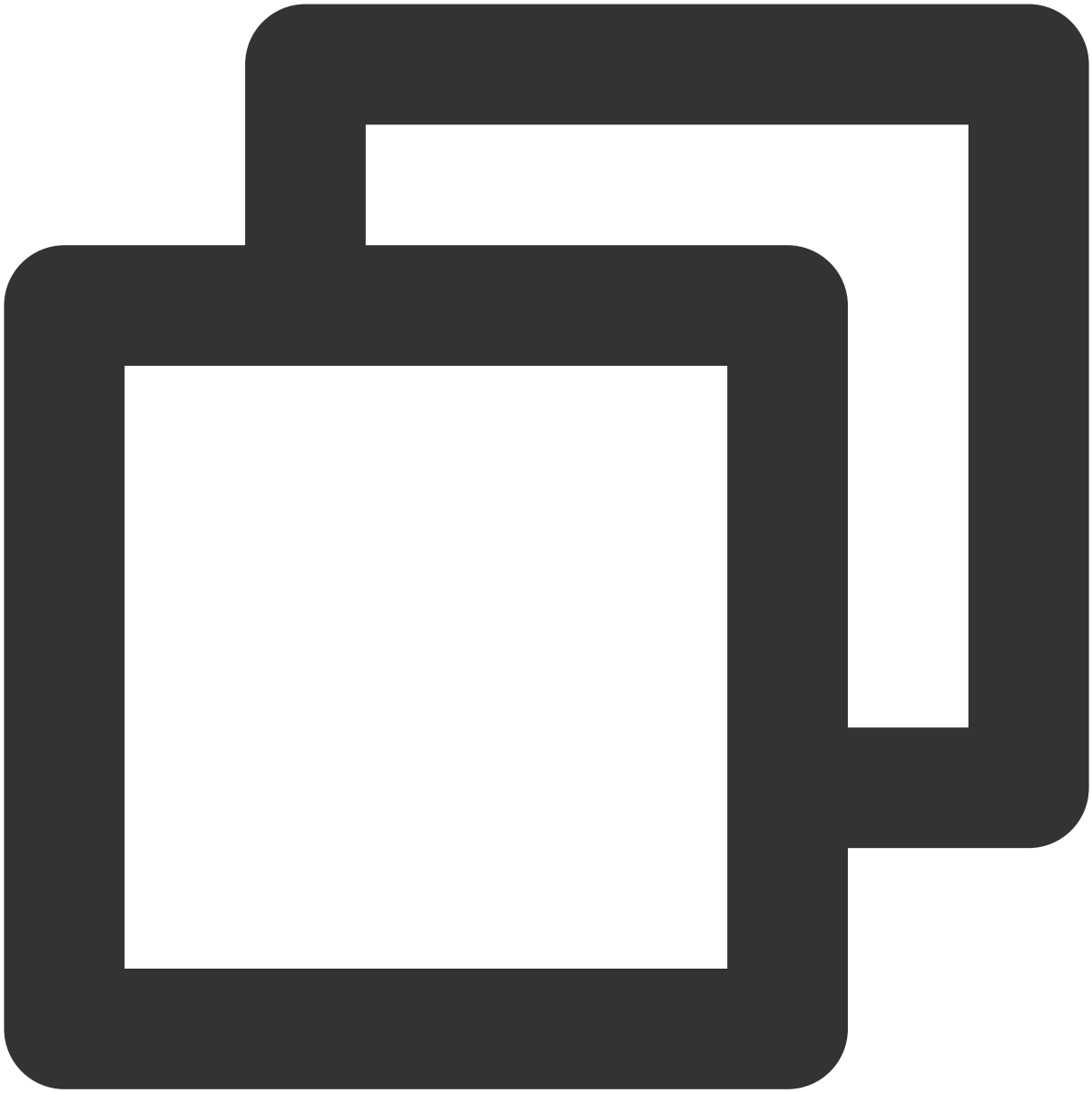
Assumes that the following COS event is received:



```
{
   "specversion":"1.0",
   "id":"13a3f42d-7258-4ada-da6d-023a333b4662",
   "type":"cos:created:object",
```

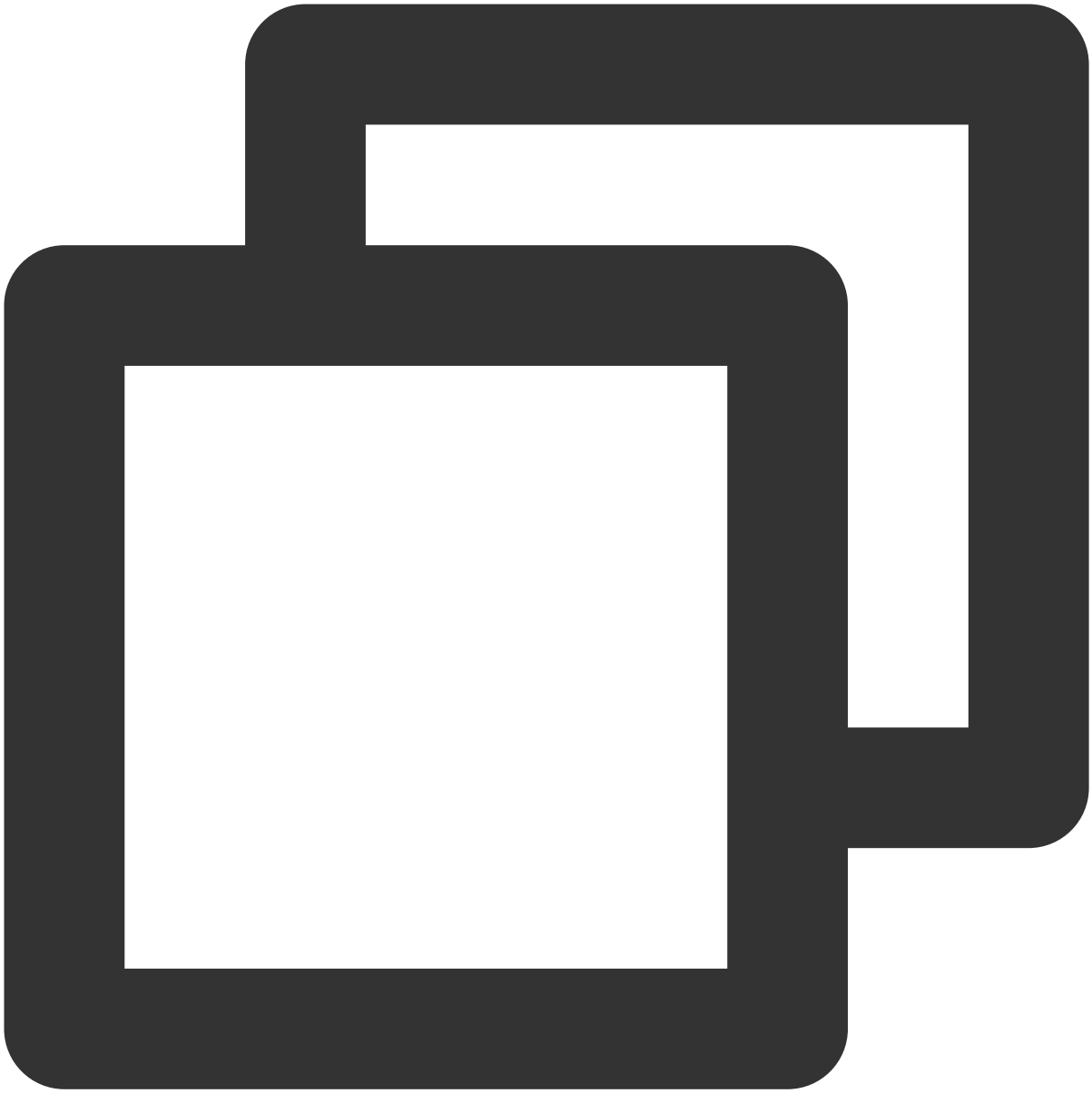
```
    "source":"cos.cloud.tencent",
    "subject":"qcs::cos:ap-guangzhou:uid1250000000:bucketname",
    "time":"1615430559146",
    "region":"ap-guangzhou",
    "datacontenttype": "application/json;charset=utf-8",
    "resource":[
     "qcs::eb:ap-guangzhou:uid1250000000:eventbusid/eventruleid"
    ],
    "data":{
       "name":"testname",
       "scope":100
    }
}
```

To match events with anything but `teset1` in the `name` of `data`, the statement should be:



```
{
    "data": {
        "name": [{
            "anything-but":"test1"
        }]
    }
}
```

To match the event with anything by the specified value in the `name` of `data`, the statement should be:

