TDMQ for CKafka

Getting Started

Product Documentation

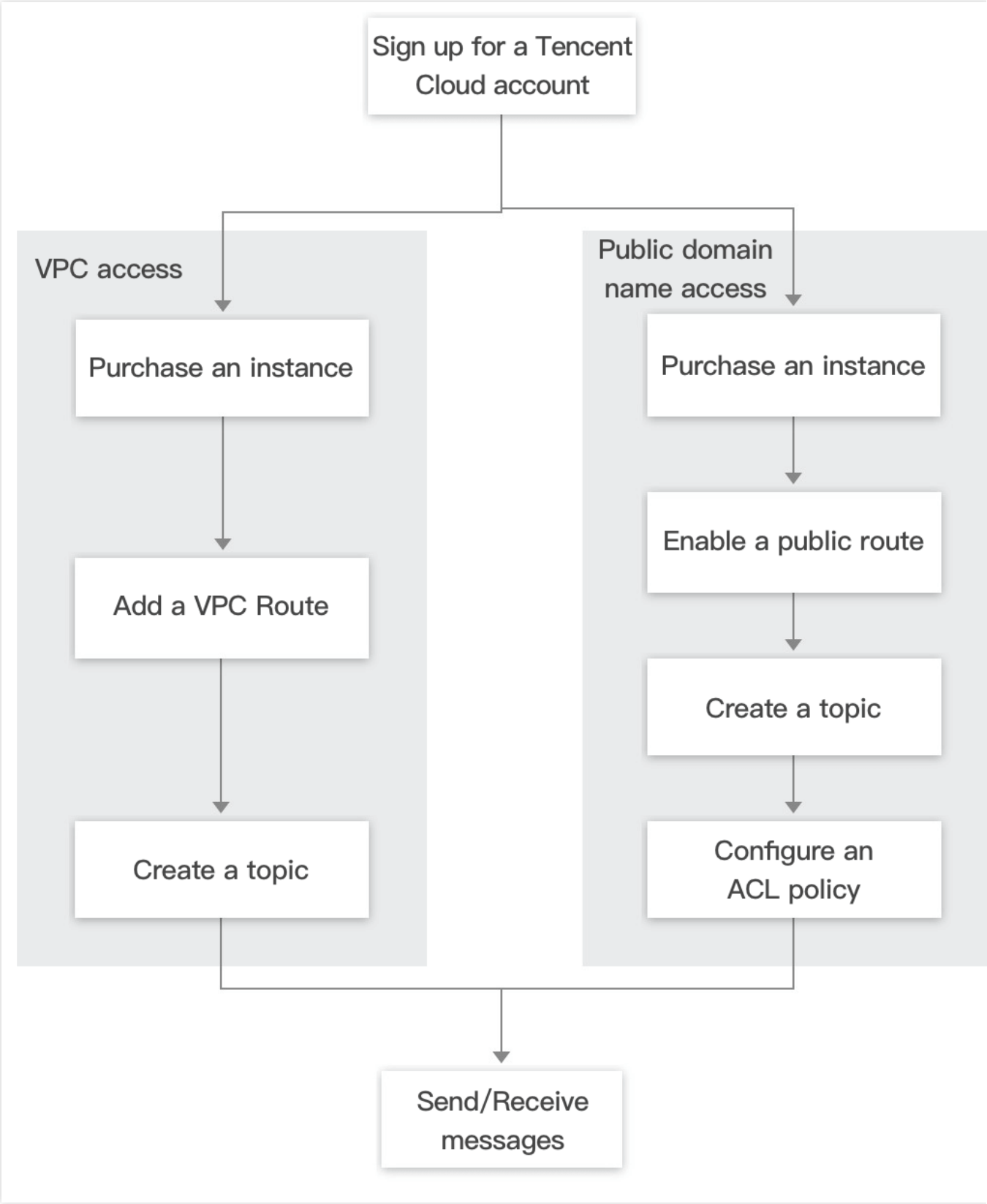
Contents

Getting Started
Process Overview

Last updated：2024-01-09 14:45:02

The process of accessing CKafka varies by network type:
For access via VPC, you can select an appropriate VPC according to your business needs.
For access via public network route, you need to enable a separate public route and configure an ACL policy for the topic.

Flowchart


Sign up for a Tencent Cloud account
VPC access
Purchase an instance
Add a VPC Route
Create a topic
Public domain name access
Purchase an instance
Enable a public route
Create a topic
Configure an ACL policy
Send/Receive messages

Obtaining Access Permission
Getting Access Authorization

Last updated：2024-09-09 21:17:10

CAM Basic Concepts

The root account authorizes sub-accounts by associating policies. These policies can be set with precision across various dimensions, including **[API, Resource, User/User Group, Allow/Deny, and Condition]**.

Account System

Root account: It owns all Tencent Cloud resources and can access any of its resources.
Sub-account: It includes sub-users and collaborators.
Sub-user: It is created and fully owned by a root account.
Collaborator: It already has a root account identity and is added as a collaborator under another root account. This user then becomes a sub-account of the current root account but can switch back to their original root account identity.
Identity credential: It includes login credentials and access certificates. **Login credential** refers to a user's login name and password. **Access certificate** refers to TencentCloud API keys (SecretId and SecretKey).

Resource and Permissions

Resource: An object that is operated in Tencent Cloud Services, such as a CVM instance, a COS bucket, or a VPC instance.
Permissions: It is an authorization that allows or forbids users to perform certain operations. By default, **the root account has full access to all resources under the account**, while **a sub-account does not have access to any resources under its root account**.
Policy: It is a syntax rule that defines and describes one or more permissions. The **root account** performs authorization by **associating policies** with users/user groups.

Using CKafka with Sub-Accounts

When you use CKafka with sub-accounts, two types of permissions need to be granted:
1. In the process of using CKafka, it involves accessing other cloud product resources of the user (VPC, CVM, etc.), such as viewing information about the availability zone where the user's subnet is located. Therefore, sub-accounts need to be granted permissions to access other cloud products. For detailed operations, see Step 1: Granting the Sub-Account Permissions to Access Other Cloud Products .

2. The sub-account also needs to obtain read and write permissions to use CKafka. For detailed operations, see Step 2: Granting the Sub-Account Permissions to Use CKafka .

Step 1: Granting the Sub-Account Permissions to Access Other Cloud Products

Creating a New Custom Policy to Access Other Cloud Products

1. Log in to the **CAM Console**(https://console.tencentcloud.com/cam/overview!4169448268cee04eb156e3de8cf8c971) with the root account.
2. In the left sidebar, select **Policies** , click **Create Custom Policy** .
3. In the pop-up window for selecting policy creation method, select **Create by Policy Syntax** to enter the policy syntax creation page.
4. On the Create by Policy Syntax page, select **Policy Template** , and click **Next** .
5. You can see the interface table and policy syntax below to grant the sub-account appropriate permissions to other cloud products as needed, create the custom policy, fill in all information, and click **Complete** .

The following cloud products are involved in CKafka usage, and the root account needs to separately authorize the sub-account to ensure the use of corresponding CKafka features. The custom policy should include the following cloud product API calls related to CKafka:

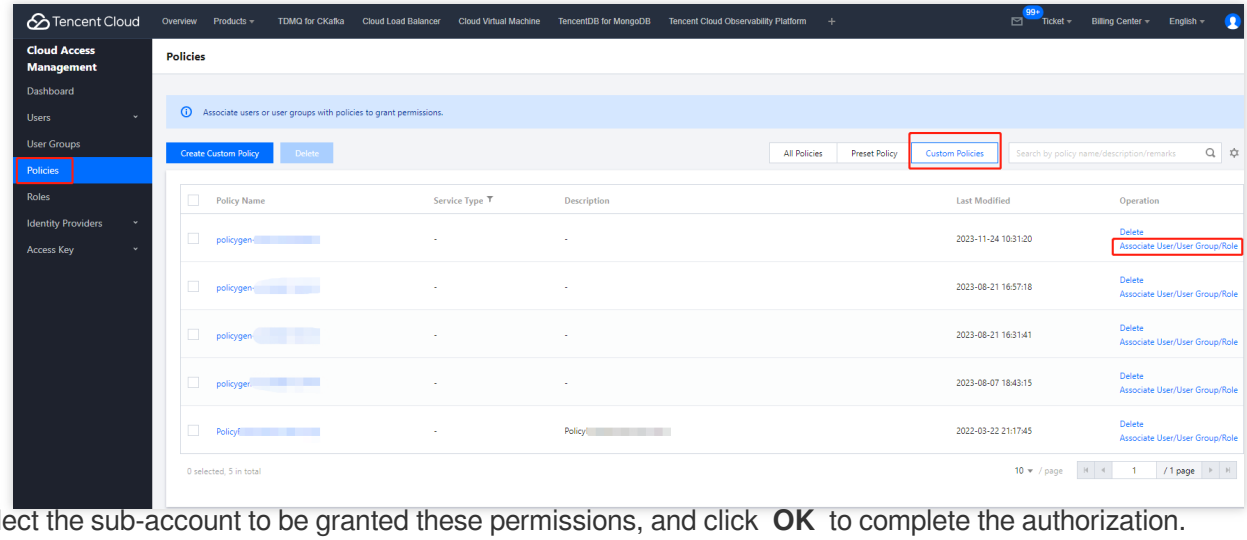
Cloud Products	API Name	API Function	Operations Affecting the TSE platform
Cloud Virtual Machine (CVM)	DescribeZones	Querying Availability Zones	It is used to view the availability zone of a subnet when the instance is created.
Virtual Private Cloud (VPC)	DescribeVpcs	Query VPC list	It is used to select the VPC of the instance access address when the instance is created.
Virtual Private Cloud (VPC)	DescribeSubnets	Query VPC list	It is used to select the subnet of the instance access address when the instance is created.
TCOP (Monitor)	GetMonitorData	Obtain metric monitoring data	It is used to view monitoring data in CKafka.
TCOP (Monitor)	DescribeDashboardMetricData	Obtain metric monitoring data	It is used to view monitoring data in CKafka.

