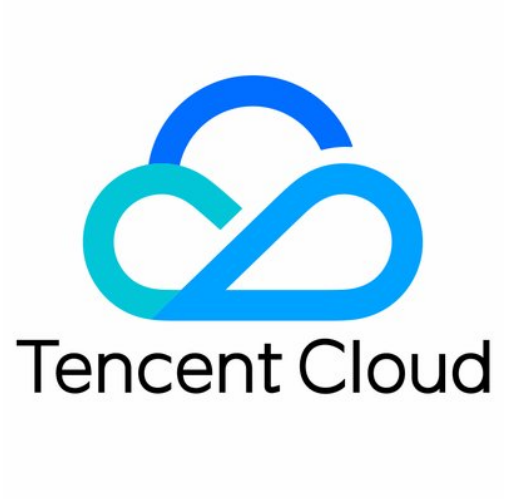
TDMQ for RocketMQ

Getting Started

Product Documentation



Contents

Getting Started
Sending/Receiving Messages over TCP
Resource Creation and Preparation

Last updated：2022-07-04 16:38:20

Overview

This document describes how to create resources such as cluster and topic in the TDMQ console and what operations you need to perform in the console before running a client.

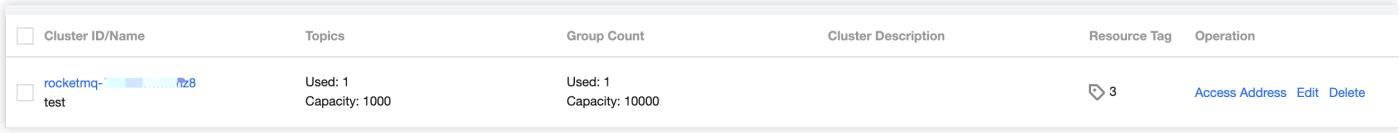
Prerequisites

- You have signed up for a Tencent Cloud account.

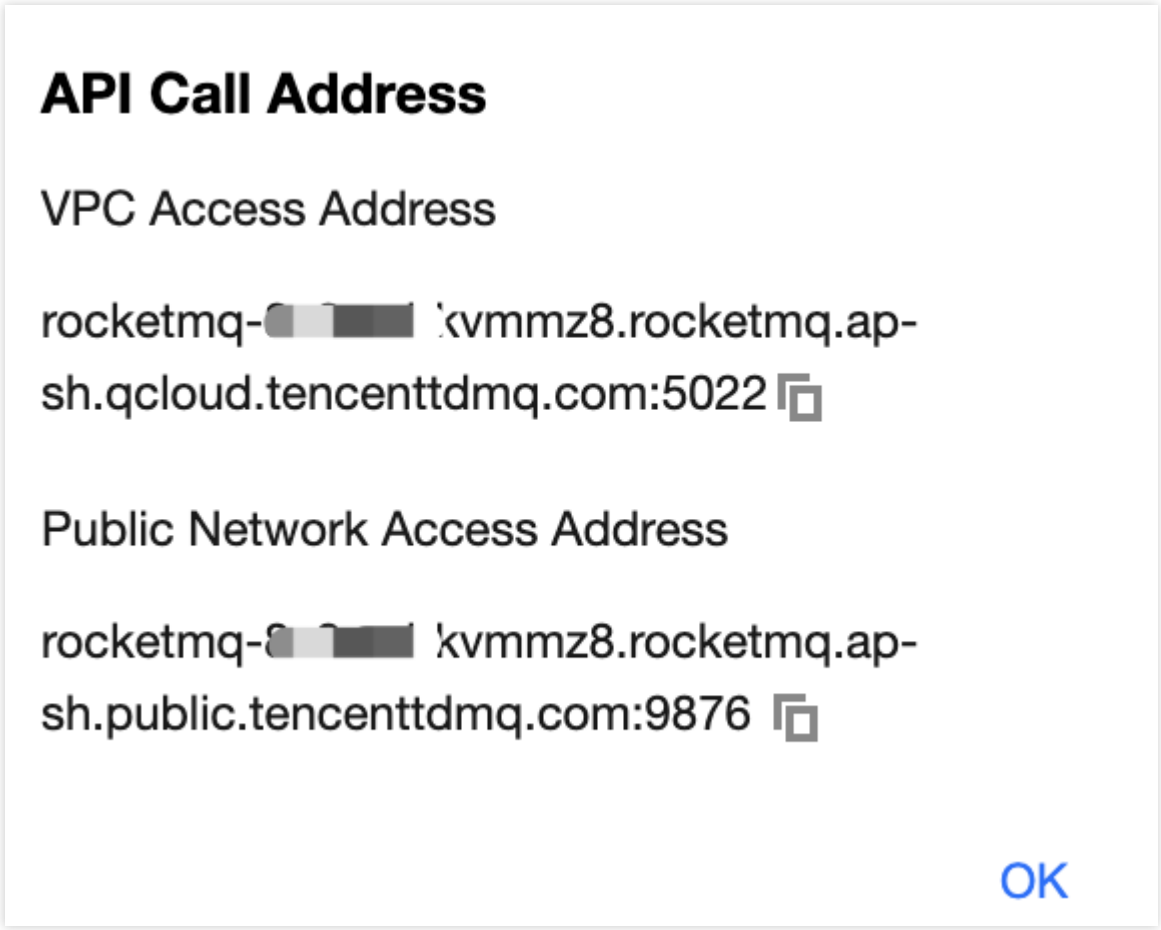
Directions

Step 1. Create a cluster

1. Log in to the TDMQ console, enter the **Cluster** page, and select the target region.
2. Click **Create Cluster** and enter the cluster name and description to create a cluster.

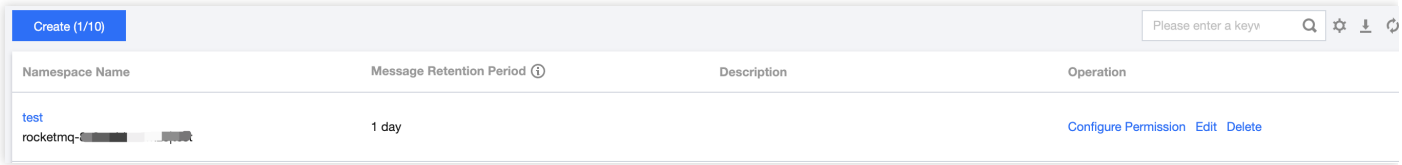


3. On the **Cluster** list page, click **Access Address** in the **Operation** column of the cluster you just created to get the connection information of the server.