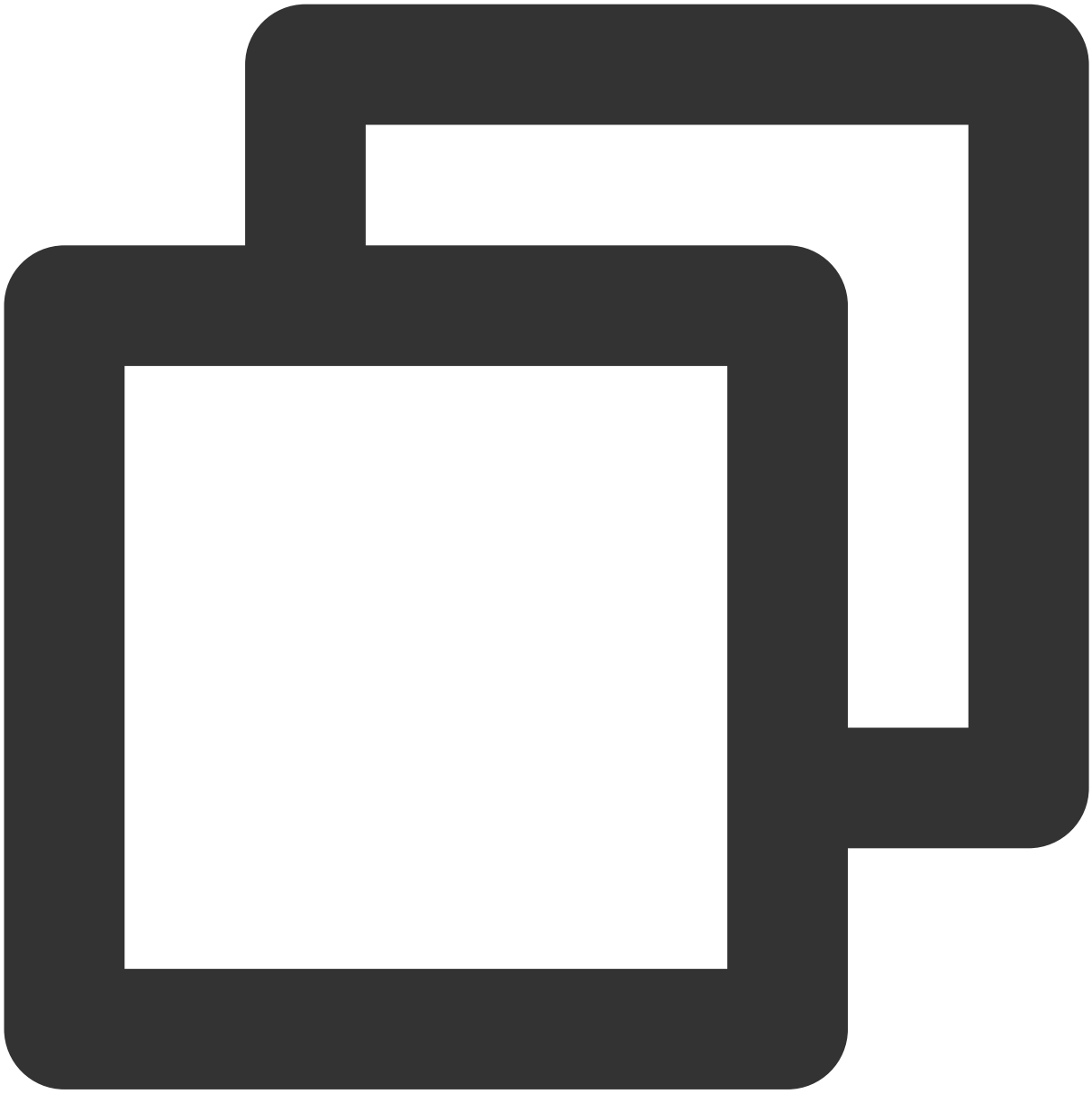


```
{
    "data": {
        "name": [{
            "anything-but":"testname"
        }]
    }
}
```

Inclusion Matching

You can match events with the specified values in the specified field of `data` by using a statement like `{ "contain": ".txt" }`.

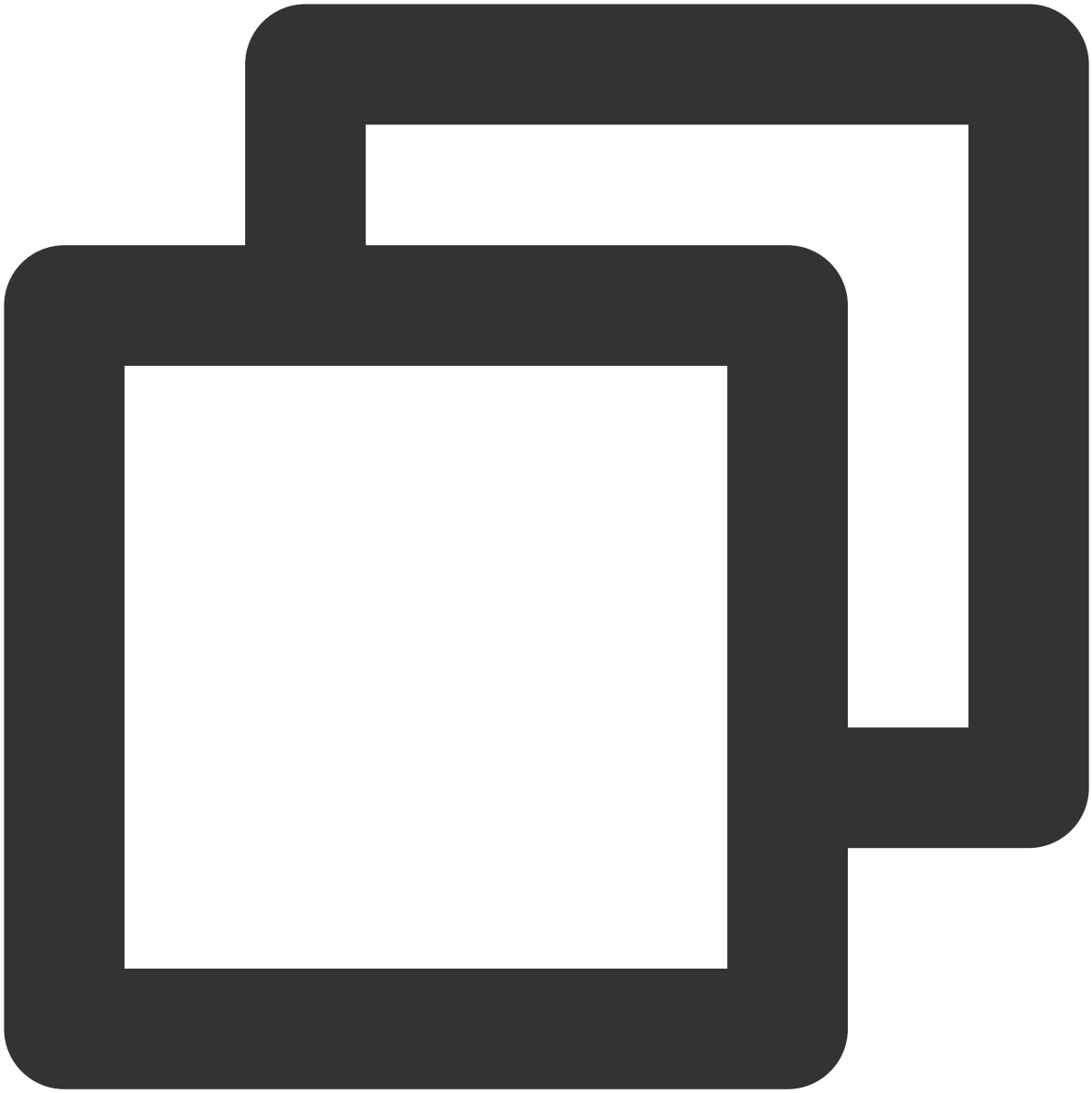
Assumes that the following TDMQ event is received:



```
{
    "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",
```

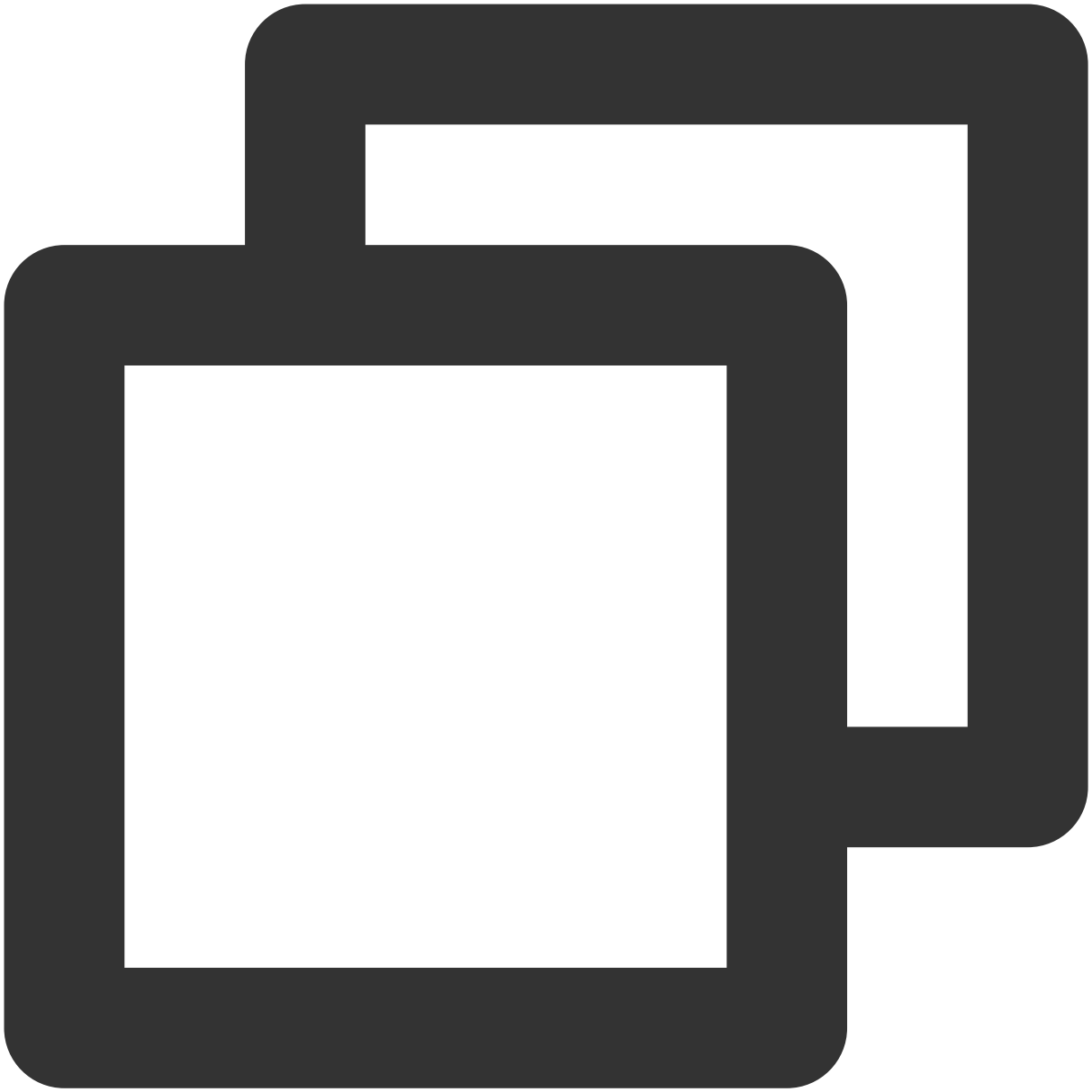
```
    "region": "ap-guangzhou",
    "datacontenttype": "application/json;charset=utf-8",
    "data": {
                    "topic":  "persistent://appid/namespace/topic-1",
                    "tags": "testtopic",
                    "TopicType": "0",
                    "subscriptionName": "xxxxxx",
                    "toTimestamp": "1603352765001",
                    "partitions": "0",
                    "msgId": "123345346",
                    "msgBody": "Hello from TDMQ!"
    }
}
```

To match the event with the specified `topic` value in `data`, the statement should be:



```
{
    "data": {
        "topic": [{
            "contain":"topic-1"
        }]
    }
}
```