The policy syntax example is as follows:

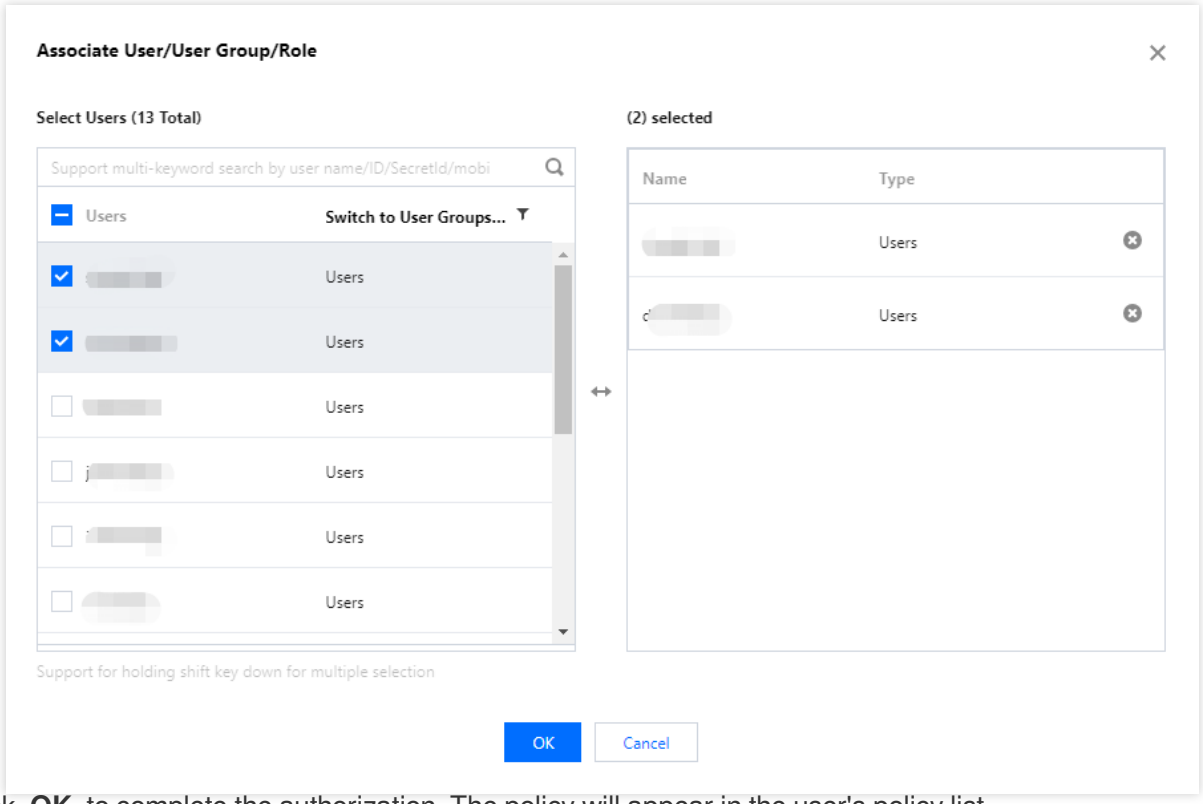
```
{
  "version": "2.0",
  "statement": [
    {
      "effect": "allow",
      "action": [
        "vpc:DescribeVpcEx",
        "vpc:DescribeSubnetEx",
        "monitor:GetMonitorData",
        "monitor:DescribeDashboardMetricData",
      ],
      "resource": [
        "*"
      ]
    }
  ]
}
```

Associating the Custom Policy with the Sub-Account

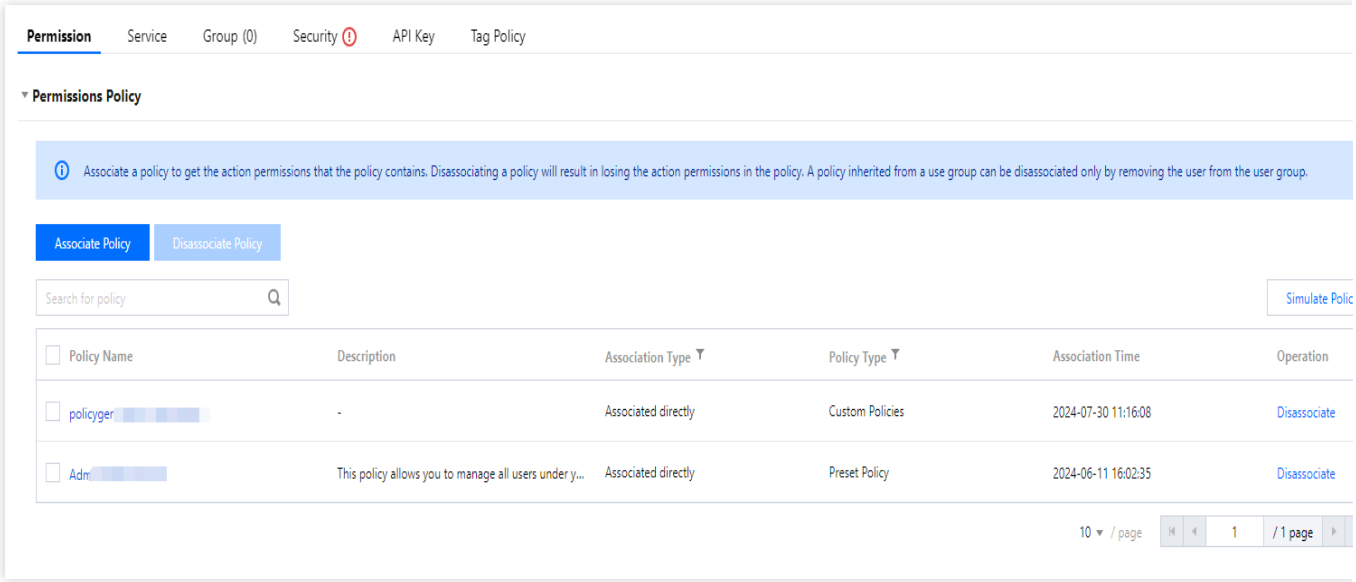
1. Log in to the CAM Console with the root account.
2. On the left sidebar, click **Policies** to enter the policy management page.
3. On the right side, click **Custom Policy** for filtering, find the custom policy created in Step 1.1, and click **Associate User/User Group/Role** in the Operation column.



4. Select the sub-account to be granted these permissions, and click **OK** to complete the authorization.



5. Click **OK** to complete the authorization. The policy will appear in the user's policy list.



Step 2: Granting the Sub-Account Permissions to Use CKafka

See the following documents for related operations:

Granting Operation-Level Permissions to Sub-Accounts

Granting Resource-Level Permissions to Sub-Accounts

Granting Tag-Level Permissions to Sub-Accounts

Granting Operation-Level Permissions to Sub-Accounts

Last updated：2024-01-09 14:45:02

Overview

This document describes how to use the Tencent Cloud root account to authorize sub-accounts at the operation level. You can grant different read and write permissions to sub-accounts as needed.

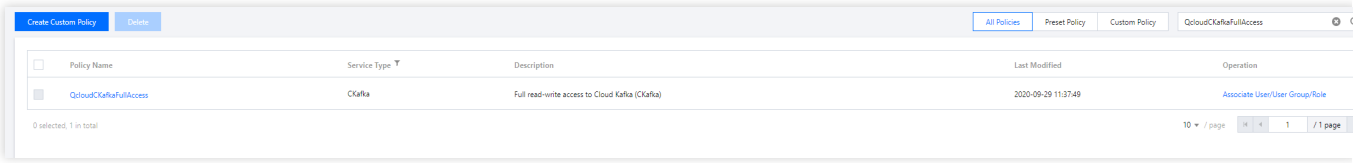
Directions

Full access permission

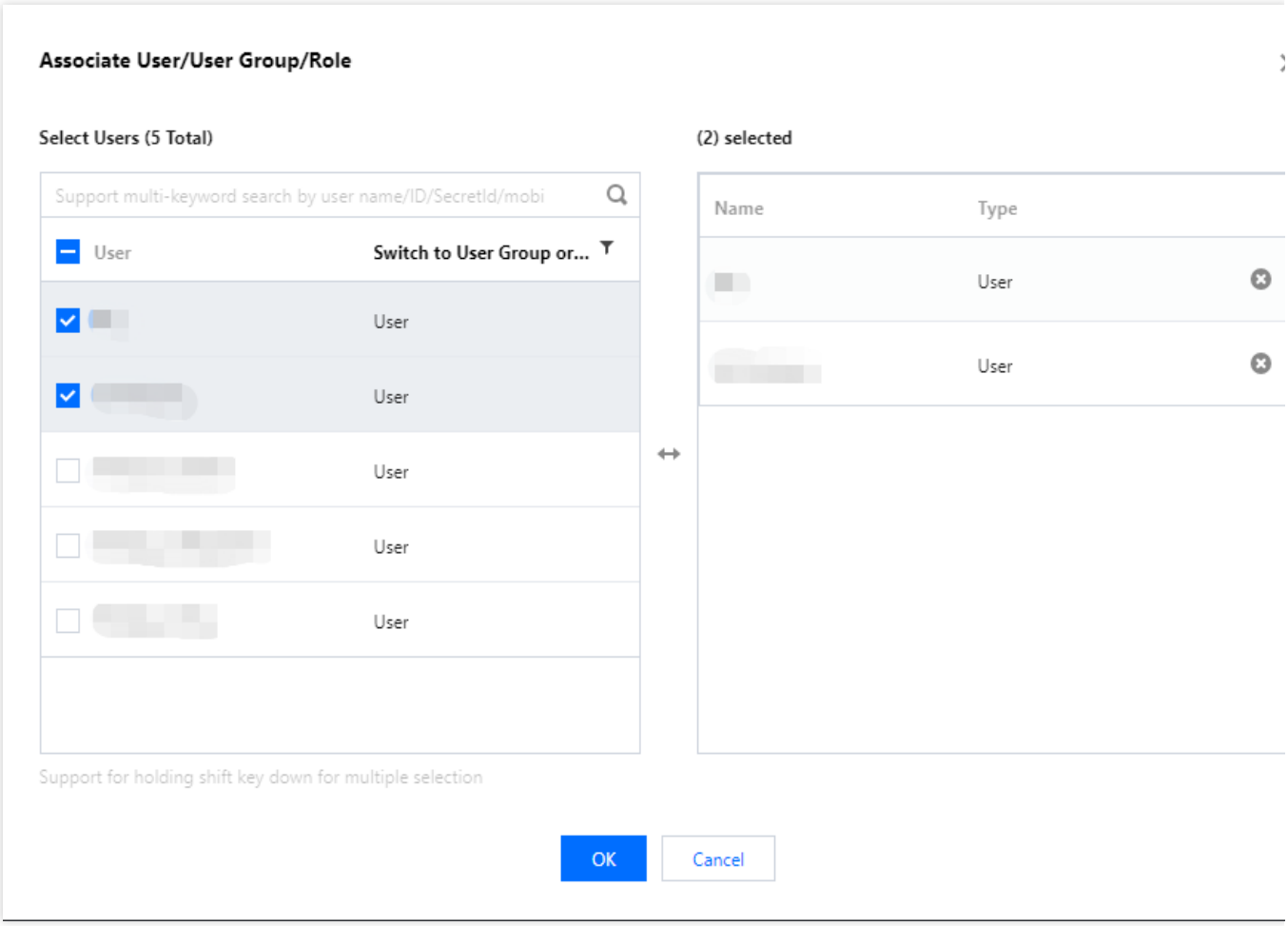
Note:

After granting full access permissions to a sub-account, the sub-account will have **full read and write capabilities** to **all resources** under the root account.