Step 2. Create a namespace

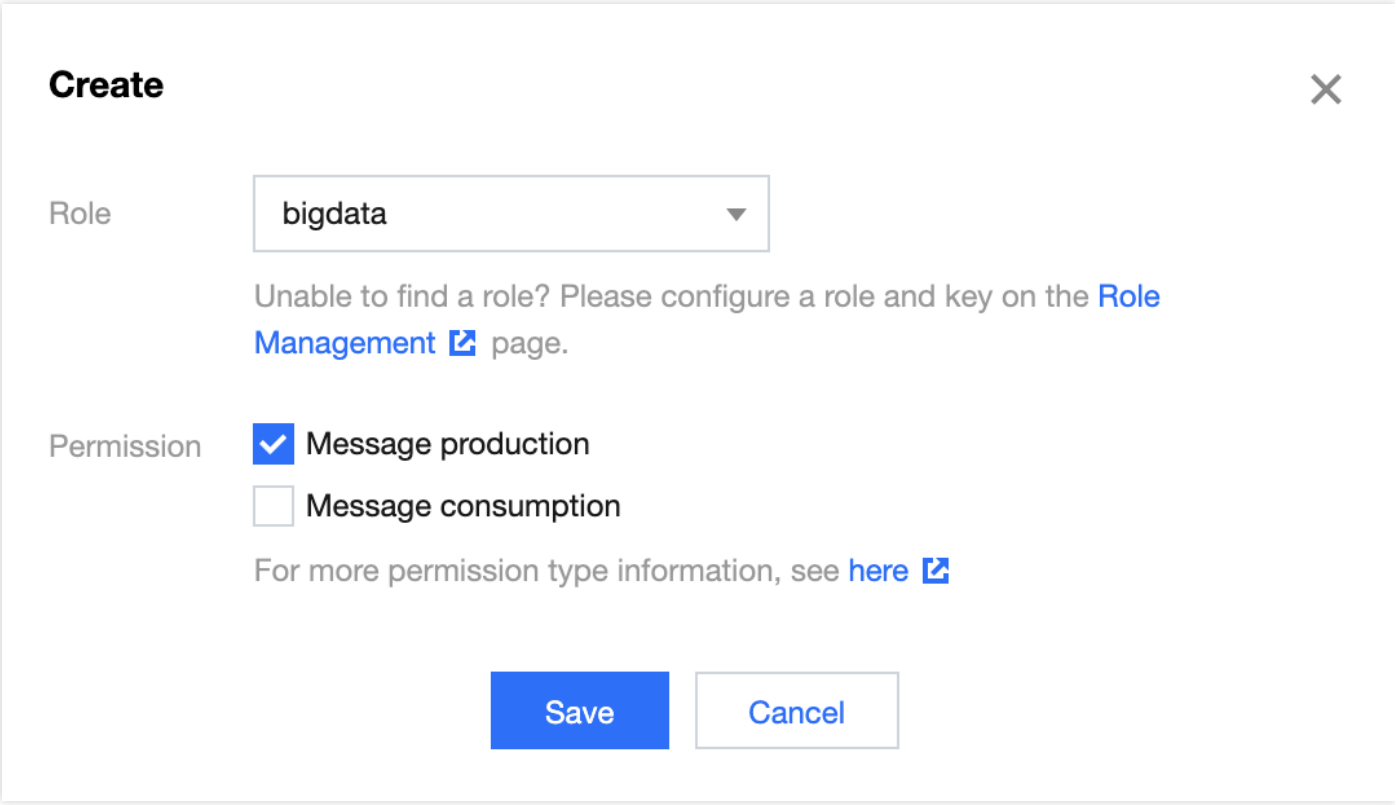
1. On the **Cluster** list page, click the ID of the cluster created in **step 1** to enter the cluster's basic information page.
2. Select the **Namespace** tab at the top, click **Create**, and set the namespace name, message retention period, and description to create a namespace.



Step 3. Create a role and configure permissions

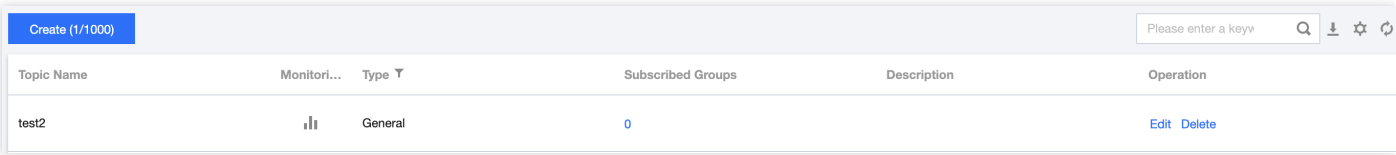
1. Select **Role Management** on the left sidebar and click **Create** to create a role.
2. On the **Cluster** page, click the ID of the cluster you just created to enter the cluster details page.
3. Select the **Namespace** tab at the top and click **Configure Permissions** in the **Operation** column of the namespace you just created.

4. On the **Configure Permission** page, click **Add Role** to add production and consumption permissions to the role you just created.



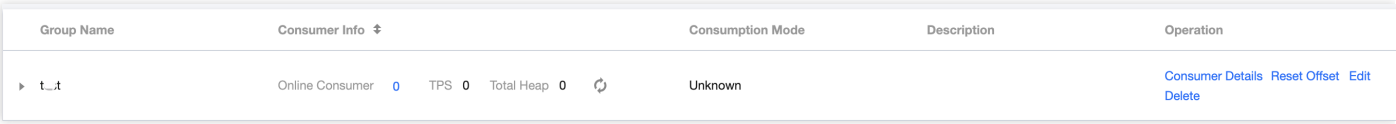
Step 4. Create a topic

1. On the **Namespace** list page, select the **Topic** tab at the top to enter the **Topic** list page.
2. Select the namespace created in step 3 and click **Create** to create a topic.



Step 5. Create a group

1. On the **Topic** list page, select the **Group** tab at the top to enter the **Group** list page.
2. Select the namespace you just created and click **Create** to create a group.



Downloading and Running Demo

Last updated：2023-03-07 16:29:05

Overview

This document describes how to use open-source SDK to send and receive messages by using the SDK for Java as an example and helps you better understand the message sending and receiving processes.

Notes

The following takes the Java client as an example. For clients in other languages, see SDK Documentation.

Prerequisites

You have created the required resources as instructed in Resource Creation and Preparation.

You have installed JDK 1.8 or later.

You have installed Maven 2.5 or later.

You have downloaded the demo.