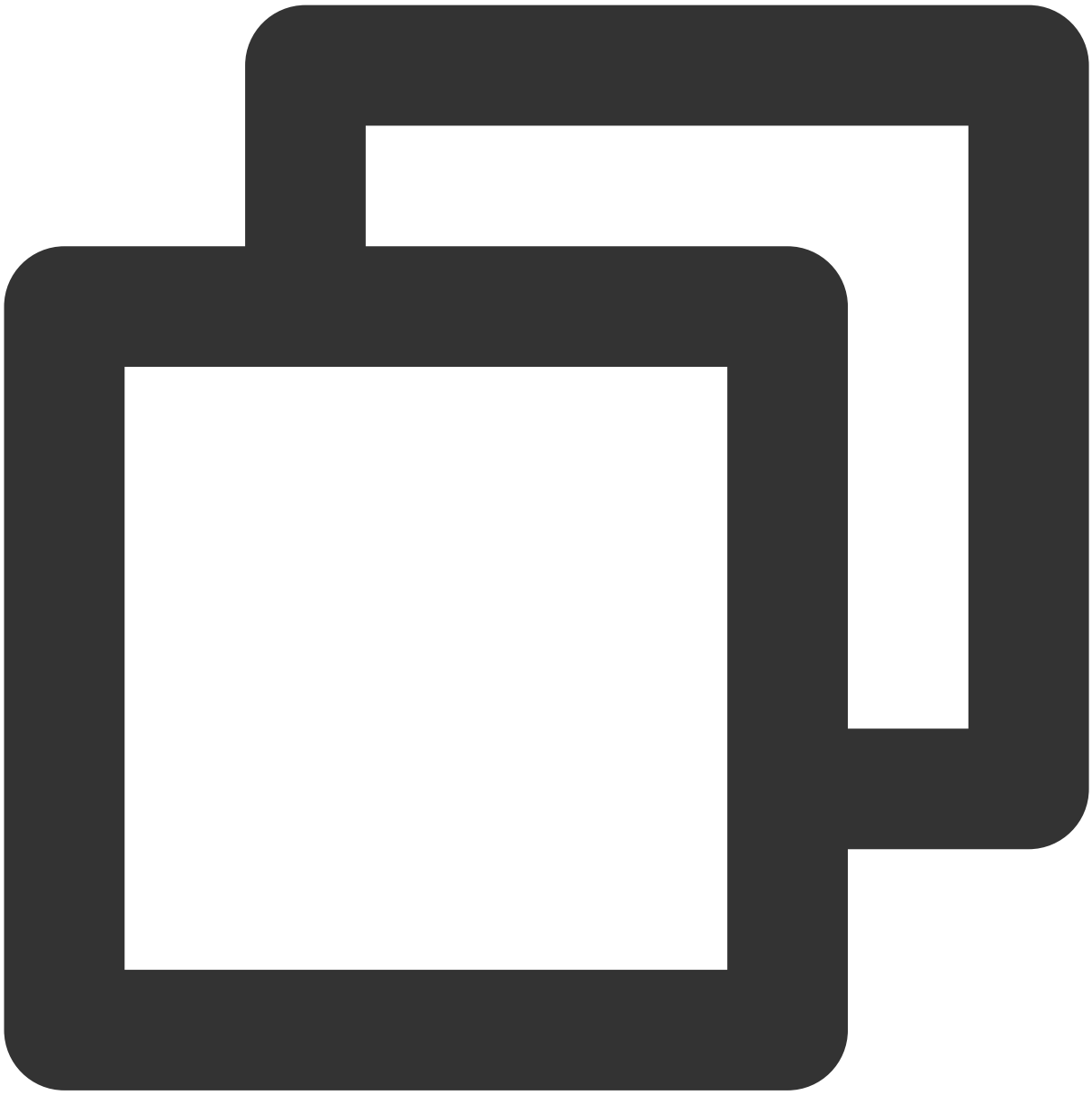
To match the event that includes all the specified `topic` values in `data`, the statement should be:



```
{
    "data": {
        "topic": [{
            "contain":["topic-1","appid"]
        }]
    }
}
```

Array Matching

You can filter array fields with a syntax statement, such as `{"array": "{\\"key1\\":\\"value1\\"}"}`. Assumes that the following DTS event is received:

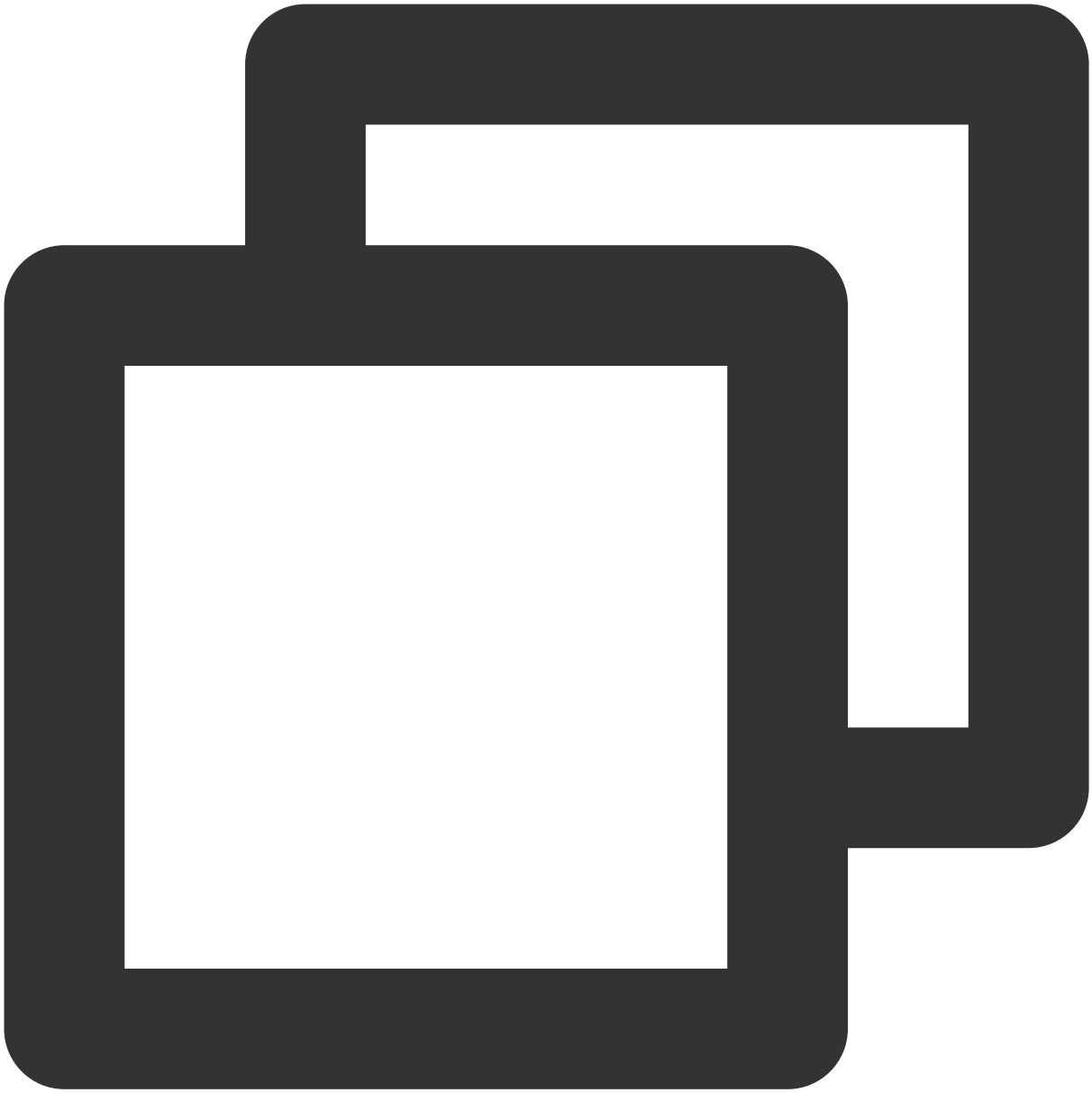


```
{
  "id": "13a3f42d-7258-4ada-da6d-023a33******",
  "type": "dts:mysql:update",
  "specversion": "1.0",
  "source": "dts.cloud.tencent",
  "subject": "cdb-xxx",
  "time": 1660013278609,
  "region": "ap-guangzhou",
```

```
  "dataContentType": "application/json;charset=utf-8",
  "tags": {
    "key1": "value1",
    "key2": "value2"
  },
  "data": {
    "topic": "topic-subs-xxx-cdb-xxx",
    "partition": 0,
    "offset": 72235,
    "partition_seq": 72236,
    "event": {
      "dmlEvent": {
        "dmlEventType": 1,
        "columns": [
          {
            "name": "time",
            "originalType": "time"
          },
          {
            "name": "id",
            "originalType": "int(11)",
            "isKey": true
          }
        ],
        "rows": [
          {
            "oldColumns": [
              {
                "dataType": 13,
                "charset": "utf8",
                "bv": "c3NzYWFhcWFxMTEx"
              }
            ],
            "newColumns": [
              {
                "dataType": 13,
                "charset": "utf8",
                "bv": "MjA6MTI6MjI="
              }
            ]
          }
        ]
      }
    },
    "header": {
      "sourceType": 1,
      "messageType": 2,
```

```
        "timestamp": 1648555949,
        "serverId": 109741,
        "fileName": "mysql-bin.000005",
        "position": 11172920,
        "gtid": "38cecd93-a9c2-11ec-b952-043f72d8da53:55",
        "schemaName": "dts",
        "tableName": "dts_mysql",
        "seqId": 72286,
        "isLast": true
      },
      "eb_consumer_time": "2022-03-29T20:12:29+08:00",
      "eb_connector": "cdb-xxx"
    }
}
```