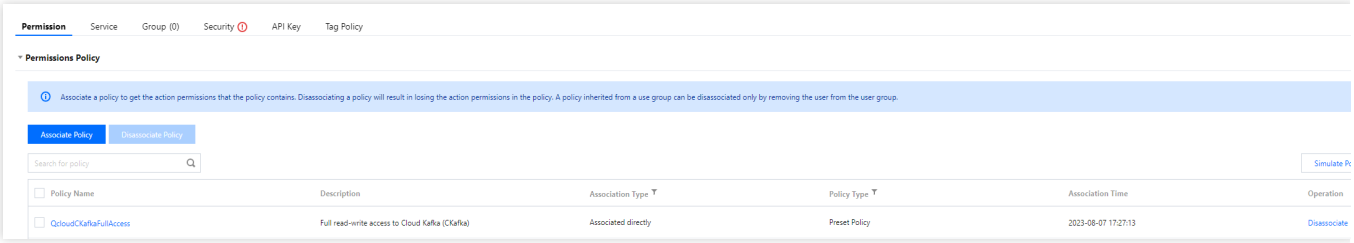
1. Log in to the CAM Console with the root account.
2. In the left sidebar, click **Policies** to go to the policy management page.
3. Search for **QcloudCKafkaFullAccess** on the right.



4. In the search results, click the **Associated Users/Groups** of **QcloudCKafkaFullAccess** and select the sub-account to be authorized.



5. Click **OK** to complete the authorization, which will be displayed in the **Policy List** of the user.

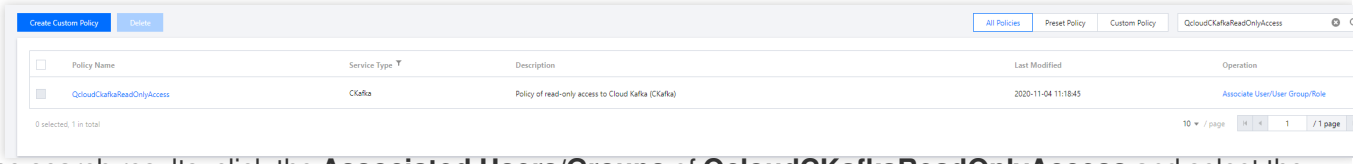


Read-only permission

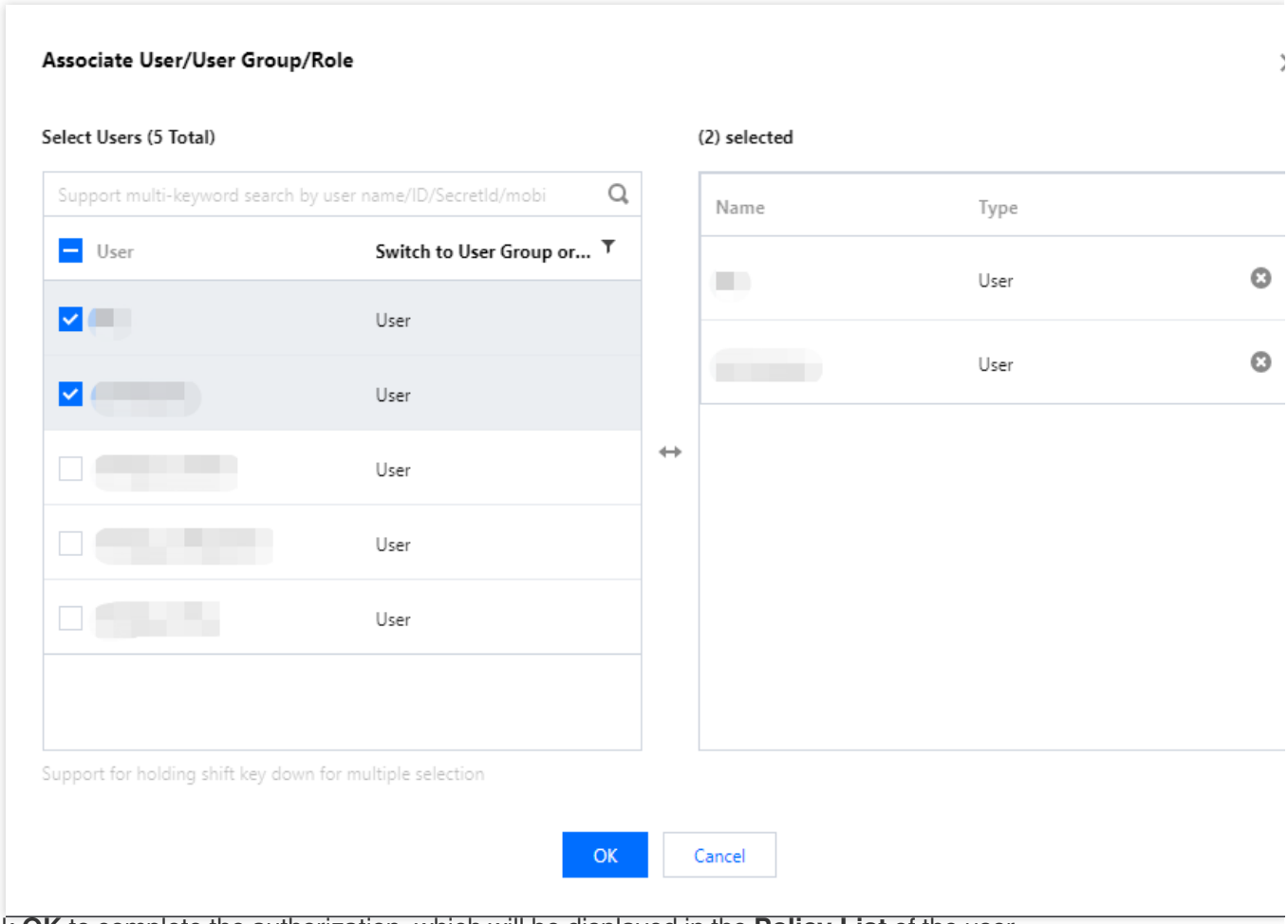
Note:

After granting the read-only permission to a sub-account, the sub-account will have **read-only capability** to **all resources** under the root account.

1. Log in to the CAM Console with the root account.
2. In the left sidebar, click **Policies** to go to the policy management page.
3. Search for **QcloudCKafkaReadOnlyAccess** on the right.

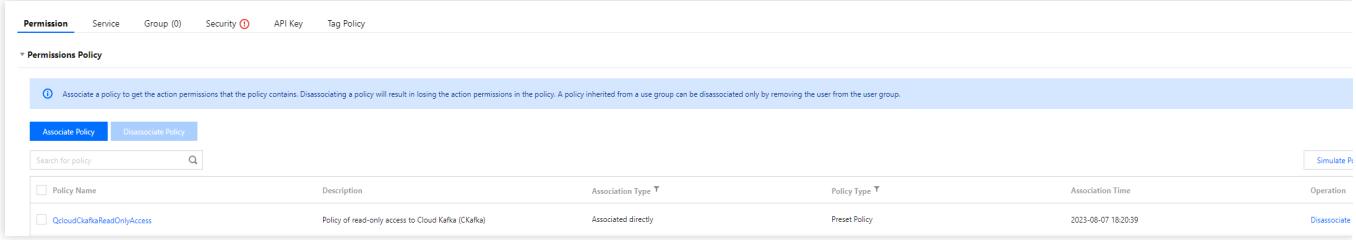


4. In the search results, click the **Associated Users/Groups** of **QcloudCKafkaReadOnlyAccess** and select the sub-account to be authorized.



5. Click **OK** to complete the authorization, which will be displayed in the **Policy List** of the user.

6.



Other methods

Resource-Level Authorization

Tag-Level Authorization

Granting Resource-Level Permissions to Sub-Accounts

Last updated：2024-01-09 14:45:02

Overview

This document describes how to use the root account to authorize sub-accounts at the resource level. After successful authorization, the sub-accounts will have the capability to control a certain resource.

Prerequisites

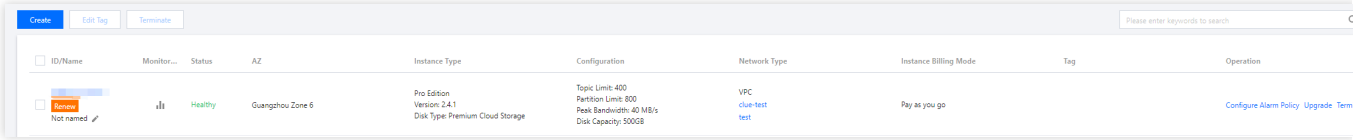
You must have a Tencent Cloud root account and have activated the Cloud Access Management (CAM) service.
Your root account must have at least one sub-account, and you have completed the authorization as instructed in Getting Access Authorization.
You must have at least one CKafka instance.

Directions

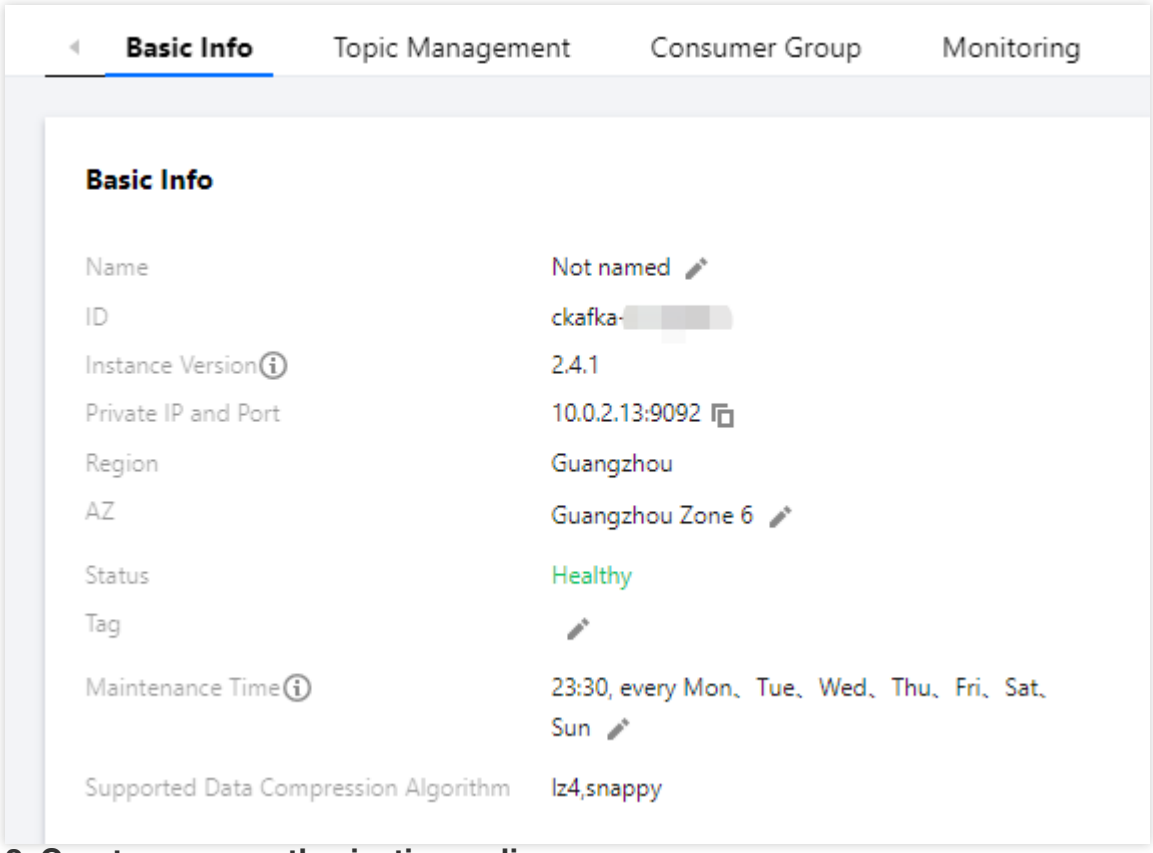
By using the policy feature in the CAM console, you can grant a sub-account access to the CKafka resources owned by the root account. Taking cluster resource as an example, the following describes the detailed steps for **granting the sub-account access to CKafka resources**, which also apply to other types of resources.