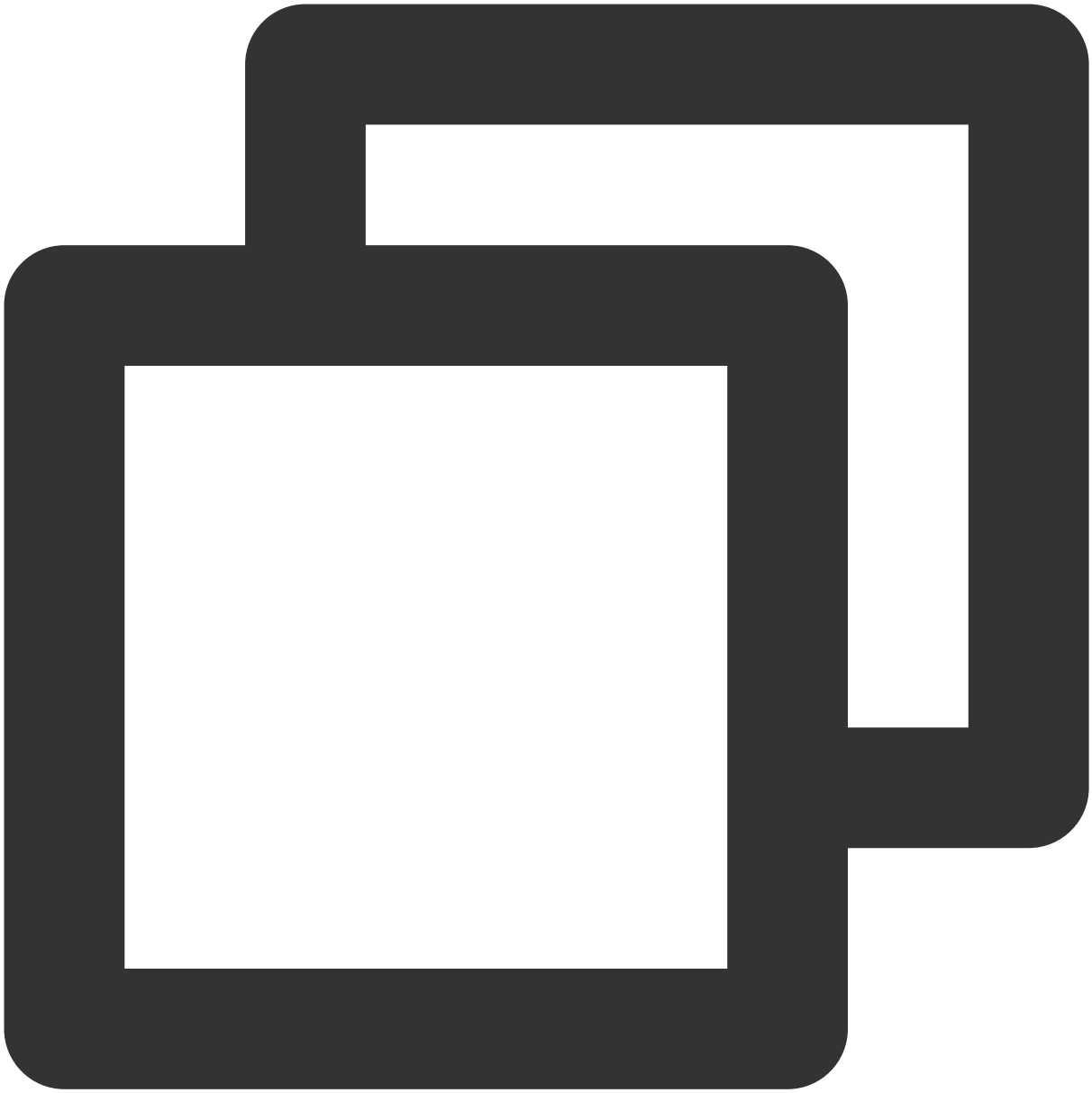
Directions

Step 1. Install the Java dependency library

Introduce dependencies in a Java project and add the following dependencies to the `pom.xml` file. This document uses a Maven project as an example.

Notes

The dependency version must be v4.9.3 or later.



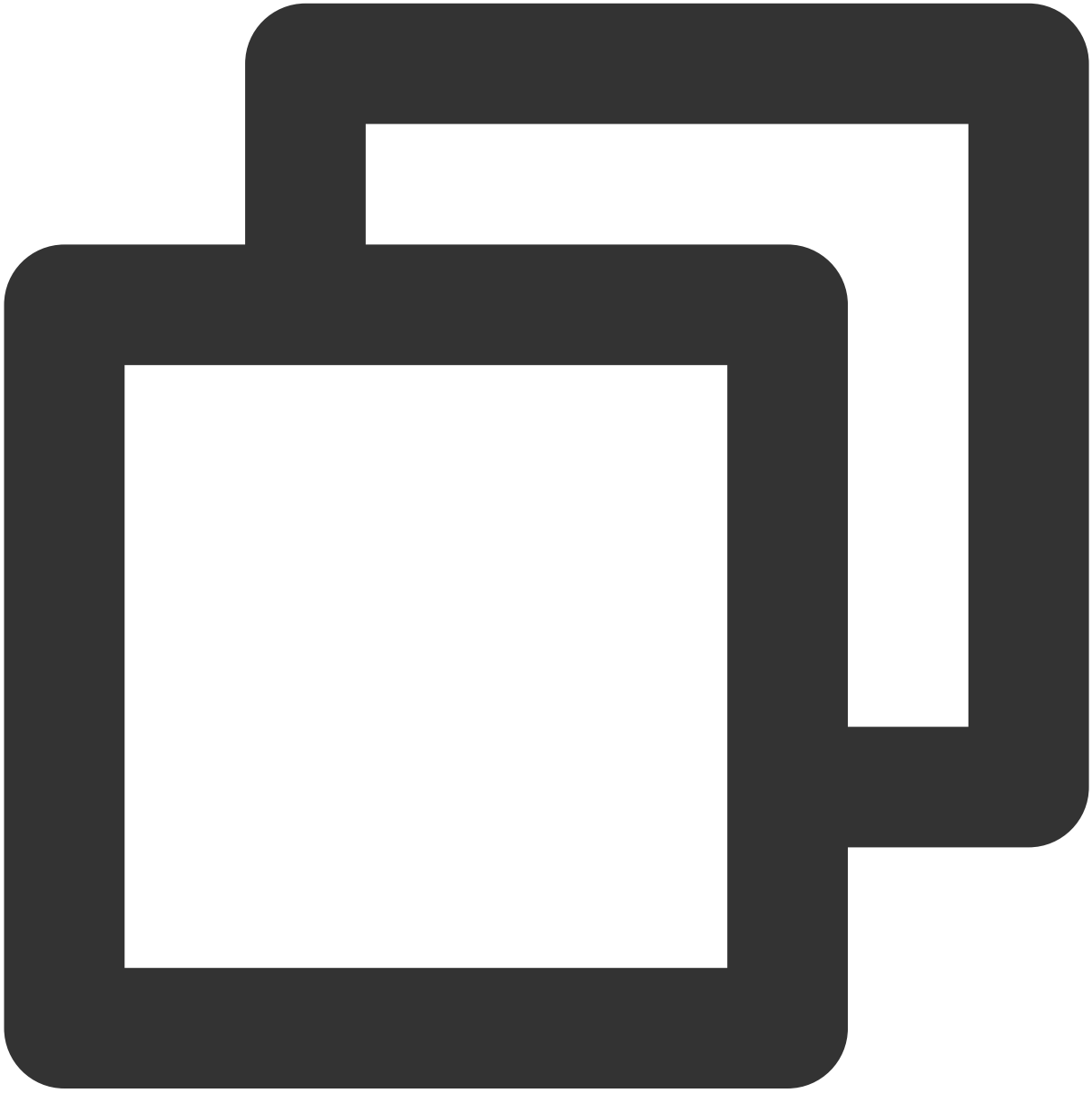
```
<!-- in your <dependencies> block -->
<dependency>
     <groupId>org.apache.rocketmq</groupId>
     <artifactId>rocketmq-client</artifactId>
     <version>4.6.1</version>
</dependency>

<dependency>
     <groupId>org.apache.rocketmq</groupId>
     <artifactId>rocketmq-acl</artifactId>
     <version>4.6.1</version>
```

```
</dependency>
```

Step 2. Produce messages

1. Create a message producer



```
// Instantiate the message producer
DefaultMQProducer producer = new DefaultMQProducer(
     namespace,
     groupName,
     new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)) // ACL perm
```