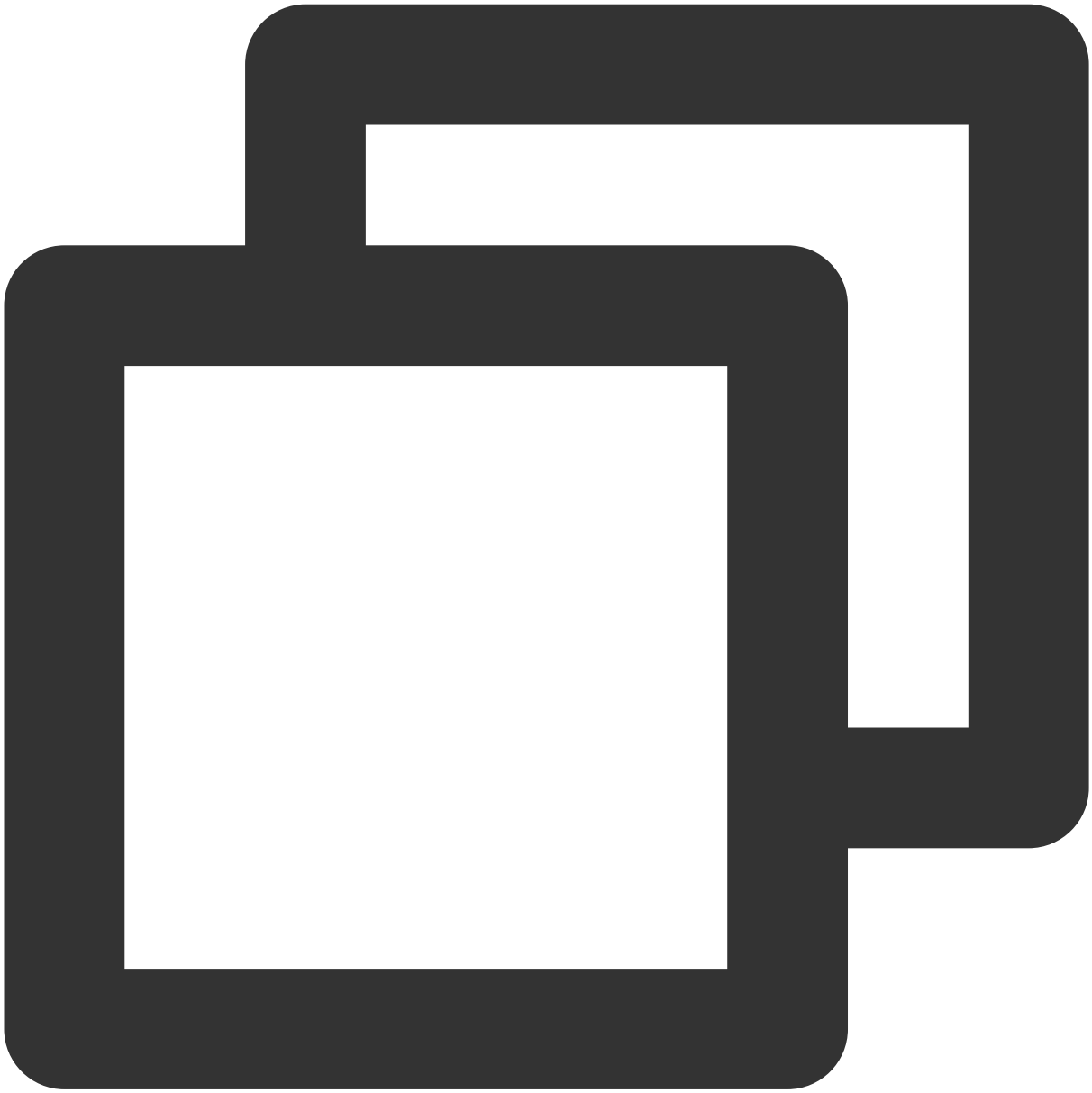
To match the event via the `columns` fields, the statement should be:



```
{
    "source": "dts.cloud.tencent",
    "type": "dts:mysql:update",
    "data": {
        "event": {
            "dmlEvent": {
                "columns": [{
                    "array": "{\\"name\\":\\"time\\"}"
                }]
            }
        }
```

```
    }
}
```

Multiple filters are combined with AND.



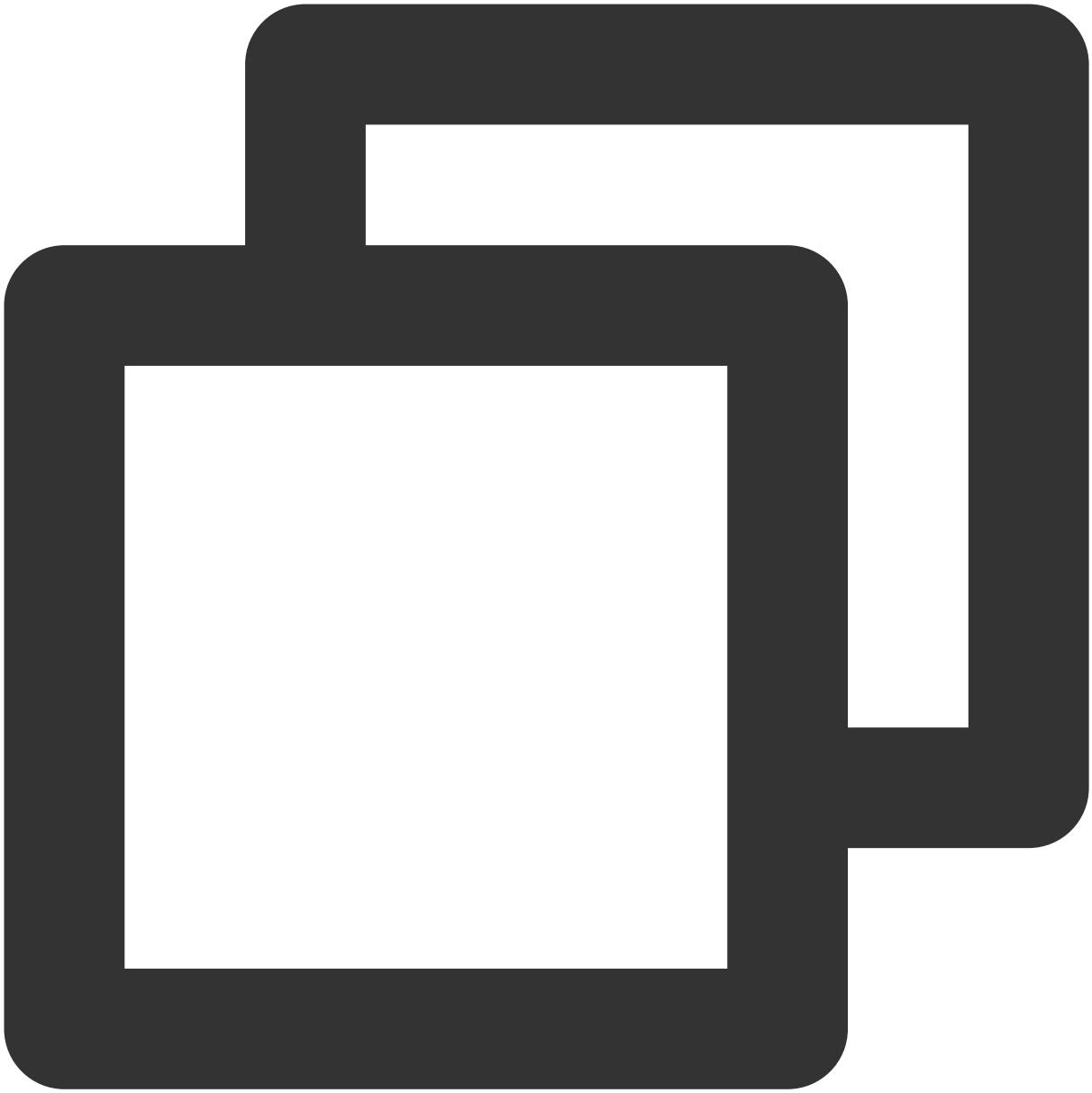
```
{
    "source": "dts.cloud.tencent",
    "type": "dts:mysql:update",
    "data": {
        "event": {
            "dmlEvent": {
                "columns": [{
```

```
                        "array": "{\\"name\\":\\"id\\",\\"originalType\\":\\"int(11)\\"
                                }]
                }
            }
        }
    }
```

IP Matching

You can specify an IP address in the `data` field as a match condition. For example, if the statement is `{ "cidr": "10.0.0.0/24" }`, events whose IP is within the `10.0.0.0/24` IP range are returned.

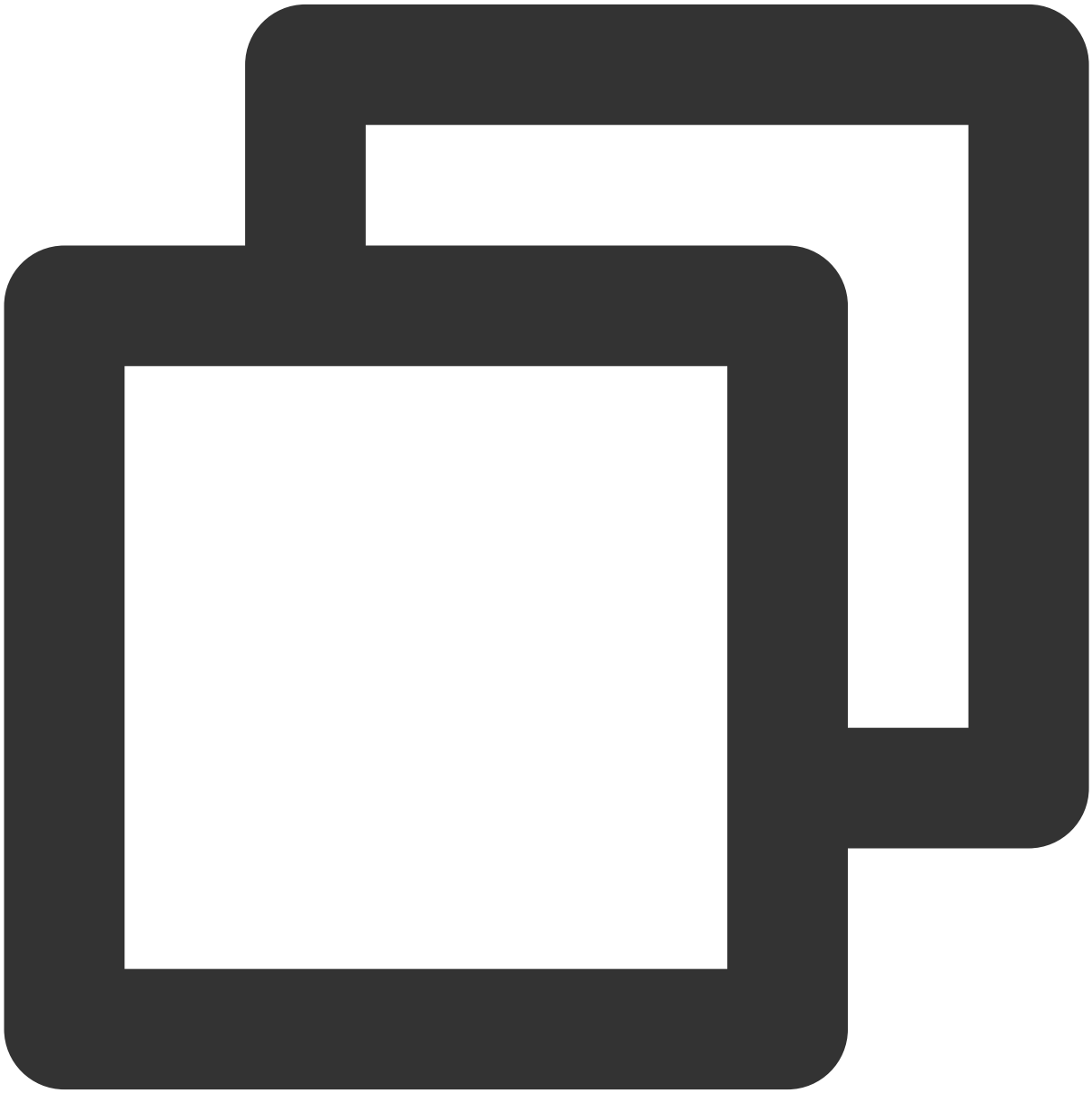
Assumes that the following COS event is received:



```
{
    "specversion": "1.0",
    "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",
    "resource": [
        "qcs::eb:ap-guangzhou:uid1250000000:eventbusid/eventruleid"
```

```
    ],
    "data": {
        "name": "testname",
        "scope": 100,
        "source-ip": "10.0.0.123"
    }
}
```