Step 1. Obtain the CKafka cluster ID

1. Log in to the CKafka console with **root account**, select an existing cluster instance, and click it to enter the details page.

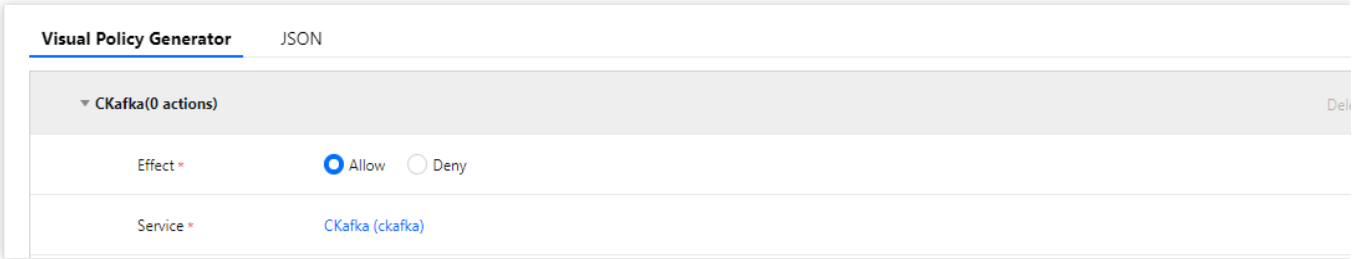


2. In **Basic Info**, the field **ID** indicates the ID of the current CKafka cluster.

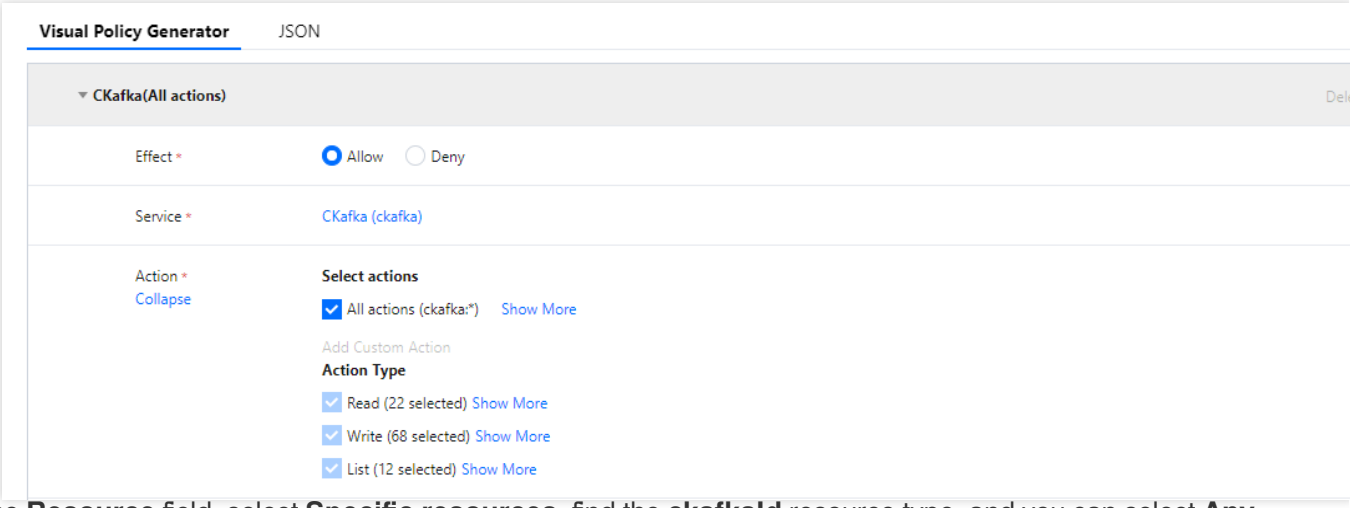


Step 2. Create a new authorization policy

1. Log in to the CAM console and click **Policies** on the left sidebar.
2. Click **Create Custom Policy** > **Create by Policy Generator**.
3. In the visual policy generator, select **Allow** for **Effect**, enter CKafka in **Service** to filter, and select **CKafka (ckafka)**.