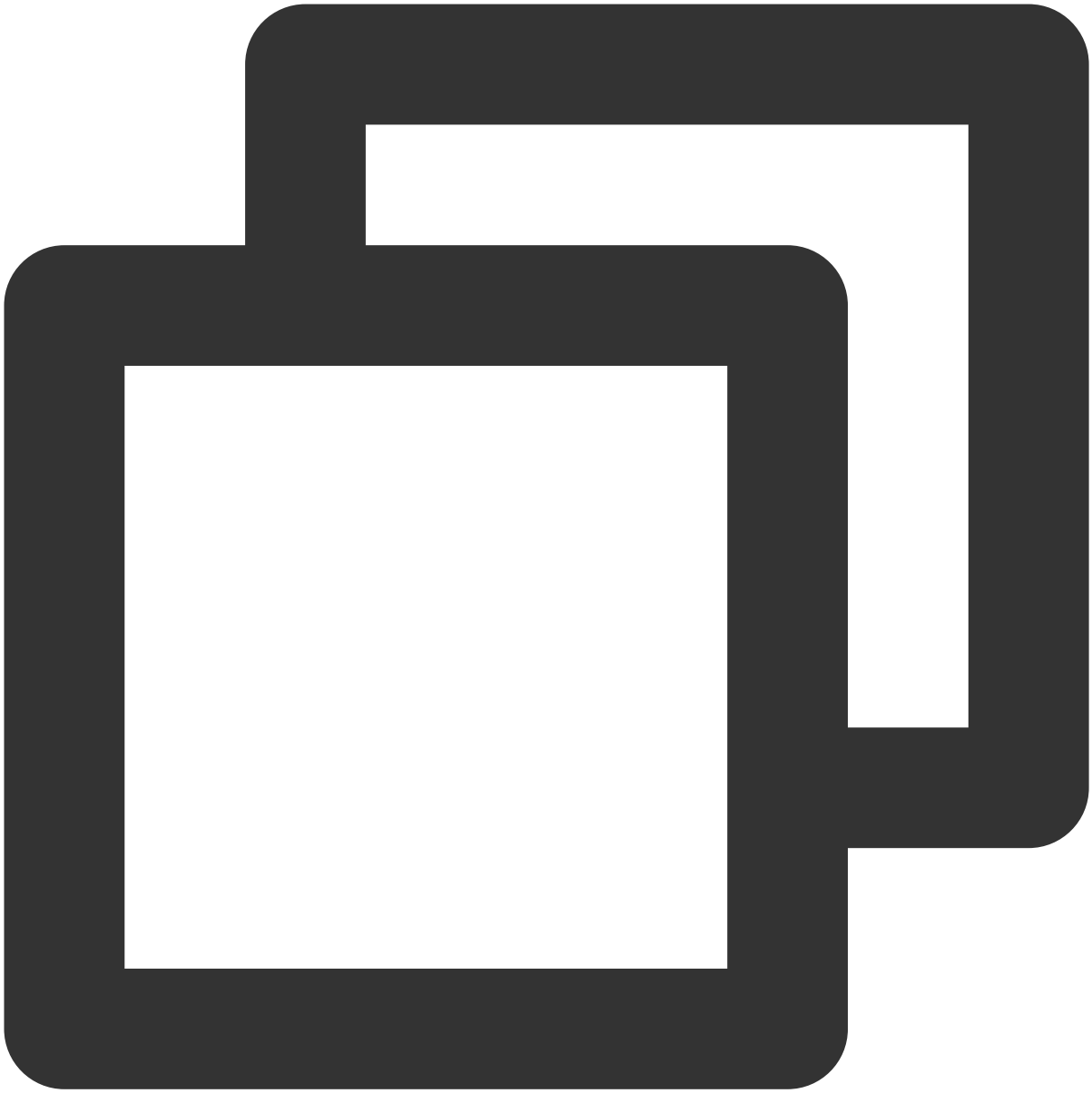
```
);
// Set the NameServer address
producer.setNamesrvAddr(nameserver);
// Start the producer instances
producer.start();
```

Parameter	Description
namespace	Namespace name, which can be copied under the **Namespace** tab in the console. Its format is clus
groupName	Producer group name, which can be copied under the **Group** tab on the **Cluster** page in the consol
nameserver	Cluster access address, which can be obtained from **Access Address** in the **Operation** column or console.
secretKey	Role name, which can be copied on the Role Management page.
accessKey	Role token, which can be copied in the **Token** column on the Role Management page. 

2. Send a message

Messages can be sent in the sync, async, or one-way mode.

Sync sending

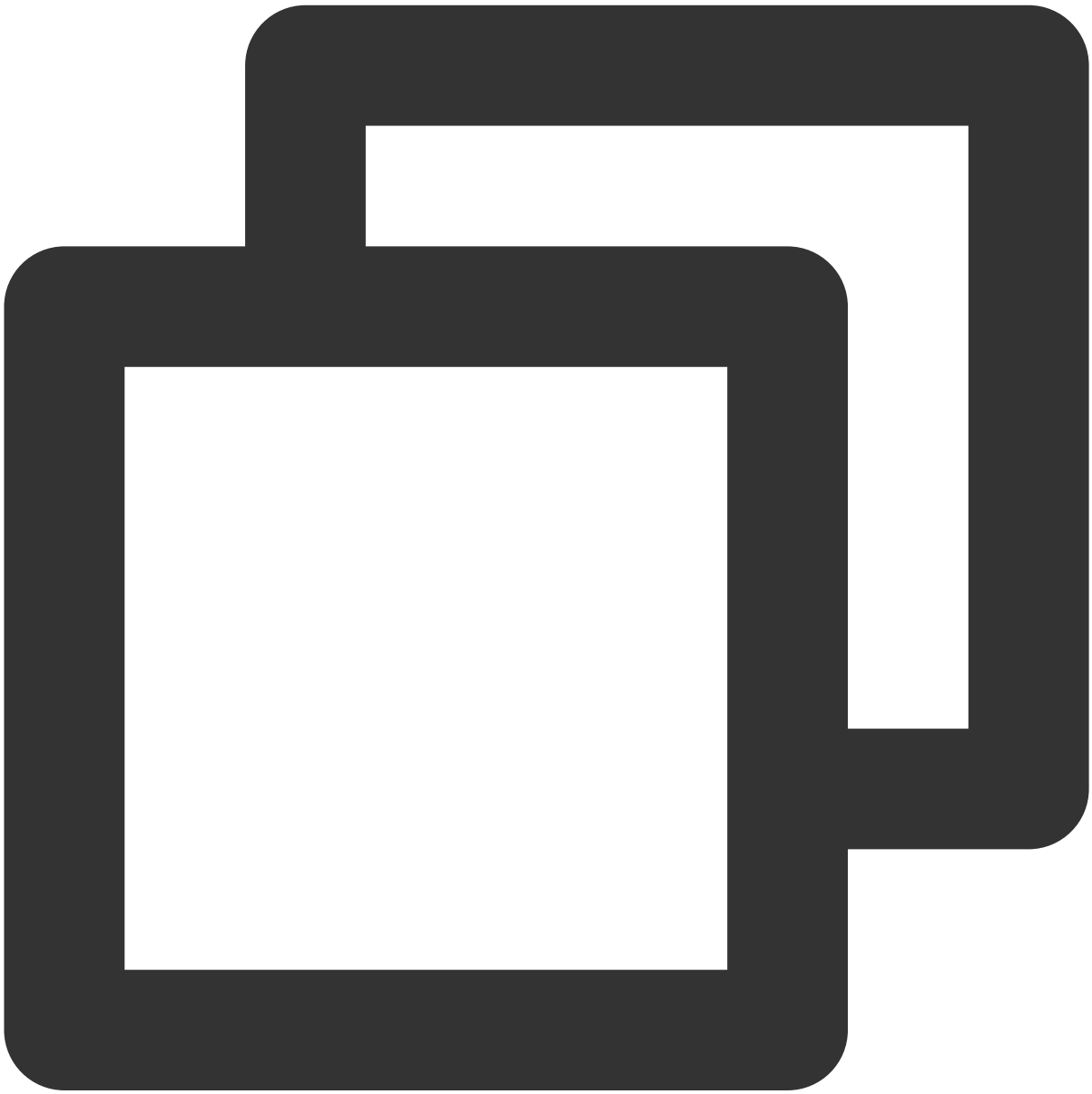


```
for (int i = 0; i < 10; i++) {
     // Create a message instance and set the topic and message content
     Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " + i).getBytes(
     // Send the message
     SendResult sendResult = producer.send(msg);
     System.out.printf("%s%n", sendResult);
}
```

Parameter	Description

topic_name	Topic name, which can be copied under the **Topic** tab on the **Cluster** page in the conso
TAG	A parameter used to set the message tag.