To match the event with the specified `source-ip`, the statement should be:



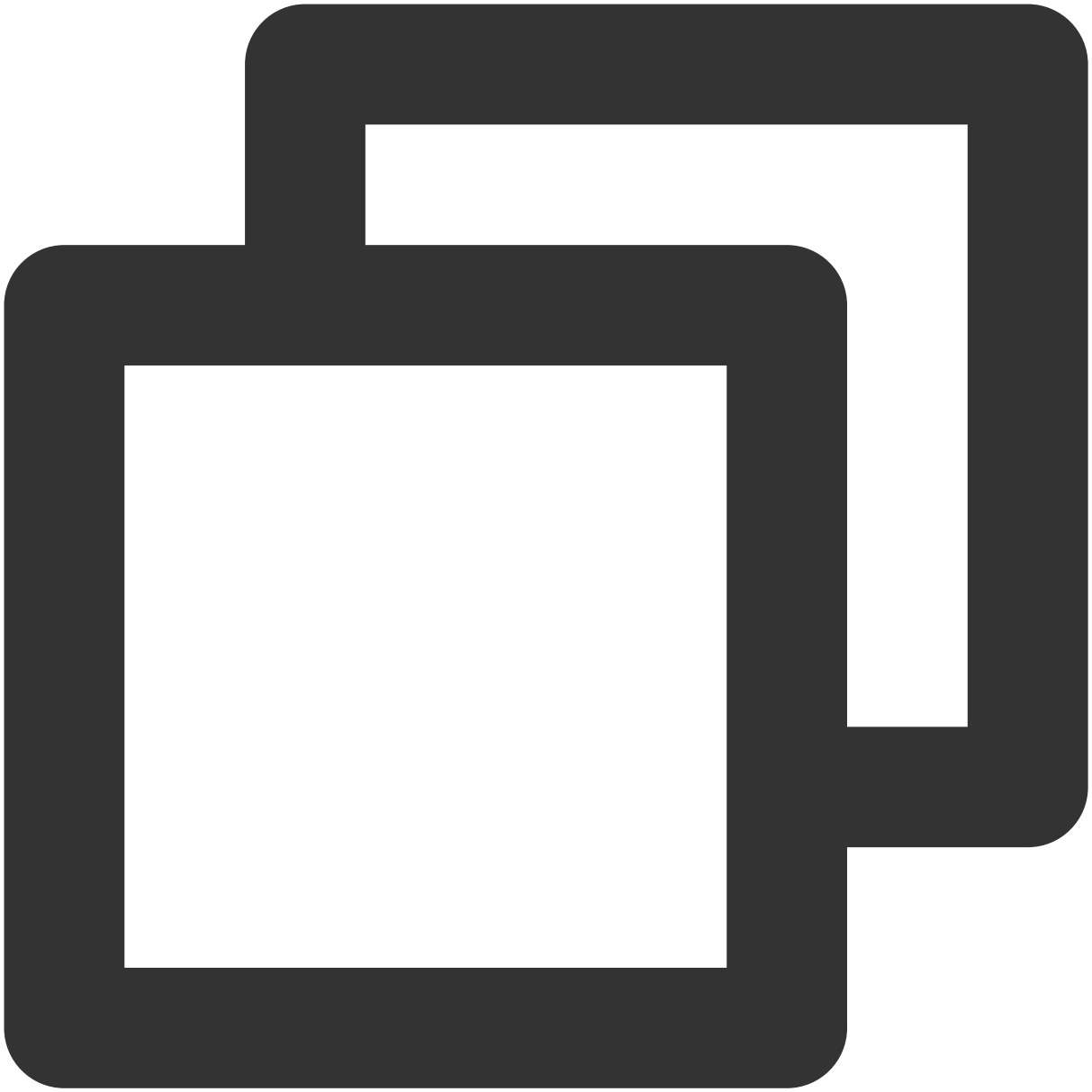
```
{
```

```
    "data": {
        "source-ip": [{
            "cidr": "10.0.0.0/24"
        }]
    }
}
```

More

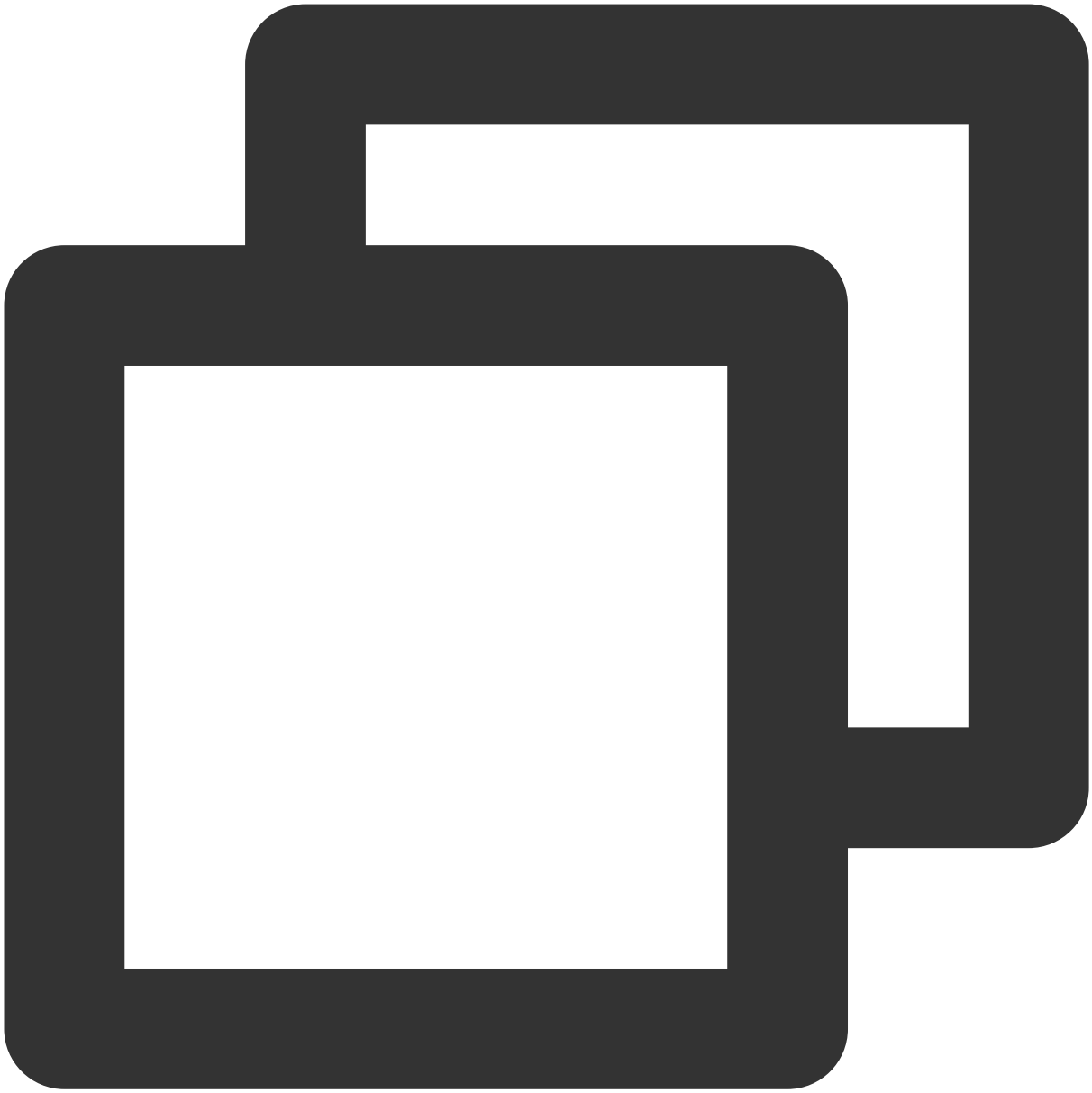
A null value is different from an empty string, and null values do not match with a pattern that is used to match empty strings.