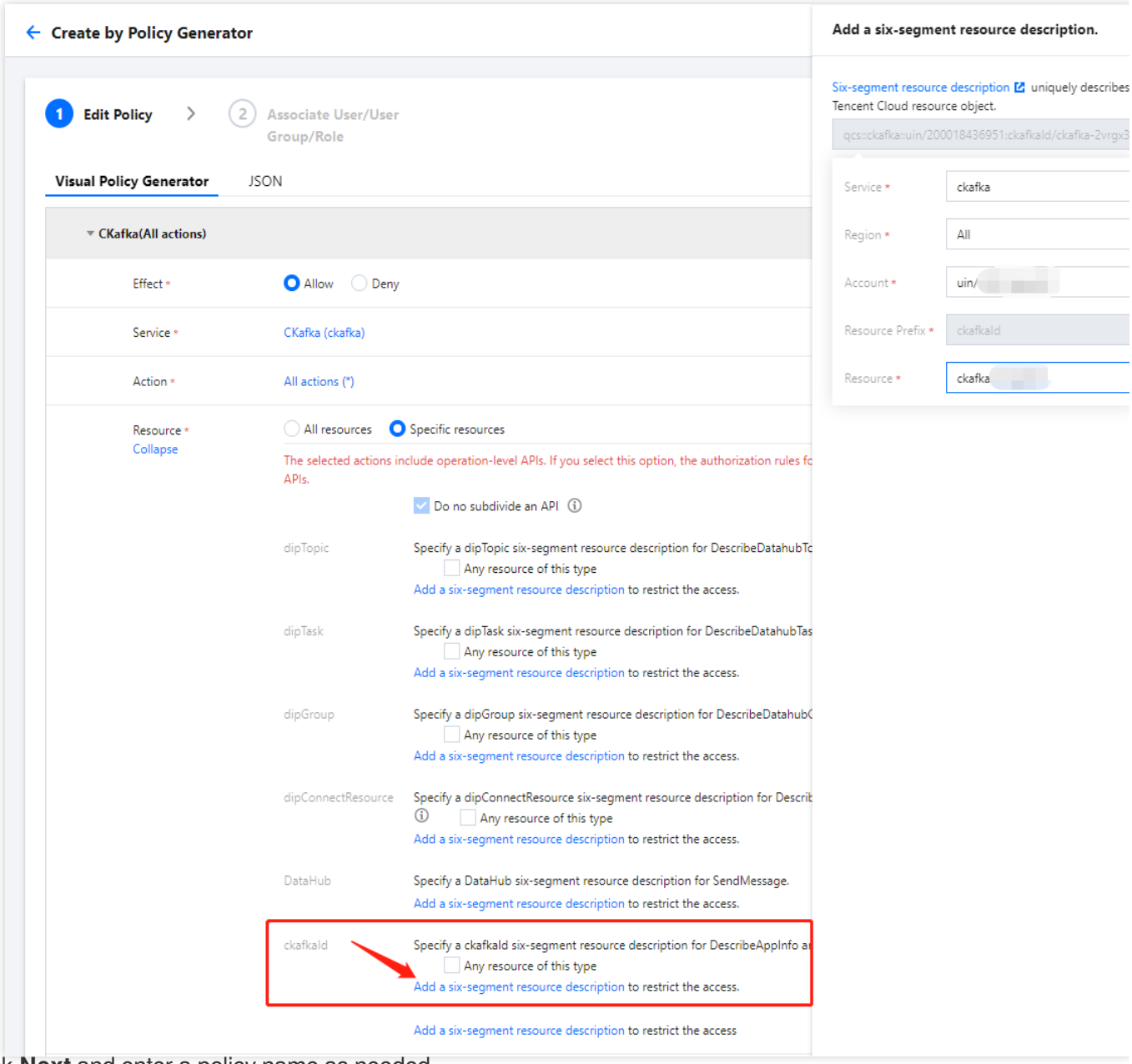


4. Select **All actions** in **Action**, and you can also select the action type as needed.



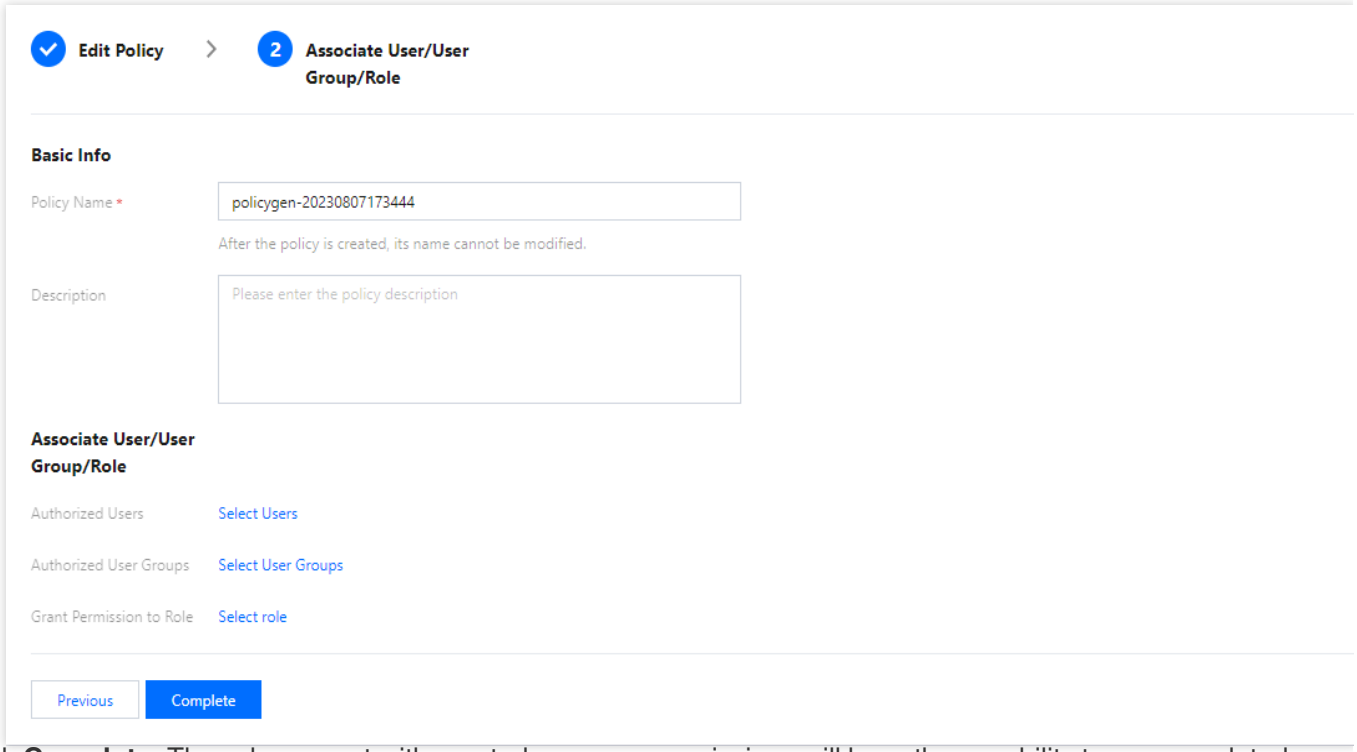
5. In the **Resource** field, select **Specific resources**, find the **ckafkaId** resource type, and you can select **Any resource of this type** on the right to authorize all cluster resources, or click **Add a six-segment resource description** to authorize specific cluster resources.

6. If you click **Add a six-segment resource description**, enter the **cluster ID** for **Resource** in the pop-up dialog box. For how to obtain the cluster ID, see Step 1.



7. Click **Next** and enter a policy name as needed.

8. Click **Select Users** or **Select User Groups** to select the users or user groups that need to be granted resource permissions.



9. Click **Complete**. The sub-account with granted resource permissions will have the capability to access related resources.

Other authorization methods

Operation-Level Authorization

Tag-Level Authorization

Granting Tag-Level Permissions to Sub-Accounts

Last updated：2024-01-09 14:45:02

Overview

This document describes how to use the root account to authorize sub-accounts at the tag level. After successful authorization, the sub-accounts will have the capability to control a certain resource under the authorized tag.

Prerequisites

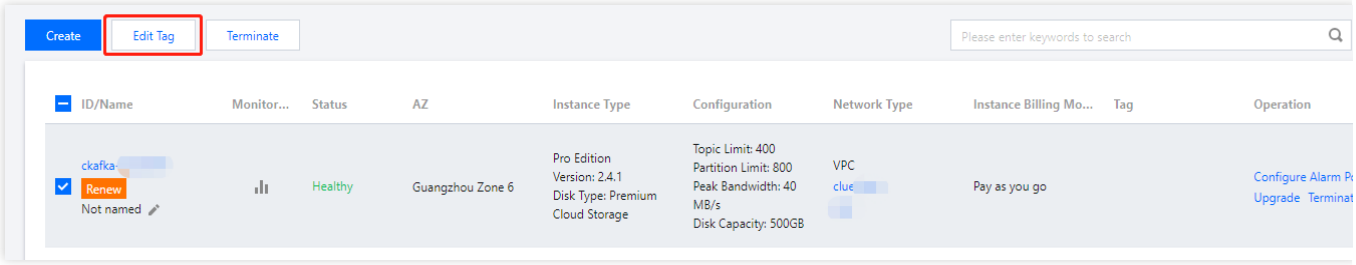
You must have a Tencent Cloud root account and have activated the Cloud Access Management (CAM) service.
Your root account must have at least one sub-account, and you have completed the authorization as instructed in Getting Access Authorization.
You must have at least one CKafka cluster instance.
You must have at least one **tag**, if you don't have one, you can go to the Tag console > **Tag List** to create a new one.

Directions

By using the policy feature in the CAM console, you can grant a sub-account full access to the tagged CKafka resources owned by the root account through the tag authorization. The following describes the detailed steps for **granting the sub-account access to CKafka resources by tag**

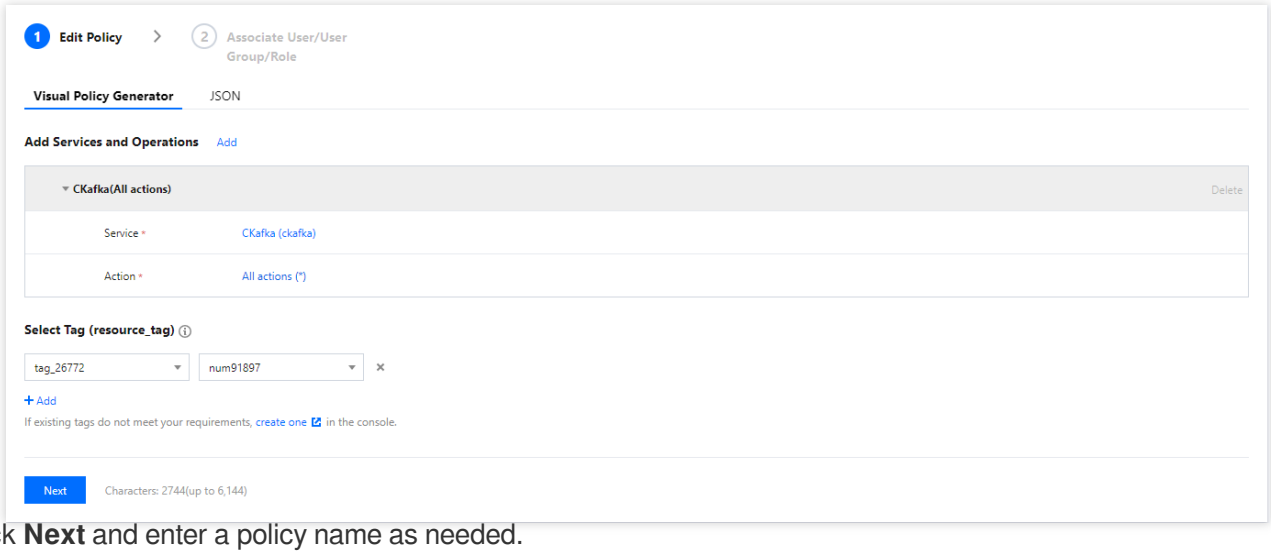
Step 1. Bind tags to resources

1. Log in to the CKafka console with **root account**, and enter the instance list page.
2. Select the target instance, click **Edit Tag** in the upper left corner, and bind the resource tag to the instance.

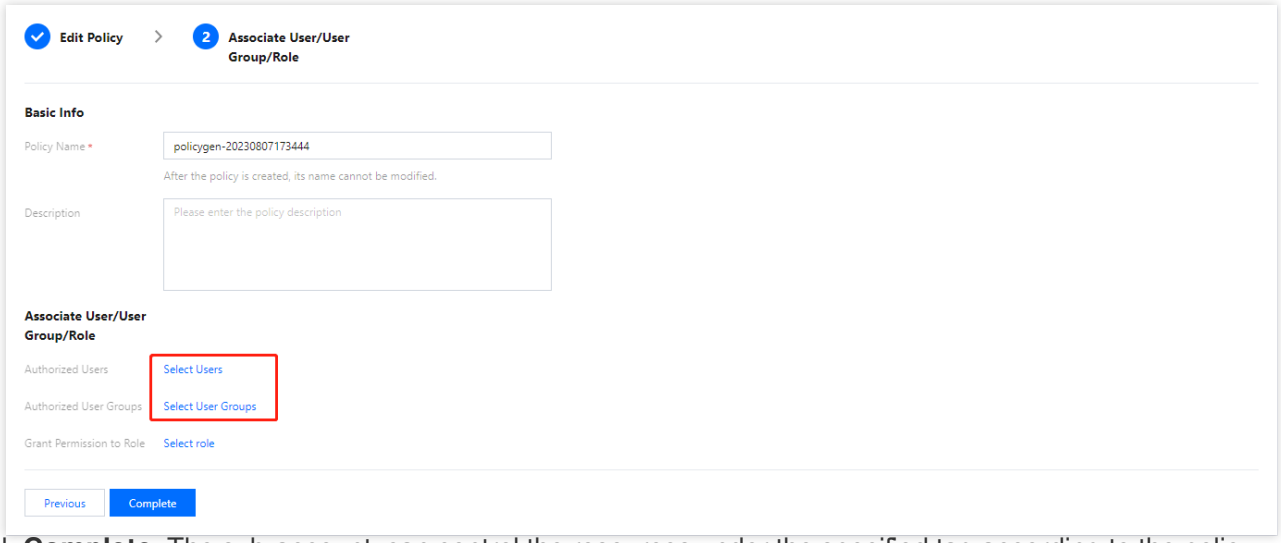


Step 2. Authorize by Tag

1. Log in to the CAM console and click **Policies** on the left sidebar.
2. Click **Create Custom Policy** > **Authorize by Tag**.
3. In the visual policy generator, enter CKafka in **Service** to filter, and select **CKafka (ckafka). Then, select** All actions in **Action**, and you can also select the action type as needed.



4. Click **Next** and enter a policy name as needed.
5. Click **Select Users** or **Select User Groups** to select the users or user groups that need to be granted resource permissions.