Async sending



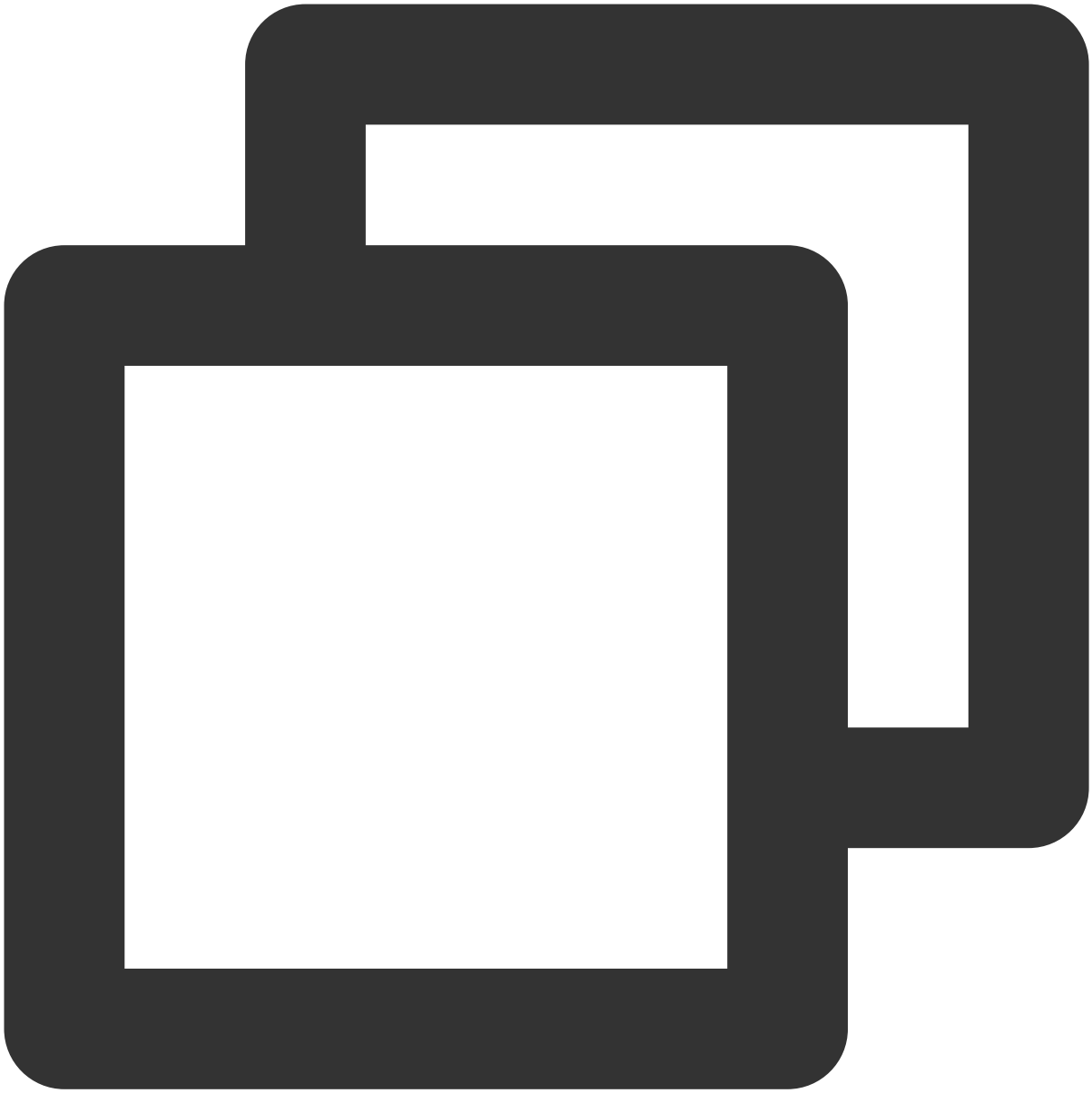
```
// Disable retry upon sending failures
producer.setRetryTimesWhenSendAsyncFailed(0);
// Set the number of messages to be sent
int messageCount = 10;
final CountDownLatch countDownLatch = new CountDownLatch(messageCount);
```

```
for (int i = 0; i < messageCount; i++) {
     try {
             final int index = i;
             // Create a message instance and set the topic and message content
             Message msg = new Message(topic_name, "TAG", ("Hello rocketMq " + inde
             producer.send(msg, new SendCallback() {
                     @Override
                     public void onSuccess(SendResult sendResult) {
                             // Logic for message sending successes
                             countDownLatch.countDown();
                             System.out.printf("%-10d OK %s %n", index, sendResult.
                     }

                     @Override
                     public void onException(Throwable e) {
                             // Logic for message sending failures
                             countDownLatch.countDown();
                             System.out.printf("%-10d Exception %s %n", index, e);
                             e.printStackTrace();
                     }
             });
     } catch (Exception e) {
             e.printStackTrace();
     }
}
countDownLatch.await(5, TimeUnit.SECONDS);
```

Parameter	Description
topic_name	Topic name, which can be copied under the **Topic** tab on the **Cluster** page in the consol
TAG	A parameter used to set the message tag.

One-way sending



```
for (int i = 0; i < 10; i++) {
     // Create a message instance and set the topic and message content
     Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " + i).getBytes(
     // Send one-way messages
     producer.sendOneway(msg);
}
```

Parameter	Description
topic_name	Topic name, which can be copied under the **Topic** tab on the **Cluster** page in the consol

TAG	A parameter used to set the message tag.