All match patterns can be nested. In the following sample, exclusion match and prefix match are nested:



```
{
  "data": {
      "name": [{
          "anything-but": {
              "prefix": "init"
          }
      }]
  }
}
```

OR is supported in all matching modes. You can specify the prefix and suffix as below:



```
{
  "data": {
      "topic": [
        {
          "prefix":"pre"
        },
        {
          "suffix":"suf"
        }
      ]
  }
```

```
}
```

Creating Event Rule

Last updated：2024-07-23 15:08:07

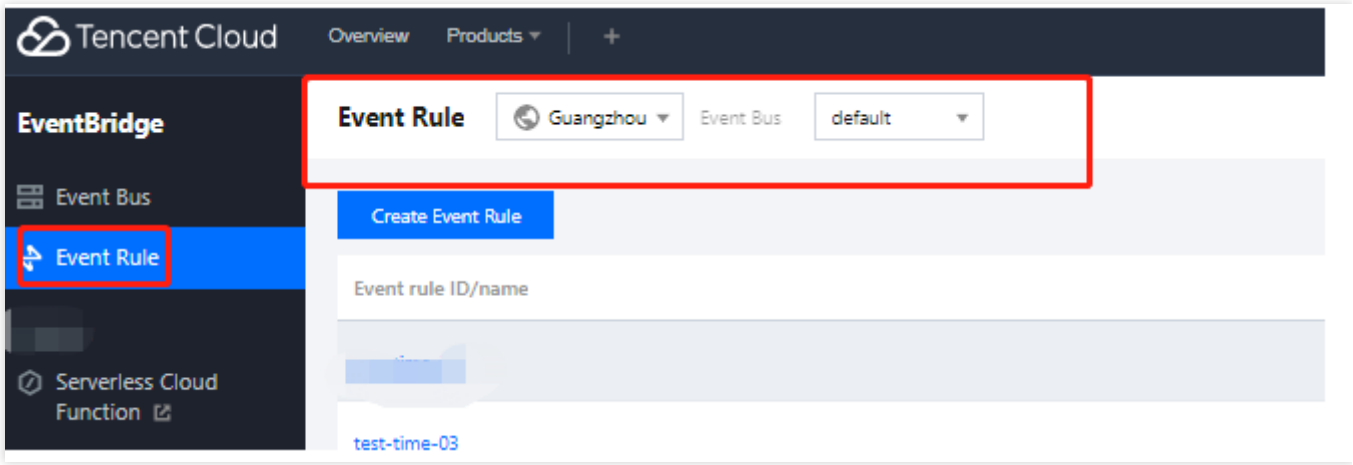
Event rule is one of the most basic resource units in EventBridge. You can configure an event bus to receive events from an event source and use event rules to filter events. This document describes how to create an event rule in the EventBridge console.

Prerequisites

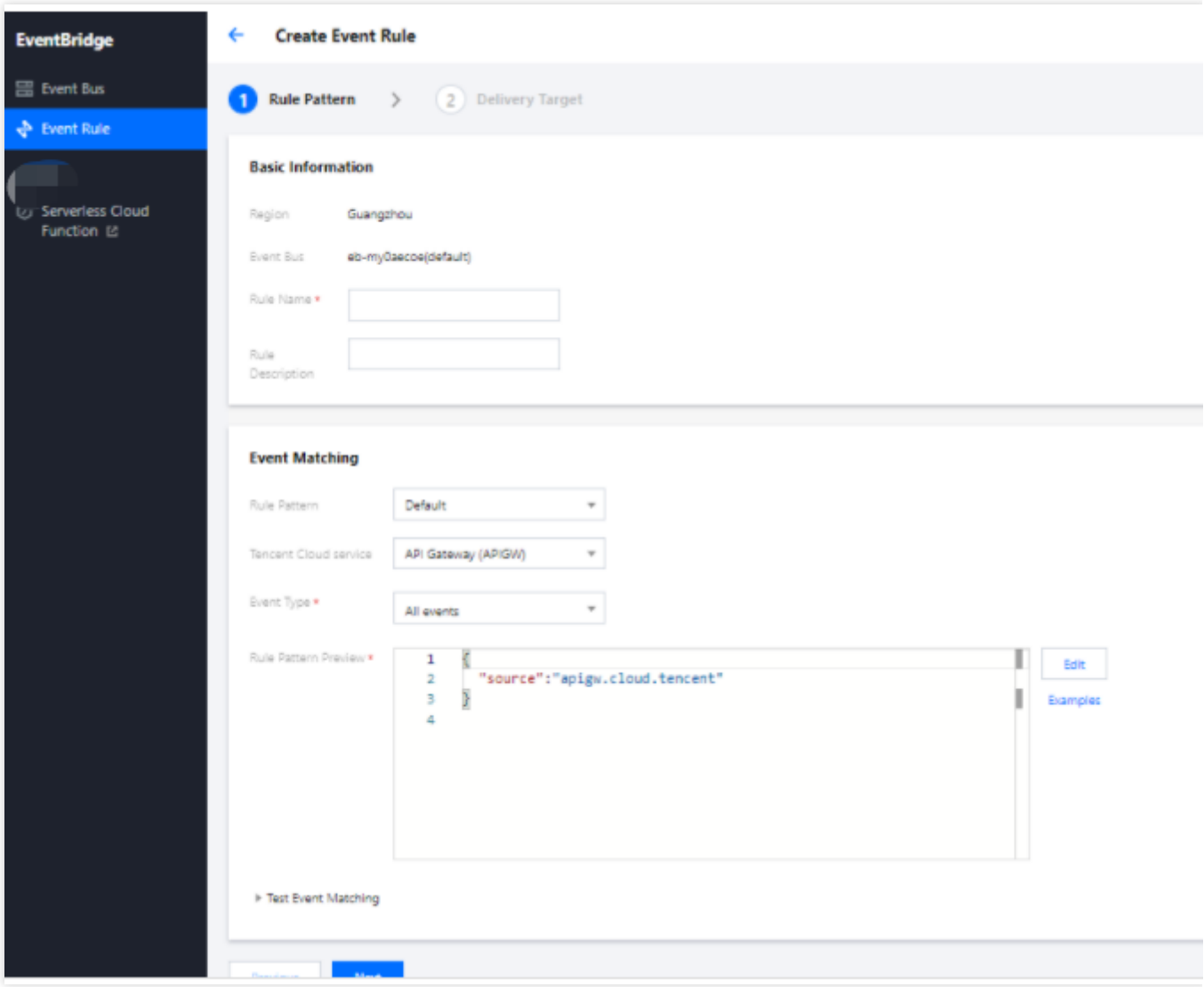
You have created an event bus.

Directions

1. Log in to the EventBridge console and select **Event Rule** on the left sidebar.
2. At the top of the **Event Rule** list page, select the event bus and region for the event rule to be created.



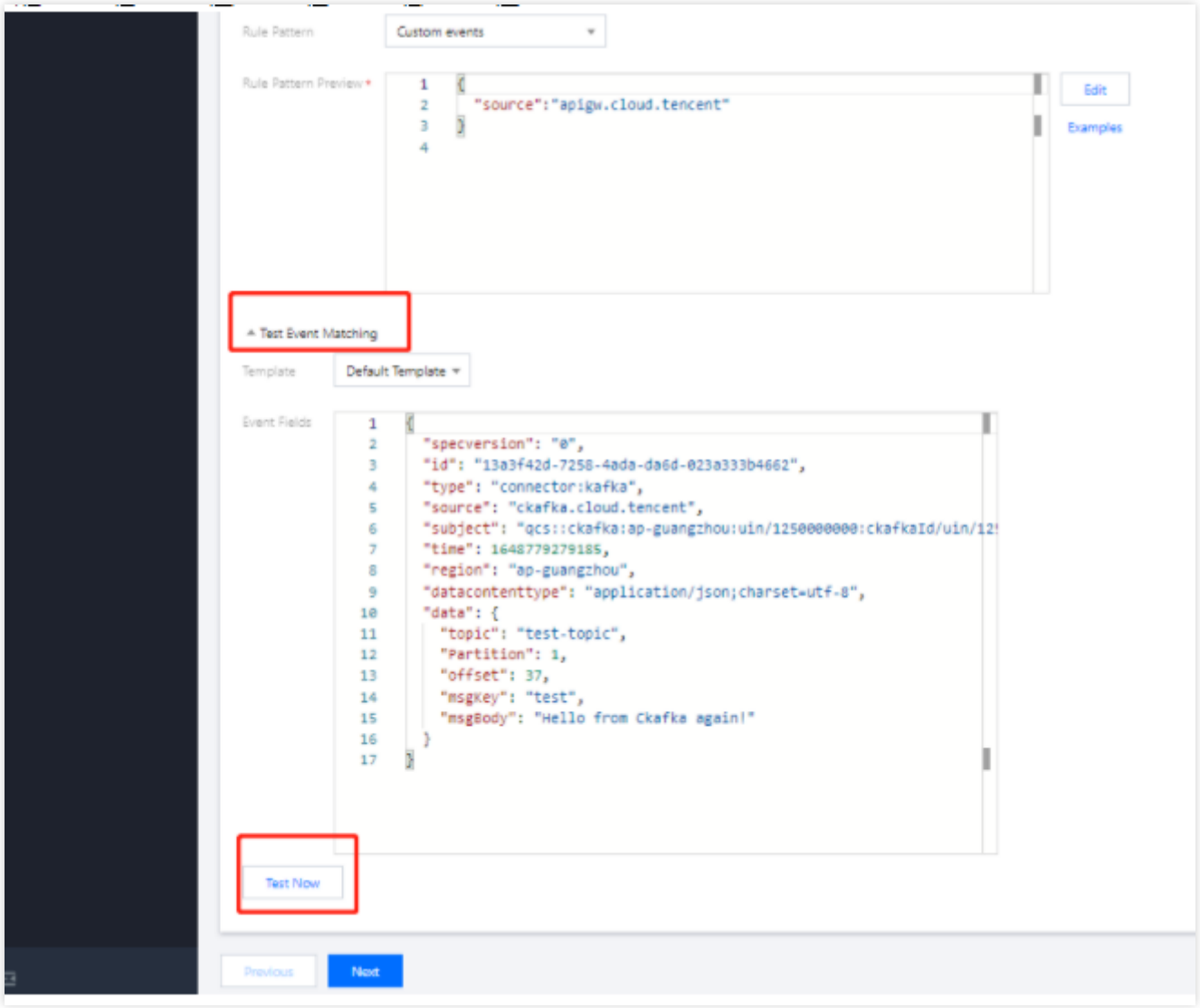
3. Click **Create Event Rule** and enter the relevant information as prompted as shown below:



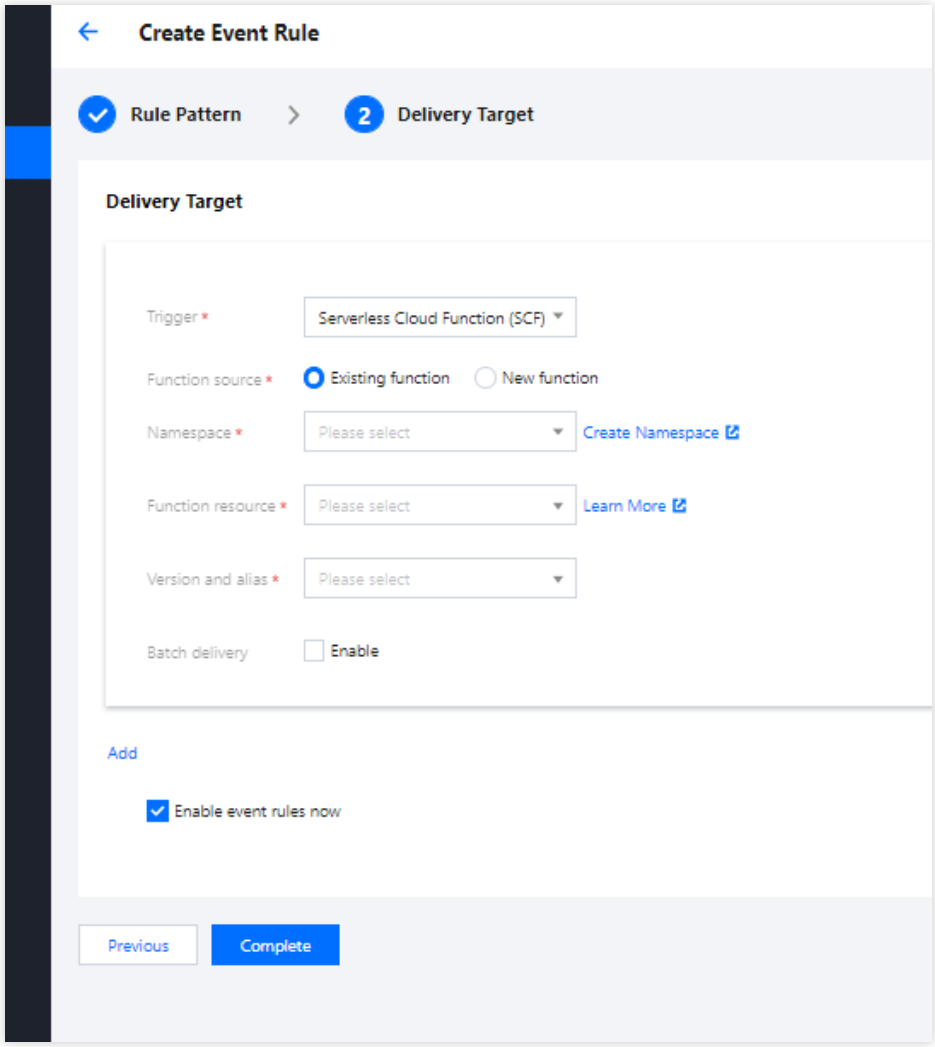
Event matching: Used for event filtering. You can specify what events can be matched. You can configure a custom event matching pattern or select an existing template rule. As shown in the figure above, all events from the TDMQ message queue can be matched. For more information about event pattern rules, see Event Pattern.

Delivery target: Specifies the target that is eventually triggered by events.

4. Click to expand **Test Event Matching**. Then you can test the defined event pattern. **Template** is preset with all currently supported templates for events generated by Tencent Cloud services and templates for events generated by connectors.



5. Click **Next** and select the event targets to bind to the current event rule. An event rule can be bound to multiple event targets.



EventBridge supports the following event targets, and you can click the links to view their configuration methods:

Serverless Cloud Function (SCF)

CKafka

Managing Event Rule

Last updated：2024-07-23 15:08:07

This document describes how to view, edit, and delete an event rule.

Preparation

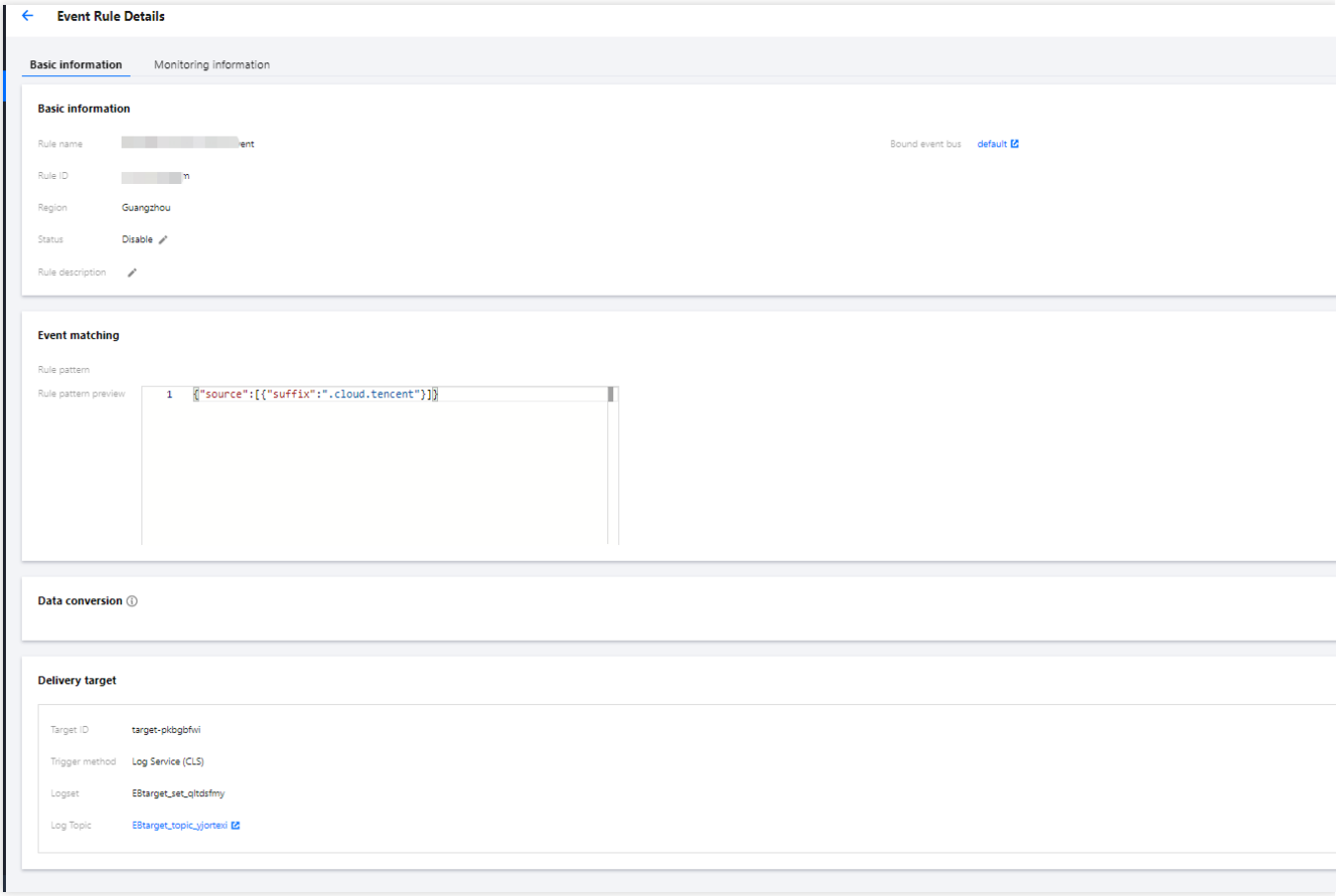
You have created an event rule.

Directions

Viewing event rule

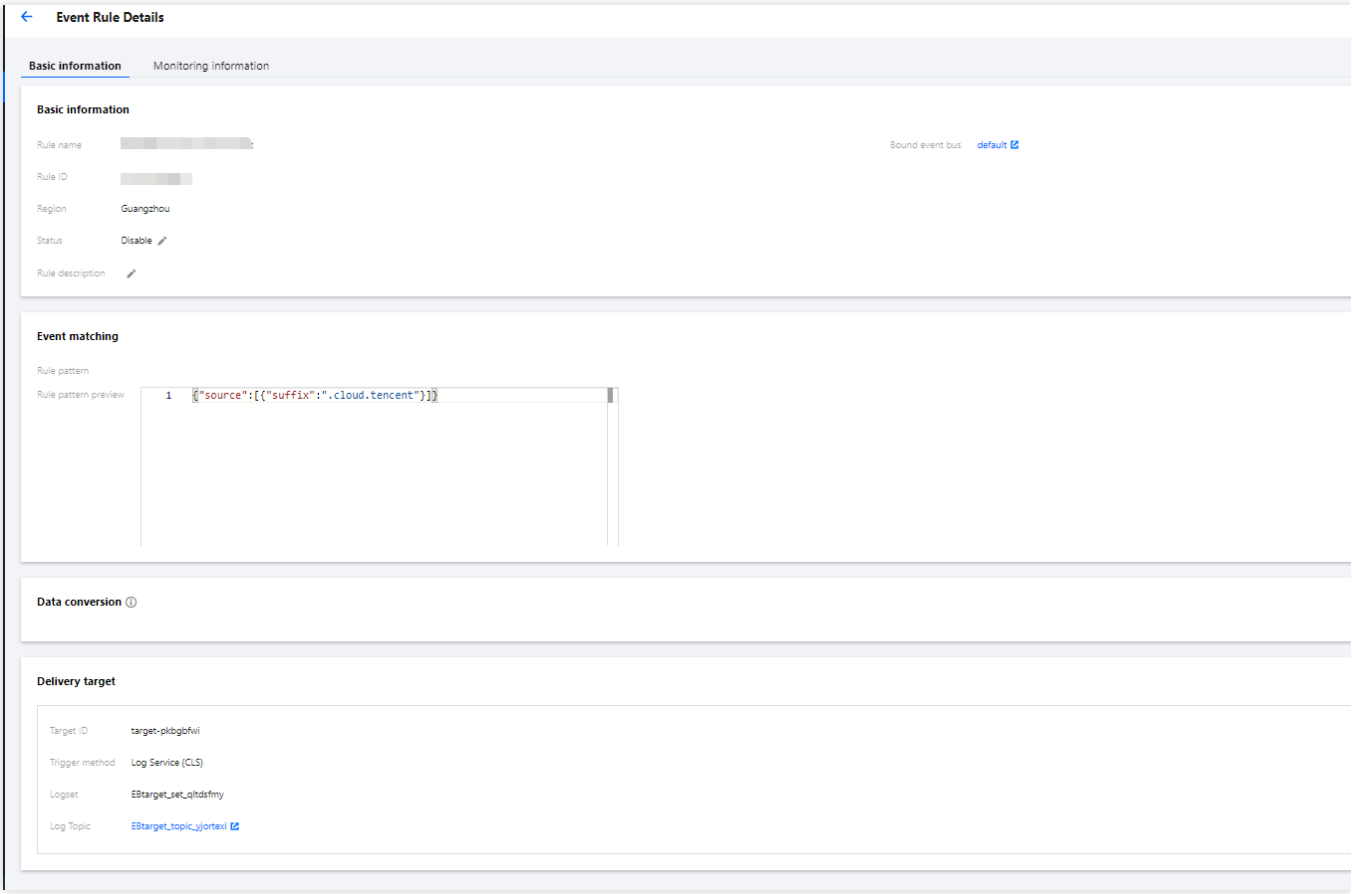
You can view the details of an event bus as follows:

1. Log in to the EventBridge console and select **Event Rule** on the left sidebar.
2. In the drop-down lists at the top of the page, select a region and event bus.
3. In the event rule list, click the target event rule.
4. The event rule page displays the event rule information such as basic information, event match information, and event target as shown below:



Editing event rule

1. Log in to the EventBridge console and select **Event Rule** on the left sidebar.
2. In the drop-down lists at the top of the page, select a region and event bus.
3. In the event rule list, click the target event rule.
4. On the event rule page, you can edit basic event information and event match information and delete/add event targets as shown below:

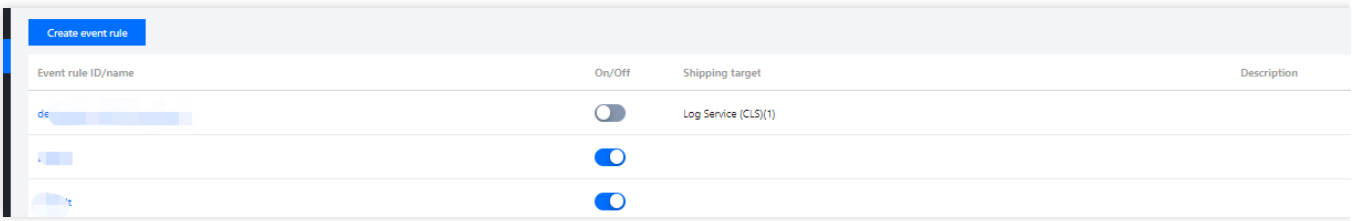


Deleting event rule

Note:

Before deleting an event rule, make sure that all event targets under the rule have been deleted.

1. Log in to the EventBridge console and select **Event Rule** on the left sidebar.
2. In the drop-down lists at the top of the page, select a region and event bus.
3. In the event rule list, find the target event rule and click **Delete** in the **Operation** column on the right as shown below:



4. In the pop-up window, click **OK**.

Configuring Data Conversion

Last updated：2024-07-23 15:08:07

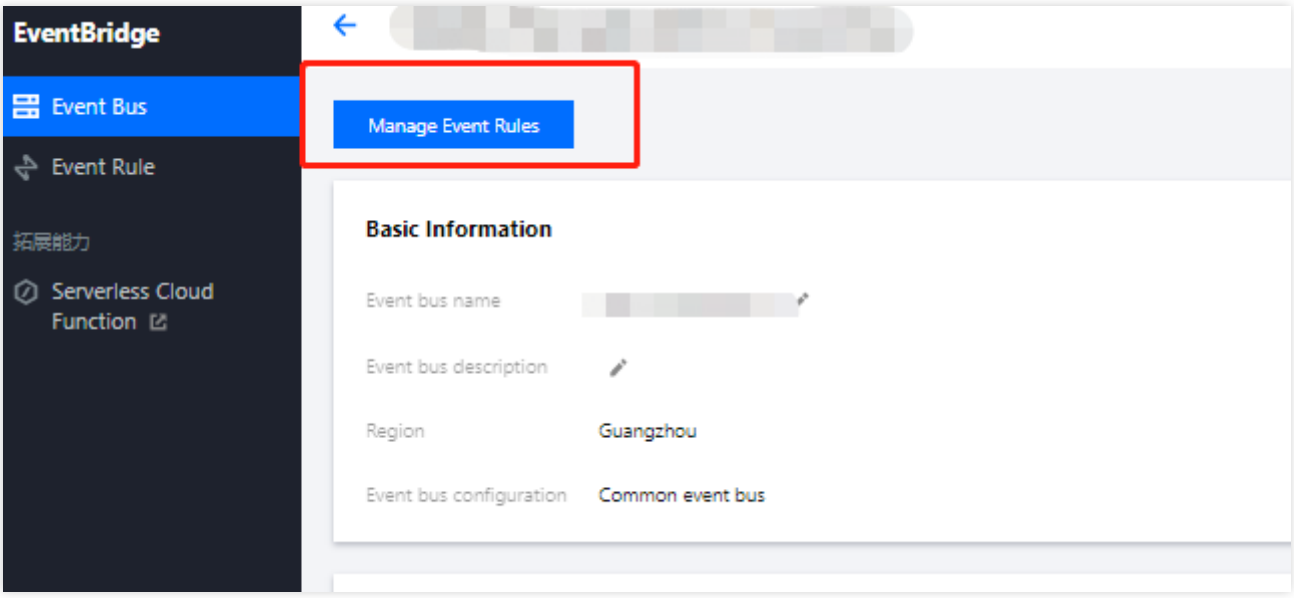
Overview

In addition to basic event filtering, EventBridge provides simple data processing capabilities. After you pass in data and configuration items to EventBridge, EventBridge formats the data and distributes the structured data obtained after processing to downstream targets, creating a bridge between data sources and data processing systems.

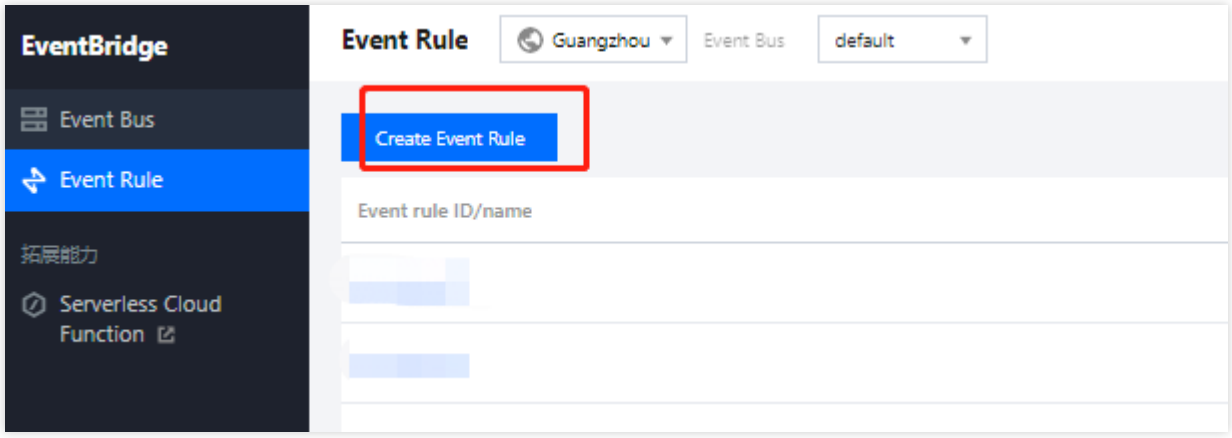
Directions

Creating a rule

1. Log in to the EventBridge console and select a specified event bus.
2. On the event bus details page, click **Manage Event Rules** and configure a new rule as shown below:



3. Go to the **Event Rule** page and click **Create Event Rule**.



4. Enter the basic task information as instructed and select **Enable data conversion**.

5. Click **Next** and set the data conversion rule.

Rule pattern: Select **Template data** or **Custom**.

Parsing mode: Select **JSON**.

6. After selecting a parsing mode, click **OK** to start data parsing.

7. After data parsing is completed, set the filtering rule and data processing mode.

Note:

Currently, the output format is JSON.

Filter: Only data that meets the configured filter rule is output. The filter supports the following matching modes: **Prefix match**, **Suffix match**, **Inclusion match (contains)**, **Exclusion match (except)**, **Data matching**, and **IP matching**.

Data processing: Valid values of `TYPE` are **Default**, **Preset**, **Mapping**, **Custom**.

TYPE = Default: `VALUE` is mapped based on the parsing result and cannot be modified.

TYPE = Preset: You can select a system preset value for `VALUE` . Currently, `DATE` (timestamp) is supported.

TYPE = Mapping: You can select an existing key. The final output value of `VALUE` is mapped by the specified key.

TYPE = Custom: You can enter a custom value for `VALUE` .

8. Click **Test** to check the test result.

9. Click **Next** to complete data target binding.

Editing a rule

On the **Event Rule Details** page, click **Edit** in the upper-right corner of the **Data conversion** module to modify a data processing rule. You can also add or delete a data processing rule on the page.

Filter rule description

The filter allows you to configure filtering rules such as field sizes to filter data. Only data that meets the specified rules will be retained.

Notes

Filter matching is exact matching down to the character and case-sensitive. During matching, no standardized operations will be performed on strings.
Values to be matched must be in JSON format, which include strings and numeric values enclosed in quotation marks as well as keywords not enclosed in quotation marks (`true` , `false` , and `null`).

Prefix match

You can perform key value matching by comparing a specified prefix with the prefix in data.
For example, for data `{"password":"topicname"}` , you can specify `top` as the prefix of the `password` value so that `{"password":"topicname"}` can be normally matched.

Suffix match

You can perform key value matching by comparing a specified suffix with the suffix in data.
For example, for data `{"password":"topicname"}` , you can specify `name` as the suffix of the `password` value so that `{"password":"topicname"}` can be normally matched.

Inclusion match

You can specify a field to be included in data as a match condition.
For example, for data `{"password":"topicname"}` , you can specify `na` to be included in the `password` value so that `{"password":"topicname"}` can be normally matched.

Exclusion match

You can specify a field to be excluded from data as a match condition.
For example, for data `{"password":"topicname"}` , you can specify `topicname` to be excluded from the `password` value so that `{"password":"topicname"}` cannot be normally matched.

Numeric match

You can specify the value or value range of a certain field as a match condition.
For example, for data `{ "numeric": 10}` , you can specify the value of `numeric` to be less than 15 (`<15`) as a match condition so that `{ "numeric": 10}` can be normally matched.
The following are examples of value match rules:
Greater than 10: Enter `>10`
Greater than or equal to 10: Enter `>=10`
Greater than or equal to 10, and less than or equal to 20: Enter `>=10&<=20`
Greater than or equal to 10, or less than or equal to 5: Enter `>=10|<=5`

IP match

You can specify an IP in CIDR notation as a match condition. For example, you can enter `1.2.3.4/24` to match IPs whose leading 24 bits start with "1.2.3.".