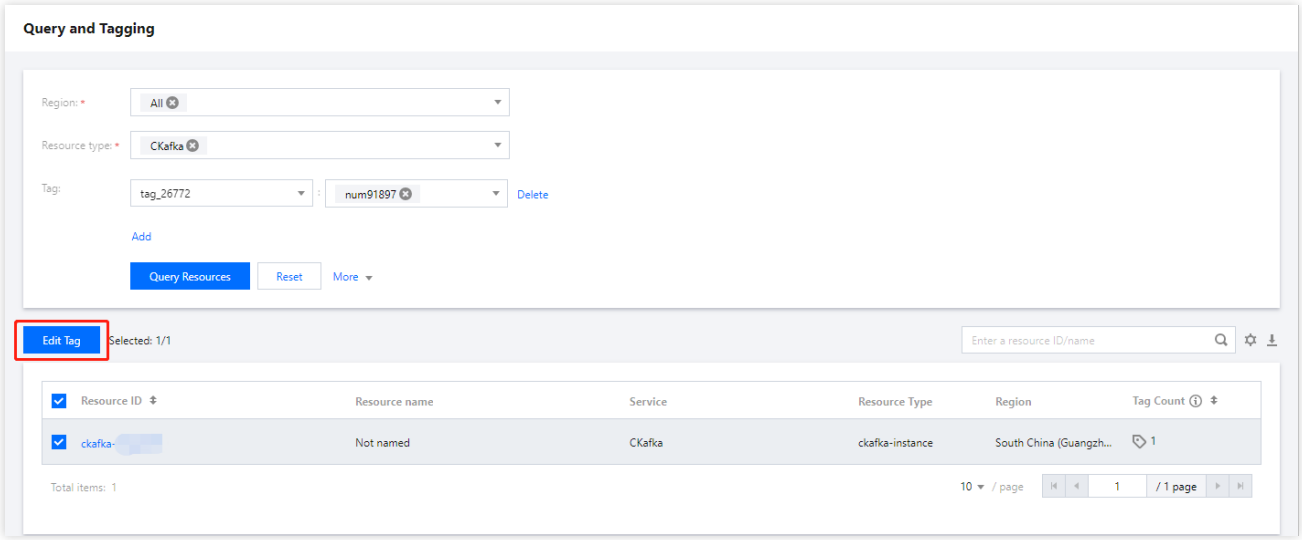


6. Click **Complete**. The sub-account can control the resources under the specified tag according to the policy.

Managing Resource Tags

You can also manage resource tags in a unified manner in the **Tag console**. The detailed operations are as follows.

1. Log in to the **Tag console**.
2. Select **Resource Tag** in the left navigation bar, select query conditions as needed, and select **CKafka** > **ckafka-instance** in **Resource type**.
3. Click **Query Resources**.
4. Select the required resources in the result and click **Edit Tag** to bind or unbind tags in batches.



Other authorization methods

Operation-Level Authorization

Resource-Level Authorization

VPC Access
Step 1. Create an Instance

Last updated：2025-03-26 21:54:14

Overview

This document describes how to create an instance and deploy a VPC in the CKafka console.

Prerequisites

You have signed up for a Tencent Cloud account.
You have created a VPC.

Directions

1. Log in to the CKafka console.
2. Select **Instance List** on the left sidebar, click **Create** to go to the instance purchase page, and enter the purchase information as needed.

Configuration Item	Parameter	Parameter Description
Basic Configuration	Product Form	**Serverful:** The classic form of CKafka. Users can purchase clusters of corresponding specifications based on requirements. As business volume changes, certain attention needs to be paid to the CKafka cluster. **Serverless:** A brand-new form of CKafka. The goal is to completely release the user's effort and focus more on business logic. **Currently in public beta.**
	Billing Mode	Pro Edition instances support two modes: **Monthly Subscription** and **Pay-As-You-Go**. Advanced Edition Instances support **Monthly Subscription** mode. **Monthly Subscription:** Payment is required in advance to use resources. It is mainly suitable for scenarios where the business is relatively stable and used for a long time. **Pay-As-You-Go:** Use resources first and then pay. It is mainly suitable for short-term situations such as testing or when the peak traffic is uncertain.

	Cluster Type	The professional edition is primarily aimed at production environment customers on a large scale. The advanced edition is primarily aimed at test environment customers in small-scale scenarios. For specific differences, refer to Product Specifications. Here you can select **Advanced Edition**.
	Region	Select a region with resources close to those of the client deployment. For regions currently supported by CKafka, see Regions and Availability Zones.
Cluster Configuration	Name	If not filled in, the default is unnamed. When purchasing multiple instances, the system supports creating instance suffix numbers automatically in ascending order and the specify pattern string function. For specific operations, refer to Batch sequential naming or naming with specified pattern strings.
	Kafka version	Choose an appropriate Kafka version based on your business requirement. See Version selection suggestion for CKafka.
	Peak Bandwidth	Estimate the resource amount of peak bandwidth according to the rule of **peak business traffic bandwidth** × **number of replicas**. CKafka will accumulate the bandwidth consumption of all replicas to calculate the actual peak bandwidth.
	Disk	The currently supported disk types are SSD Cloud Block Storage and high-performance cloud block storage. For differences in cloud disk types, see Cloud Disk Type.
	Partition specification	The Partition limit for a CKafka instance is the cumulative total of **number of partitions * number of replicas**. The number of partitions included in the package (i.e., the minimum value) is free of charge. Additional partitions are billed in units of 100. Downgrading is not supported at this time.
	Message retention	Ranges from 24 to 2160 hours. The default message retention time is 72 hours. After exceeding the set retention duration, messages will be deleted to preserve sufficient disk space. CKafka supports the automatic adjustment of disk utilization. After the disk utilization reaches the threshold, you can set the Dynamic Message Retention Policy to reduce message retention time or set the Automatic Disk Capacity Expansion to adjust disk space. For details, see Disk Water Level Processing.
	Cross-AZ Deployment	The professional edition supports deployment in a maximum of 4 different availability zones, and the advanced edition supports deployment in a maximum of 2 different availability zones. For how it works of cross-availability zone deployment, please refer to Cross-AZ Deployment.
Network Configuration	VPC Network	If users need to connect to other private networks, please refer to Add Routing Policy to modify the routing access rules. Select the network created

		in advance here.
Other configuration	Tag	Tags are used to manage resources by category from different dimensions. For method of use, see Tag Management. Leave blank here.
	Automatic Renewal	After checking, when the account balance is sufficient, instances and public network bandwidth will be auto-renewed monthly after expiration.

3. VPC: Select a suitable VPC based on your business needs.

If you want to use other VPCs, follow the steps in Adding Routing Policy to modify the routing rules.

4. Click **Buy Now**. The created instance will be displayed in the instance list in about 3–5 minutes.

Step 2. Create a Topic

Last updated：2025-03-26 21:54:14

Overview

This document describes how to create a topic under an existing instance in the CKafka console.

Directions

1. Log in to the CKafka console.
2. On the **Instance List** page, click the **ID/Name** of the instance created in Step 1. Create an Instance to enter the instance details page.
3. On the instance details page, click **Topic Management** at the top of the page, and click **Create**.
4. In the **Create Topic** dialog box, set parameters as needed.

Parameter	Fill in an Example	Description
Name	Input Topic name	Topic name, cannot be changed after input. The name can only contain letters, numbers, underscores, "-", and ".". Double underscores at the beginning are not supported.
Partition Count	Keep default values for 3 partitions	The concept of physical partition. A Topic can contain one or more partitions. CKafka uses partitions as the allocation unit. The deployment architecture defaults to at least 3 nodes. It is recommended to start with at least 3 partitions for a more balanced data distribution. For parameter configuration instructions on the number of partitions, see Configuration Parameter Description.
Number of Replicas	Keep default values for 2 replicas	The number of replicas of a Partition is used to ensure high availability of the Partition. To ensure data reliability, 2 replicas are enabled by default. The number of replicas is also counted as the number of partitions. For example, if a customer creates 1 Topic, 6 partitions, and 2 replicas, then the total Partition quota used is 1 × 6 × 2 = 12. **Note:** Setting it to a single replica cannot guarantee availability. Proceed with caution.
Tag	Leave blank	Tags are used to manage resources by category from different dimensions. For more details about tags, see Tag Management.

retention.ms	Keep default values for 3 days	Message retention time in the Topic dimension, ranging from 1 minute to 90 days.

5. Click **Submit**.

Step 3. Add a VPC Route

Last updated：2025-03-26 21:54:14

Overview

This document describes how to add a VPC route for a created instance in the CKafka console.

Prerequisites

You have created an instance. For more information, see Step 1. Create an Instance.

Directions

1. On the Instance List page, click the ID/name of the instance created in Step 1. Create an Instance.
2. On the instance details page, click **Add a routing policy** in the **Access Mode** section to add a VPC route.

Then, you can get the domain name and port for VPC access.

Step 4. Send/Receive Messages
Using SDK to Receive/Send Message (Recommended)

Last updated：2024-01-09 14:45:02

Overview

This document describes how to access CKafka to receive/send messages with the SDK for Java in a VPC.
For clients in other languages, see SDK Documentation.

Prerequisites

You have installed JDK 1.8 or later.
You have installed Maven 2.5 or later.
You have downloaded the demo.

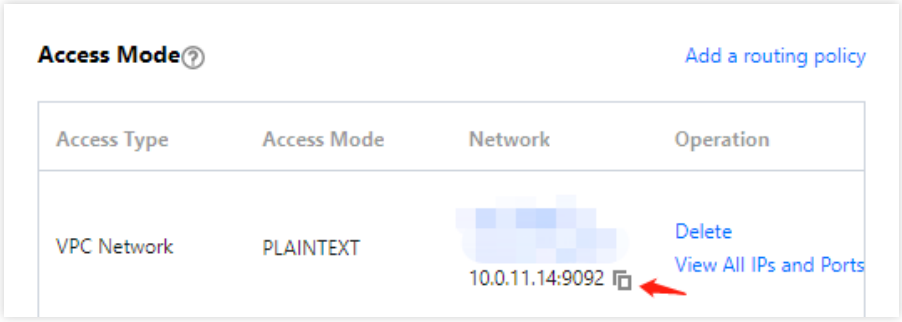
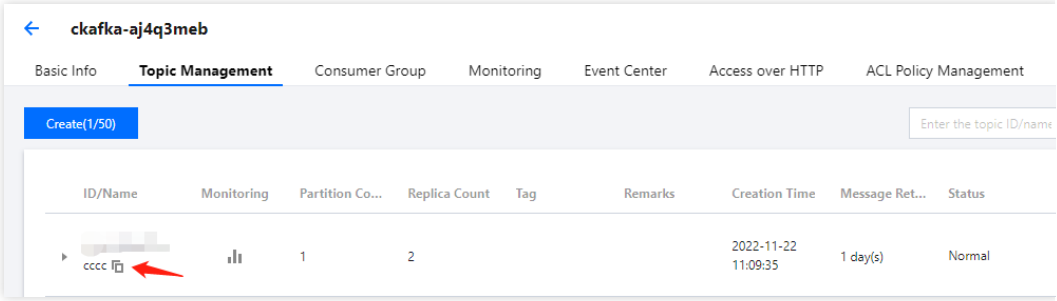
Directions

Step 1. Prepare configurations

1. Upload the downloaded demo to the Linux server under the same VPC, log in to the server, and enter the VPC directory under `javakafkademo` .
2. Modify `kafka.properties` in the `resources` directory under the VPC project.

```
## Configure the accessed network by copying the information in the **Network**
column in the **Access Mode** section on the instance details** page in the
console.
bootstrap.servers=xx.xx.xx.xx:xxxx
## Configure the topic by copying the information on the **Topic Management**
page in the console
topic=XXX
## Configure the consumer group as needed
group.id=XXX
```

Parameter	Description

bootstrap.servers	Access network, which can be copied in the **Network** column in the **Access Mode** section or details page in the console 
topic	Topic name, which can be copied on the **Topic Management** page in the console. 
group.id	You can customize it. After the demo runs successfully, you can see the consumer on the **Con** page.