Notes

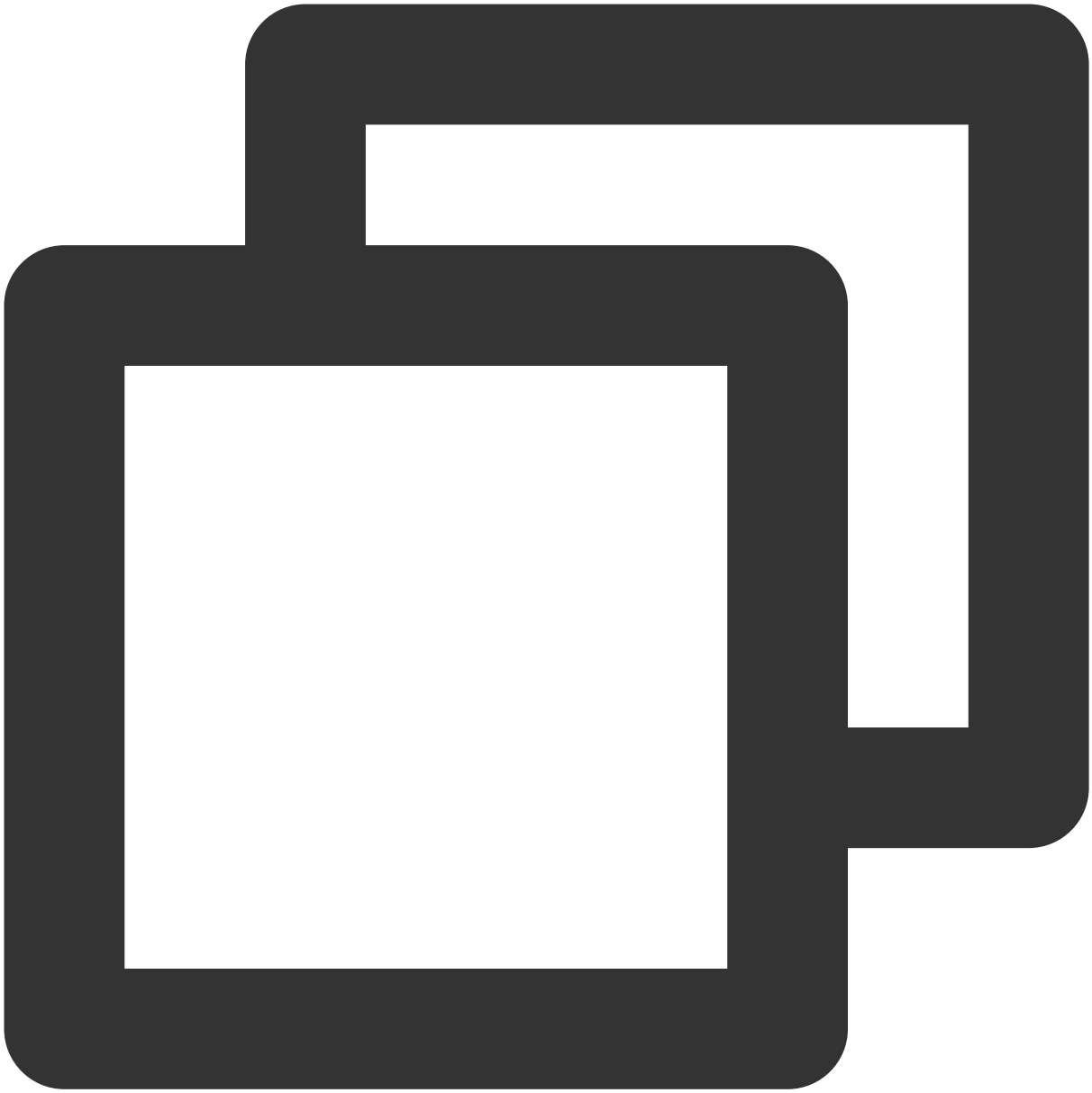
For more information on batch sending or other scenarios, see Demo or RocketMQ documentation.

Step 3. Consume messages

1. Create a consumer

TDMQ for RocketMQ supports two consumption modes: push and pull.

Push consumer

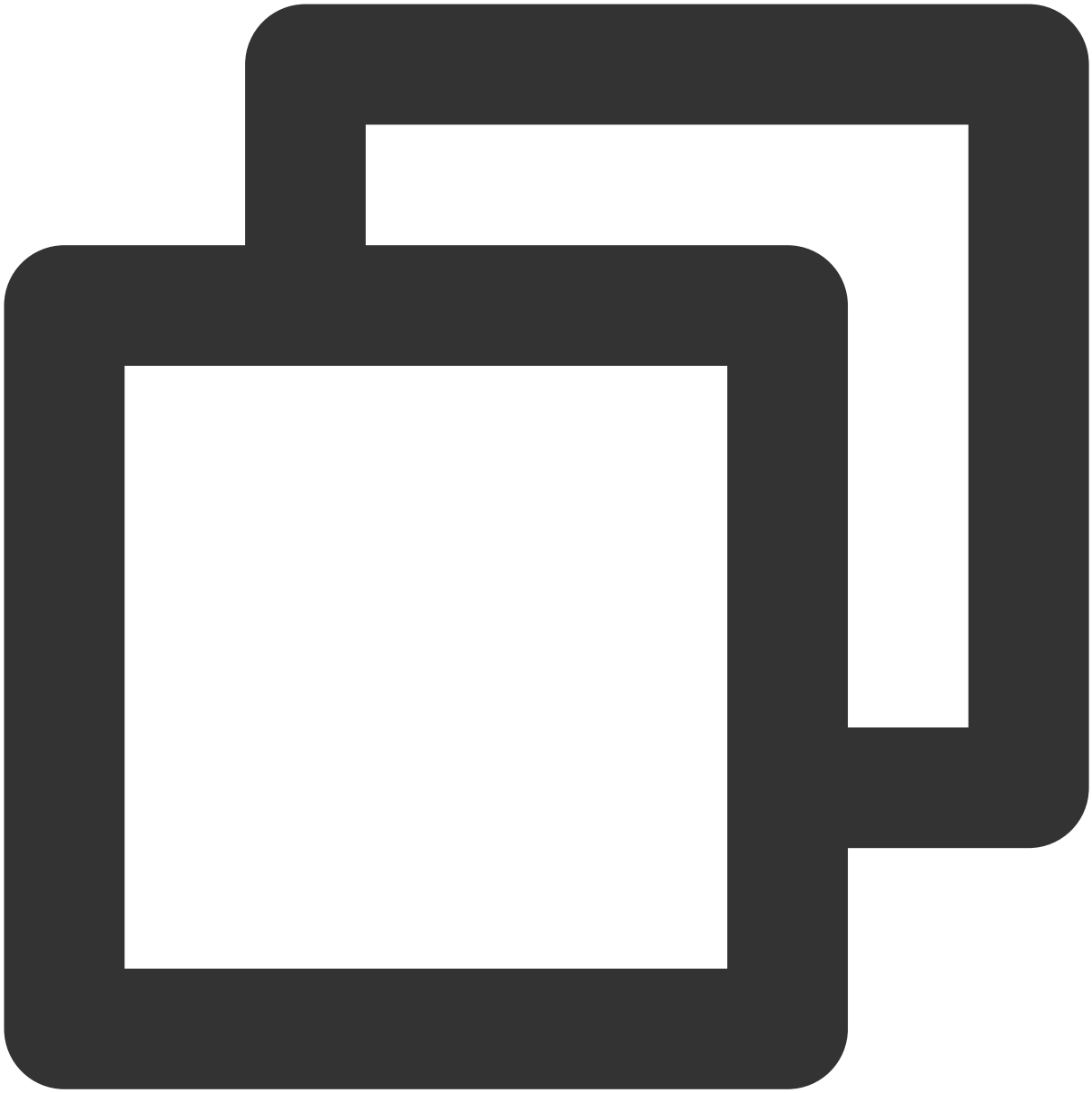


```
// Instantiate the consumer
DefaultMQPushConsumer pushConsumer = new DefaultMQPushConsumer(
     namespace,
     groupName,
     new AclClientRPCHook(new SessionCredentials(accessKey, secretKey))); //ACL per
// Set the NameServer address
pushConsumer.setNamesrvAddr(nameserver);
```

Parameter	Description

namespace	Namespace name, which can be copied under the **Namespace** tab in the console. Its format is clus
groupName	Producer group name, which can be copied under the **Group** tab on the **Cluster** page in the consol
nameserver	Cluster access address, which can be obtained from **Access Address** in the **Operation** column or
secretKey	Role name, which can be copied on the Role Management page.
accessKey	Role token, which can be copied in the **Token** column on the Role Management page. 