Step 2. Send messages

1. Compile and run the message production program `CKafkaProducerDemo.java` .

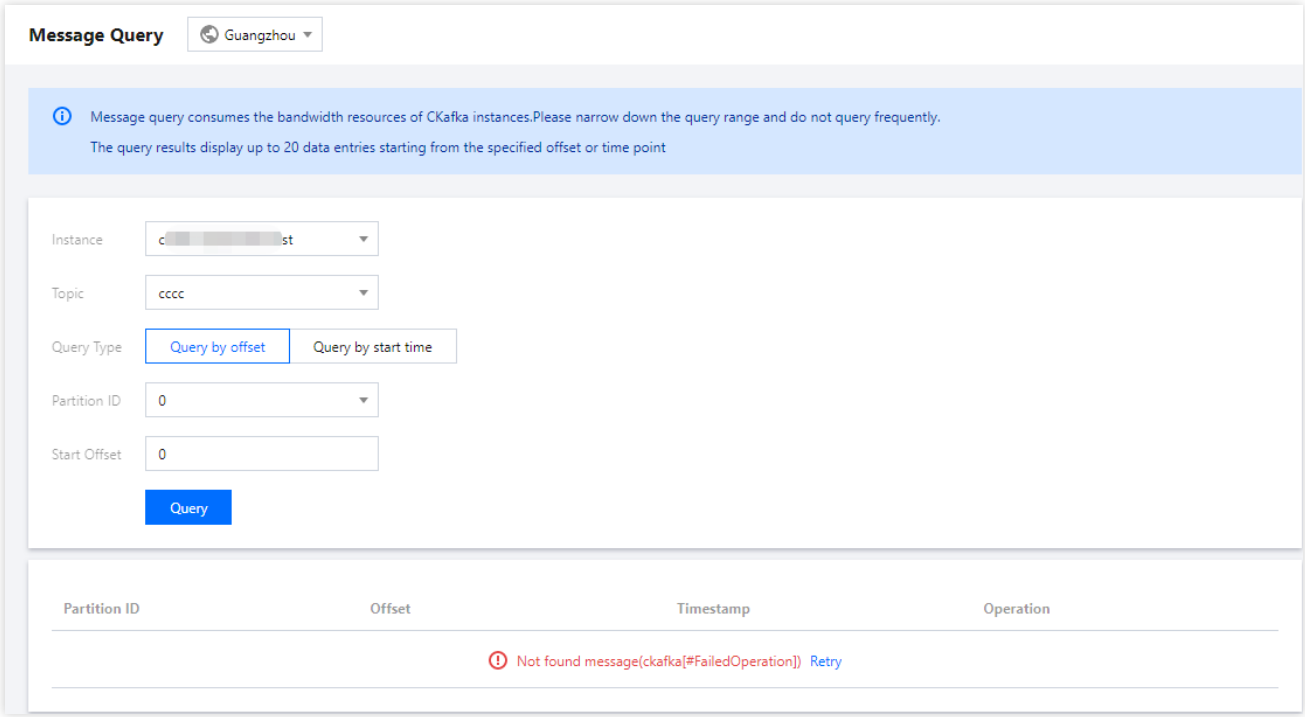
```
public class CKafkaProducerDemo {

    public static void main(String args[]) {
        //Load `kafka.properties`
        Properties kafkaProperties = CKafkaConfigurer.getCKafkaProperties();

        Properties properties = new Properties();
        //Set the access point. Obtain the access point of the corresponding topic
        properties.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaProperties.get

        //Set the method for serializing Kafka messages. `StringSerializer` is used
        properties.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
                "org.apache.kafka.common.serialization.StringSerializer");
```

```
        properties.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
                "org.apache.kafka.common.serialization.StringSerializer");
        //Set the maximum time to wait for a request
        properties.put(ProducerConfig.MAX_BLOCK_MS_CONFIG, 30 * 1000);
        //Set the number of retries for the client
        properties.put(ProducerConfig.RETRIES_CONFIG, 5);
        //Set the interval between retries for the client
        properties.put(ProducerConfig.RECONNECT_BACKOFF_MS_CONFIG, 3000);
        //Construct a producer object
        KafkaProducer<String, String> producer = new KafkaProducer<>(properties);

        //Construct a Kafka message
        String topic = kafkaProperties.getProperty("topic"); //Topic of the message
        String value = "this is ckafka msg value"; //Message content.

        try {
            //Batch obtaining future objects can speed up the process, but the batc
            List<Future<RecordMetadata>> futureList = new ArrayList<>(128);
            for (int i = 0; i < 10; i++) {
                //Send the message and obtain a future object
                ProducerRecord<String, String> kafkaMsg = new ProducerRecord<>(topi
                        value + ": " + i);
                Future<RecordMetadata> metadataFuture = producer.send(kafkaMsg);
                futureList.add(metadataFuture);

            }
            producer.flush();
            for (Future<RecordMetadata> future : futureList) {
                //Sync the future object obtained
                RecordMetadata recordMetadata = future.get();
                System.out.println("produce send ok: " + recordMetadata.toString())
            }
        } catch (Exception e){
            //If the sending still fails after client internal retries, the system
            System.out.println("error occurred");
        }
    }
}
```

2. View the execution result.

```
Produce ok:ckafka-topic-demo-0@198
Produce ok:ckafka-topic-demo-0@199
```

3. Go to the [CKafka console[(https://console.tencentcloud.com/ckafka!85c1cf838df0405887dc01b41e7972fc), select the **Topic Management** tab on the instance details page, select the target topic, and click **More** > **Message Query** to view the message just sent.



Step 3. Consume messages

1. Compile and run the message subscription program `CKafkaConsumerDemo.java` .

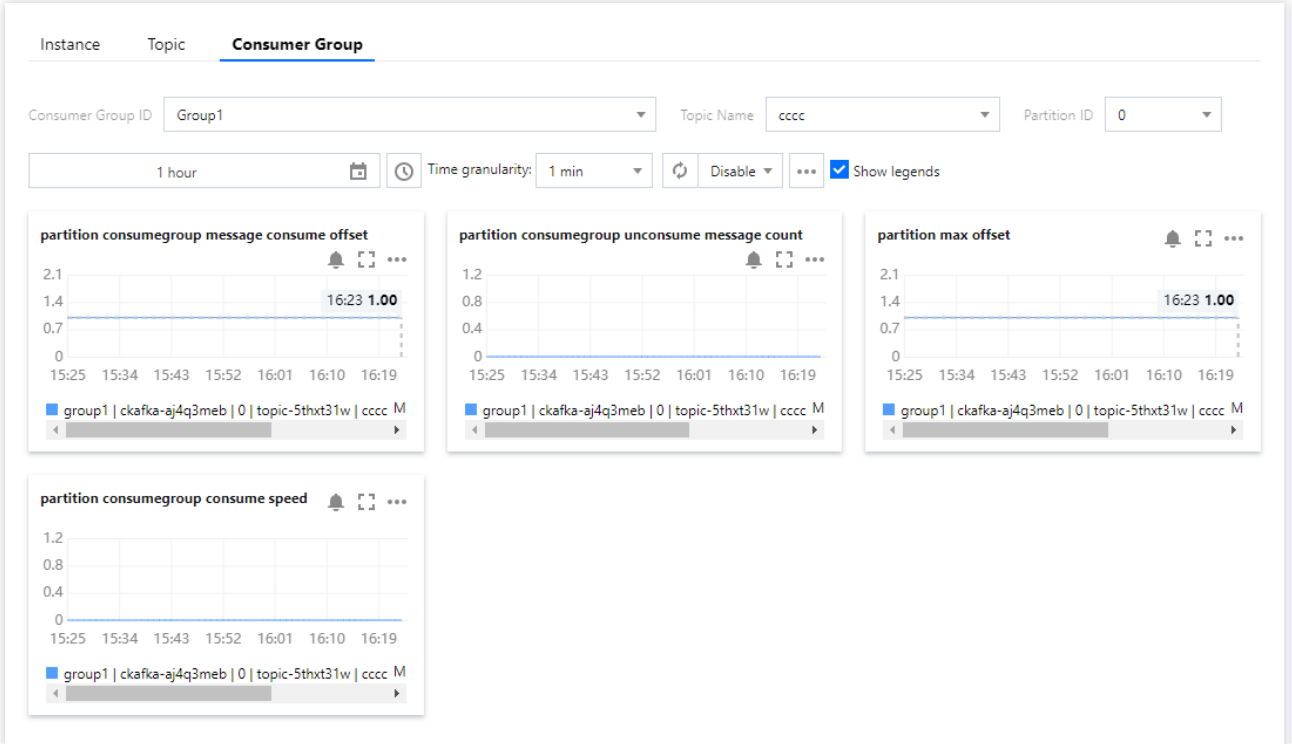
```
public class CKafkaConsumerDemo {

    public static void main(String args[]) {
        //Load `kafka.properties`
        Properties kafkaProperties = CKafkaConfigurer.getCKafkaProperties();

        Properties props = new Properties();
        //Set the access point. Obtain the access point of the topic via the consol
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaProperties.getPrope
        //Set the maximum interval between two polls
        //If the consumer does not return a heartbeat message within the interval,
        props.put(ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG, 30000);
        //Set the maximum number of messages that can be polled at a time
        //Do not set this parameter to an excessively large value. If polled messag
        props.put(ConsumerConfig.MAX_POLL_RECORDS_CONFIG, 30);
        //Set the method for deserializing messages
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
                "org.apache.kafka.common.serialization.StringDeserializer");
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
                "org.apache.kafka.common.serialization.StringDeserializer");
        //The instances in the same consumer group consume messages in load balanci
        props.put(ConsumerConfig.GROUP_ID_CONFIG, kafkaProperties.getProperty("grou
        //Construct a consumer object. This generates a consumer instance
        KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
        //Set one or more topics to which the consumer group subscribes
```

```
        //You are advised to configure consumer instances with the same `GROUP_ID_C
        List<String> subscribedTopics = new ArrayList<>();
        //If you want to subscribe to multiple topics, add the topics here
        //You must create the topics in the console in advance.
        String topicStr = kafkaProperties.getProperty("topic");
        String[] topics = topicStr.split(",");
        for (String topic : topics) {
            subscribedTopics.add(topic.trim());
        }
        consumer.subscribe(subscribedTopics);

        //Consume messages in loop
        while (true){
            try {
                ConsumerRecords<String, String> records = consumer.poll(1000);
                //All messages must be consumed before the next poll, and the total
                //You are advised to create a separate thread to consume messages a
                for (ConsumerRecord<String, String> record : records) {
                    System.out.println(
                            String.format("Consume partition:%d offset:%d", record.
                }
            } catch (Exception e){
                System.out.println("consumer error!");
            }
        }
    }
}
```

2. View the execution result.

```
Consume partition:0 offset:298
Consume partition:0 offset:299
```

3. On the **Consumer Group** tab page in the CKafka console, select the corresponding consumer group name, enter the topic name, and click **View Details** to view the consumption details.