Pull consumer



```
// Instantiate the consumer
DefaultLitePullConsumer pullConsumer = new DefaultLitePullConsumer(
     namespace,
     groupName,
     new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)));
// Set the NameServer address
pullConsumer.setNamesrvAddr(nameserver);
// Specify the first offset as the start offset for consumption
pullConsumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET);
```

Parameter	Description
namespace	Namespace name, which can be copied under the **Namespace** tab in the console. Its format is clus
groupName	Producer group name, which can be copied under the **Group** tab on the **Cluster** page in the consol
nameserver	Cluster access address, which can be obtained from **Access Address** in the **Operation** column or console.
secretKey	Role name, which can be copied on the Role Management page.
accessKey	Role token, which can be copied in the **Token** column on the Role Management page. 

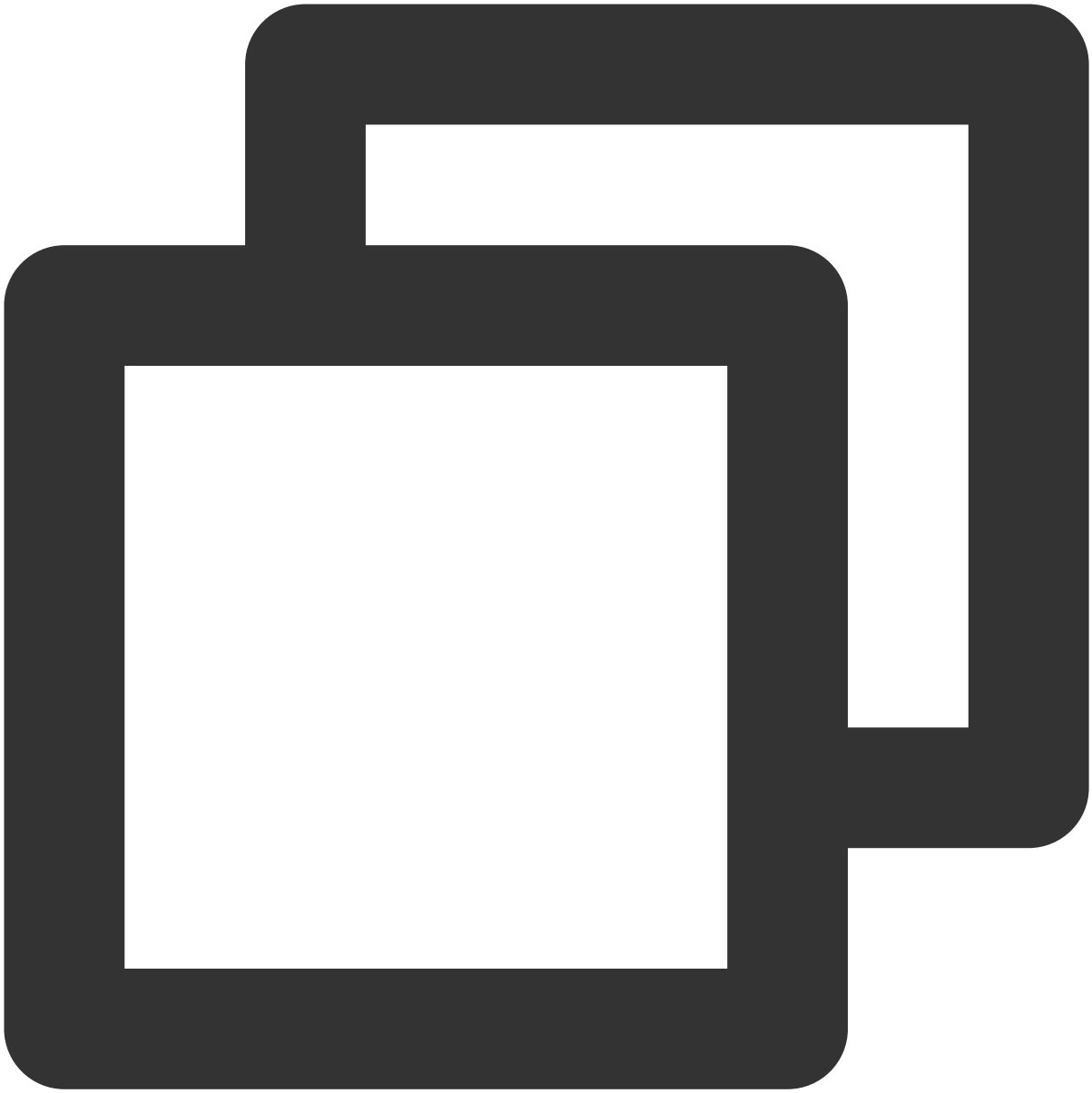
Notes

For more consumption mode information, see Demo or RocketMQ documentation.

2. Subscribe to messages

The subscription modes vary by consumption mode.

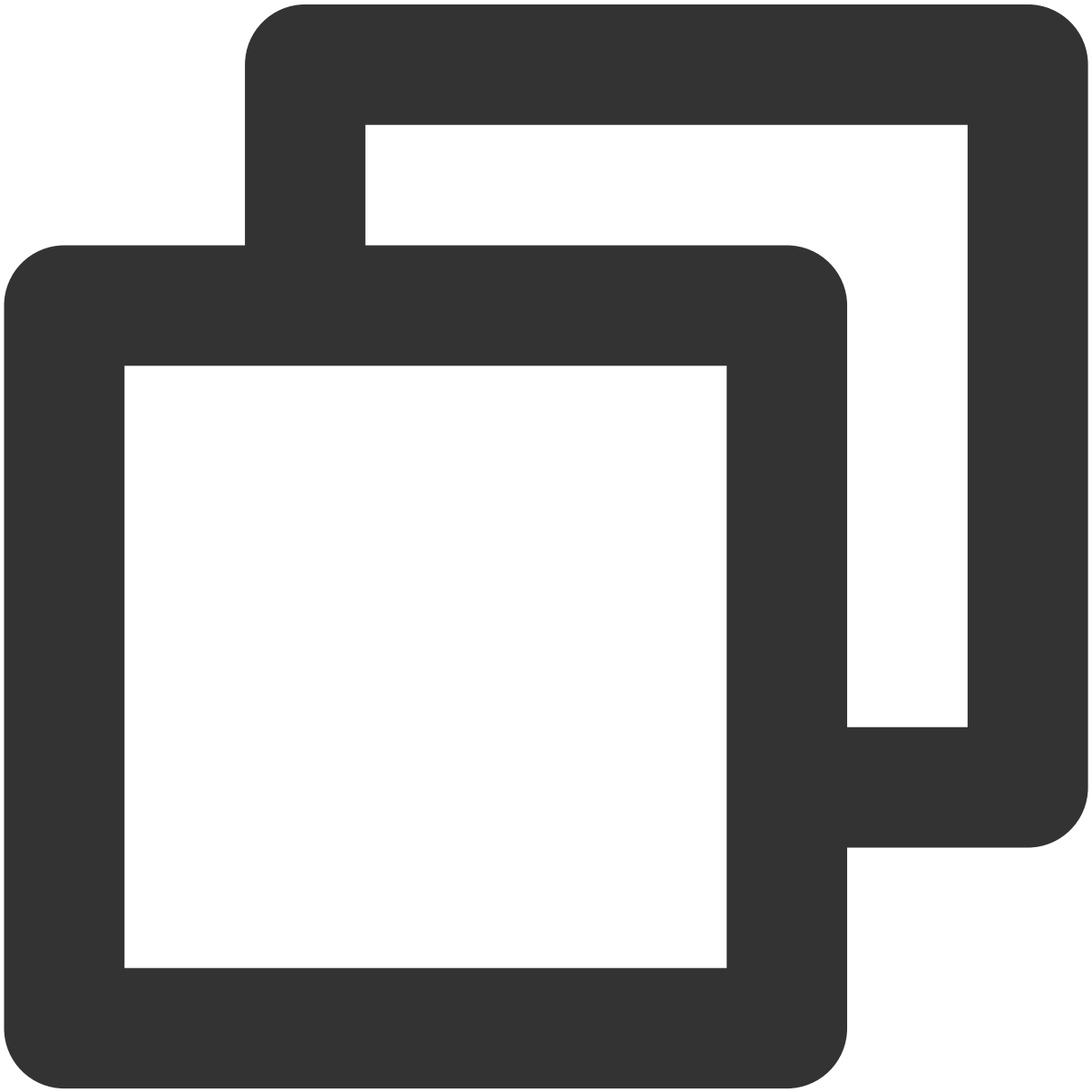
Push subscription



```
// Subscribe to a topic
pushConsumer.subscribe(topic_name, "*");
// Register a callback implementation class to process messages pulled from the bro
pushConsumer.registerMessageListener((MessageListenerConcurrently) (msgs, context)
     // Message processing logic
     System.out.printf("%s Receive New Messages: %s %n", Thread.currentThread().get
     // Mark the message as being successfully consumed and return the consumption
     return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
});
// Start the consumer instance
pushConsumer.start();
```

Parameter	Description
topic_name	Topic name, which can be copied under the **Topic** tab on the **Cluster** page in the console.
"*"	If the subscription expression is left empty or specified as asterisk (*), all messages are subscribed `tag2 || tag3` means subscribing to multiple types of tags.

Pull subscription



```
// Subscribe to a topic
```

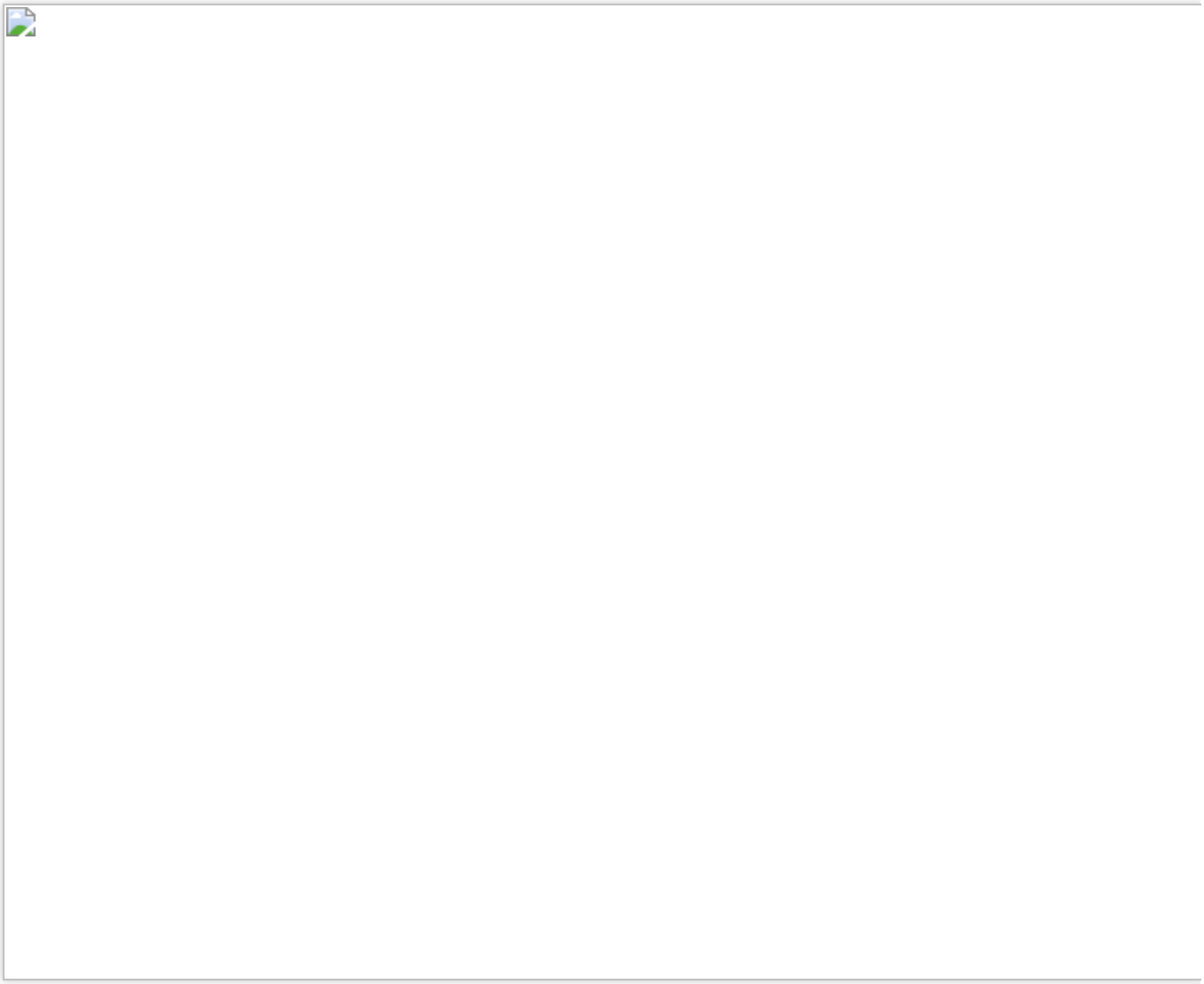
```
pullConsumer.subscribe(topic_name, "*");
// Start the consumer instance
pullConsumer.start();
try {
     System.out.printf("Consumer Started.%n");
     while (true) {
             // Pull the message
             List<MessageExt> messageExts = pullConsumer.poll();
             System.out.printf("%s%n", messageExts);
     }
} finally {
     pullConsumer.shutdown();
}
```

Parameter	Description
topic_name	Topic name, which can be copied under the **Topic** tab on the **Cluster** page in the console.
"*"	If the subscription expression is left empty or specified as asterisk (*), all messages are subscribed `tag2 || tag3` means subscribing to multiple types of tags.

Step 4. View consumption details

Log in to the TDMQ console, go to the **Cluster** > **Group** page, and view the list of clients connected to the group.

Click **View Details** in the **Operation** column to view consumer details.



Notes

Above is a brief introduction to message publishing and subscription. For more information, see Demo or RocketMQ documentation.