Instance
Topic
Consumer Group
Consumer Group ID
Group1
Topic Name
cccc
Partition ID
0
1 hour
Time granularity:
1 min
Disable
Show legends
partition consumegroup message consume offset
2.1
1.4
0.7
0
16:23 1.00
15:25 15:34 15:43 15:52 16:01 16:10 16:19
group1 | ckafka-aj4q3meb | 0 | topic-5thxt31w | cccc M
partition consumegroup unconsume message count
1.2
0.8
0.4
0
15:25 15:34 15:43 15:52 16:01 16:10 16:19
group1 | ckafka-aj4q3meb | 0 | topic-5thxt31w | cccc M
partition max offset
2.1
1.4
0.7
0
16:23 1.00
15:25 15:34 15:43 15:52 16:01 16:10 16:19
group1 | ckafka-aj4q3meb | 0 | topic-5thxt31w | cccc M
partition consumegroup consume speed
1.2
0.8
0.4
0
15:25 15:34 15:43 15:52 16:01 16:10 16:19
group1 | ckafka-aj4q3meb | 0 | topic-5thxt31w | cccc M

Running Kafka Client (Optional)

Last updated：2024-01-09 14:45:02

Overview

This document explains how to start using Kafka APIs after you purchase the CKafka service. After setting up a CKafka environment on a CVM instance, you need to download and decompress the Kafka installation file and perform simple testing on Kafka APIs.

Directions

Step 1. Install a JDK.

1. Check Java installation.

Open a terminal window and run this command:

```
java -version
```

If the output of the command is a Java version number, then Java is already installed in your system. If you have not installed Java yet, download and install a Java Development Kit (JDK).

2. Set up the Java environment.

Set the `JAVA_HOME` environment variable and point it to the Java installation directory on your machine.
For example, if you use Java JDK 1.8.0_20, the outputs on different operating systems are as follows:

Supported Operating Systems	Output
Windows	Set the environment variable JAVA_HOME to C:\\Program Files\\Java\\jdkjdk1.8.0_20
Linux	export JAVA_HOME=/usr/local/java-current
Mac OSX	export JAVA_HOME=/Library/Java/Home

Add the Java compiler path to the system path:

Supported Operating Systems	Output

Windows	Add `;C:\\Program Files\\Java\\jdk1.8.0_20\\bin` to the end of the system variable `Path`
Linux	export PATH=$PATH:$JAVA_HOME/bin/
Mac OSX	not required

Use the `java -version` command to check your Java installation.

Step 2. Download the Kafka installation file.

Download and decompress the Kafka installation file.

Step 3. Test Kafka APIs.

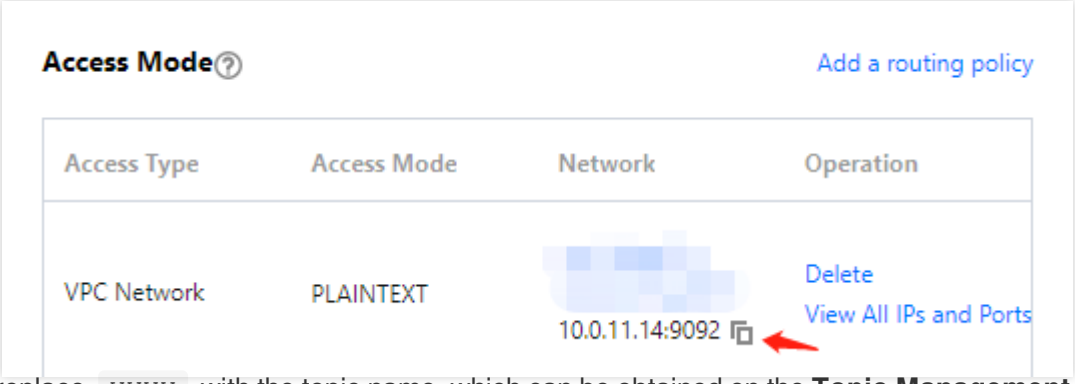
Go to the `./bin` directory, and produce and consume a message via CLI commands.

1. Open a terminal window to start a consumer.

```
bash kafka-console-consumer.sh --bootstrap-server XXXX:port --topic XXXX --
consumer.config ../config/consumer.properties
```

Note:

Replace `XXXX:port` with the domain name and port for VPC access, which can be obtained in the **Access Mode** section on the **Instance Details** page in the console.



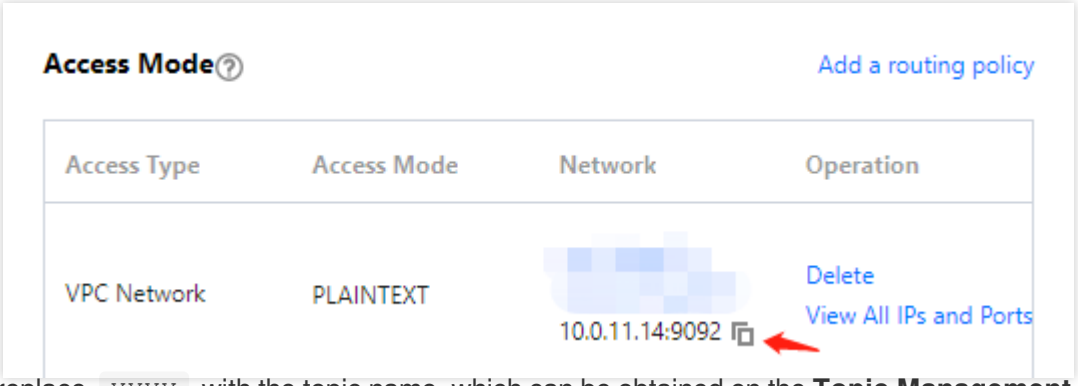
topic: replace `XXXX` with the topic name, which can be obtained on the **Topic Management** page in the console.

2. Open another terminal window to start a producer.

```
bash kafka-console-producer.sh --broker-list XXXX:port --topic XXXX --
producer.config ../config/producer.properties
```

Note:

Replace `XXXX:port` with the domain name and port for VPC access, which can be obtained in the **Access Mode** section on the **Instance Details** page in the console.



topic: replace `XXXX` with the topic name, which can be obtained on the **Topic Management** page in the console. Enter the content of the message and press Enter.

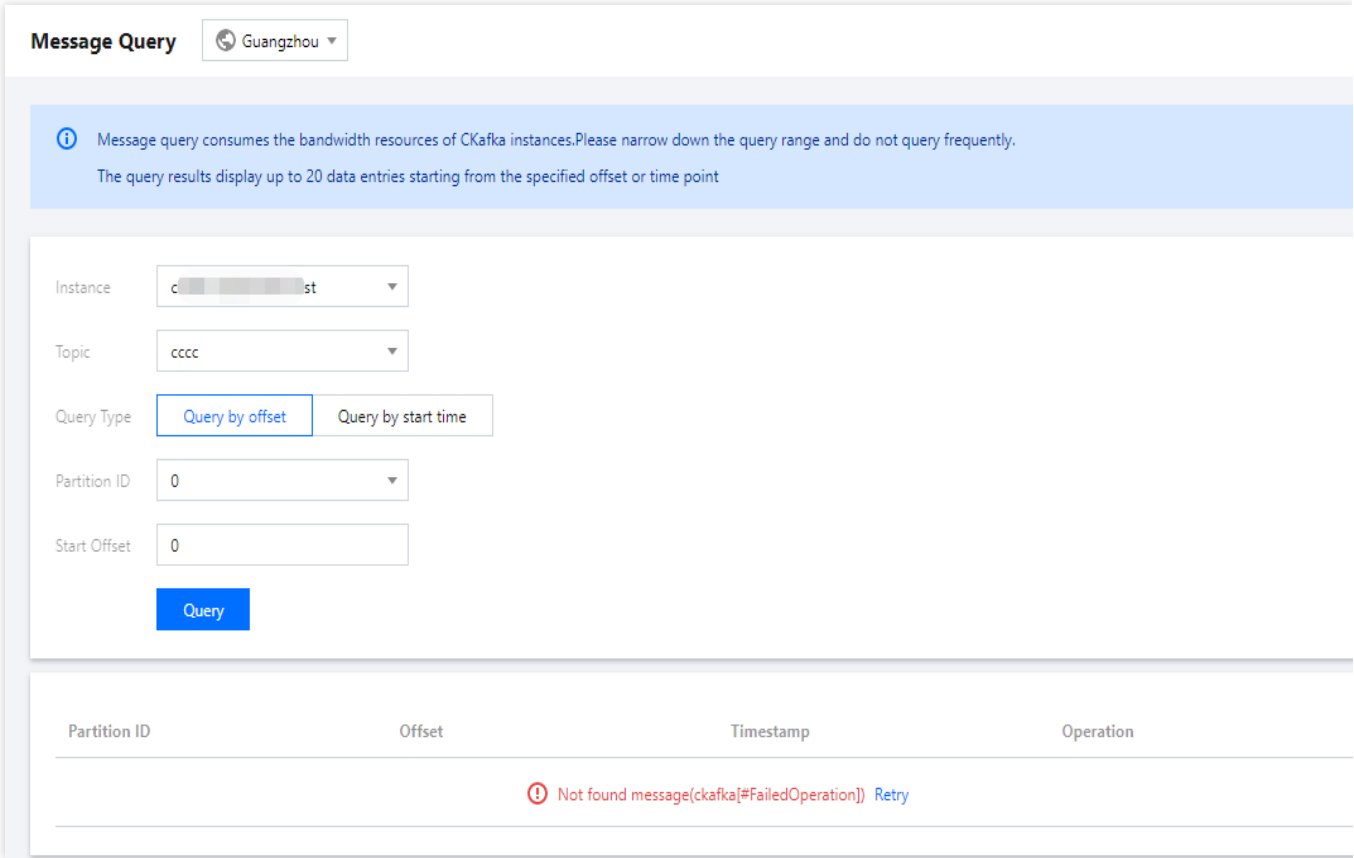
Producing a message:

```
[ bin % bash kafka-console-producer.sh --broker-list ckafka
.ap-guangzhou.ckafka.tencentcloudmq.com:6014 --topic test --producer.co
nfig ../config/producer.properties
>hello world
>this is a message
>this is another message
>
```

Consuming a message:

```
[ bin % bash kafka-console-consumer.sh --bootstrap-server c
kafka- .ap-guangzhou.ckafka.tencentcloudmq.com:6014 --topic test --consum
er.config ../config/consumer.properties
hello world
this is a message
this is another message
```

3. In the message querying page of the CKafka console, query the message sent.



The details of the message are as follows: