

TDMQ for RocketMQ

Getting Started

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

Getting Started

Overview

Messaging over TCP

Resource Creation and Preparation

Downloading and Running Demo

Getting Started

Overview

Last updated : 2023-04-14 16:54:57

TDMQ for RocketMQ supports using multi-language client SDKs to send and receive messages over the TCP and HTTP protocols. This document describes the operation process of sending and receiving general messages over these two protocols.

Notes

TDMQ for RocketMQ supports four types of messages: general, timed/delayed, sequential messages, and transactional. This document takes general message as an example. For other types of messages, refer to [Message Type](#).

Note:

Topics of different message types cannot be mixed, so the topics you create for general messages cannot be used to send and receive messages of other types.

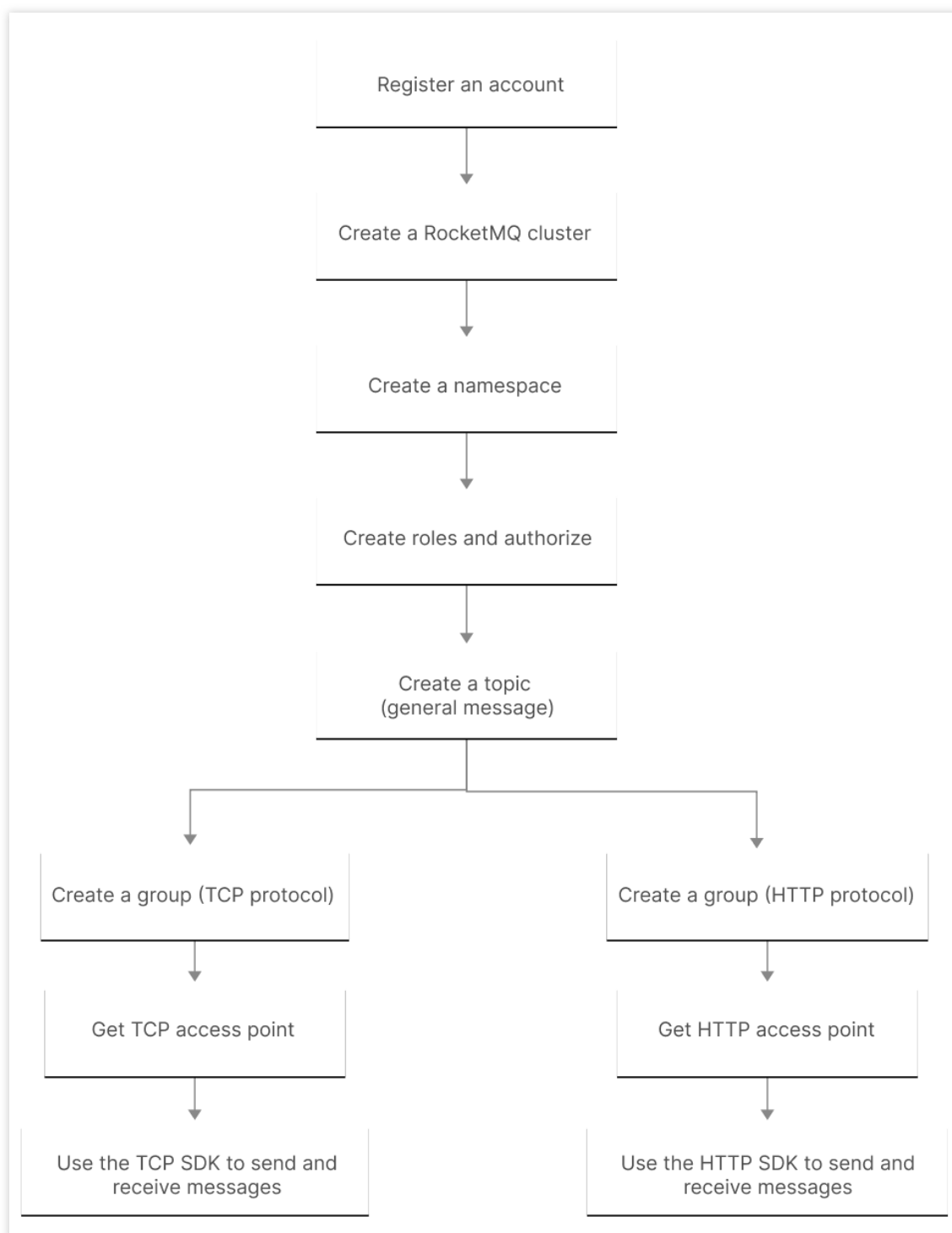
TDMQ for RocketMQ supports accesses over the TCP and HTTP protocols. Therefore, we recommend that you create corresponding types of groups for these two protocols. If multiple consumers use the same group to consume messages, with some using the TCP protocol and others using the HTTP protocol, this may result in consumption failure and message repetition or loss.

Both the TCP and HTTP protocols support the public network and VPC access addresses, and VPC is used in the production environment by default. Public network access is not enabled by default. If you are using a virtual cluster, you can [submit a ticket](#) for application to enable it. If you are using a exclusive cluster, you can [adjust public network bandwidth](#) to enable or disable it. We recommend that you use public network access only in scenarios such as testing and debugging that do not affect the production environment.

Note:

The TCP and HTTP protocols can be supported in all regions. If the region where your current instance resides does not support the HTTP protocol and you need to use it, you can [submit a ticket](#) for application.

Directions



Messaging over TCP

Resource Creation and Preparation

Last updated : 2023-09-12 16:08:15

Overview

Before using SDK to send and receive messages over TCP, you need to create resources such as clusters and topics in the TDMQ for RocketMQ console, and configure related resource information when running the client.

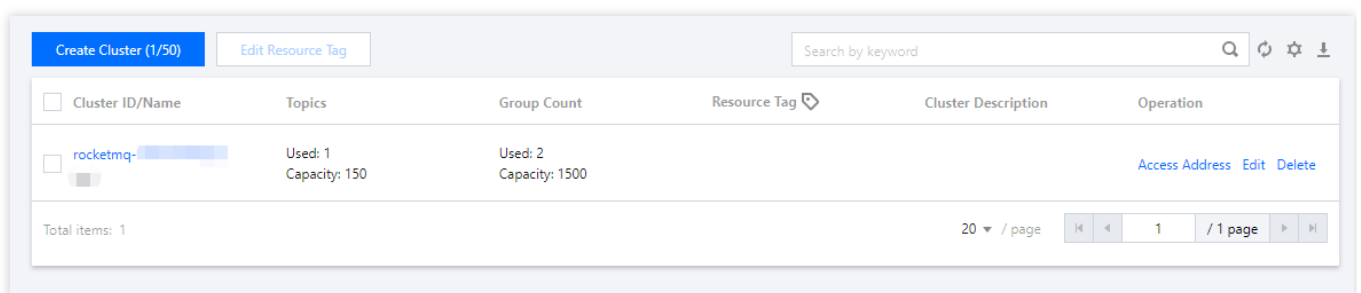
Prerequisites

You have signed up for a Tencent Cloud account as instructed in [Signing Up](#).

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 select **Virtual cluster**. Then, enter the cluster name and description, and click **OK** to create a cluster.



3. On the cluster list page, click the ID of the cluster you just created. In the network module of the cluster's basic information page, you can view the access point information of the cluster.

Network

VPC Access Address <http://rocketmq-5zkv3ew9qqqr.rocketmq.ap-gz.qcloud.tencentttdmq.com:5098>

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 and description to create a namespace.

Basic Info	Namespace	Topic	Group
<div>Create (1/10)</div> <div>Enter a keyword <input type="text"/></div>			
Namespace Name	Message Retention Period ⓘ	Description	Operation
sdaa rocketmq	3 days		Configure Permission Edit Delete
Total items: 1		20 / page	<div>1 / 1 page</div>

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 in **step 1** 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.

Create ✕

Role

super

▼

🔄

Unable to find a role? Please configure a role and token on the [Role Management](#) [🔗](#) page.

Permission

☒ Message production

☒ Message consumption

For more permission type information, see [Permission Description](#) [🔗](#).

Save

Cancel

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**. Then, enter the topic name, select **General message** as the message type, and click **OK** to create a topic.

Note

This document takes sending and receiving general messages as an example. Therefore, the topic of general messages created by referring to the above steps cannot be used for messages of other types.

Basic Info

Namespace

Topic

Group

Current Namespace

sdad

Message Retention Period

3 days

Max TPS

4000

Create (1/150)

Enter a keyword

🔍

🔄

⚙️

📄

Topic Name	Monitoring	Type ▼	Partition Count	Subscribed Groups	Description	Operation
dsada		General message	7	0		Edit Delete

Total items: 1

20 / page

1

/ 1 page

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**. Then, enter the group name, select **TCP** as the protocol type, and click **OK** to create a group.

Note

TDMQ for RocketMQ supports the TCP and HTTP protocols. Therefore, we recommend that you create corresponding types of groups for these two protocols. If multiple consumers use the same group to consume messages, with some using the TCP protocol and others using the HTTP protocol, this may result in consumption failure and message repetition or loss.

Basic InfoNamespaceTopicGroup

Current Namespace

sdaa

Message Retention Period 3 daysMax TPS ⓘ 4000

Create (2/1500)

Search by keyword

Group Name	Consumer Info ⚙	Consumption Mode	Description	Operation
<div></div>	Online Consumer0TPS0Total Heap0⚙	Unknown		Consumer Details Reset Offset Delete
<div></div>	Online Consumer0TPS0Total Heap0⚙	Unknown		Consumer Details Reset Offset Delete

Total items: 2

20 / page

1 / 1 page

Downloading and Running Demo

Last updated : 2023-03-28 10:15:36

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.

Note

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.](#)

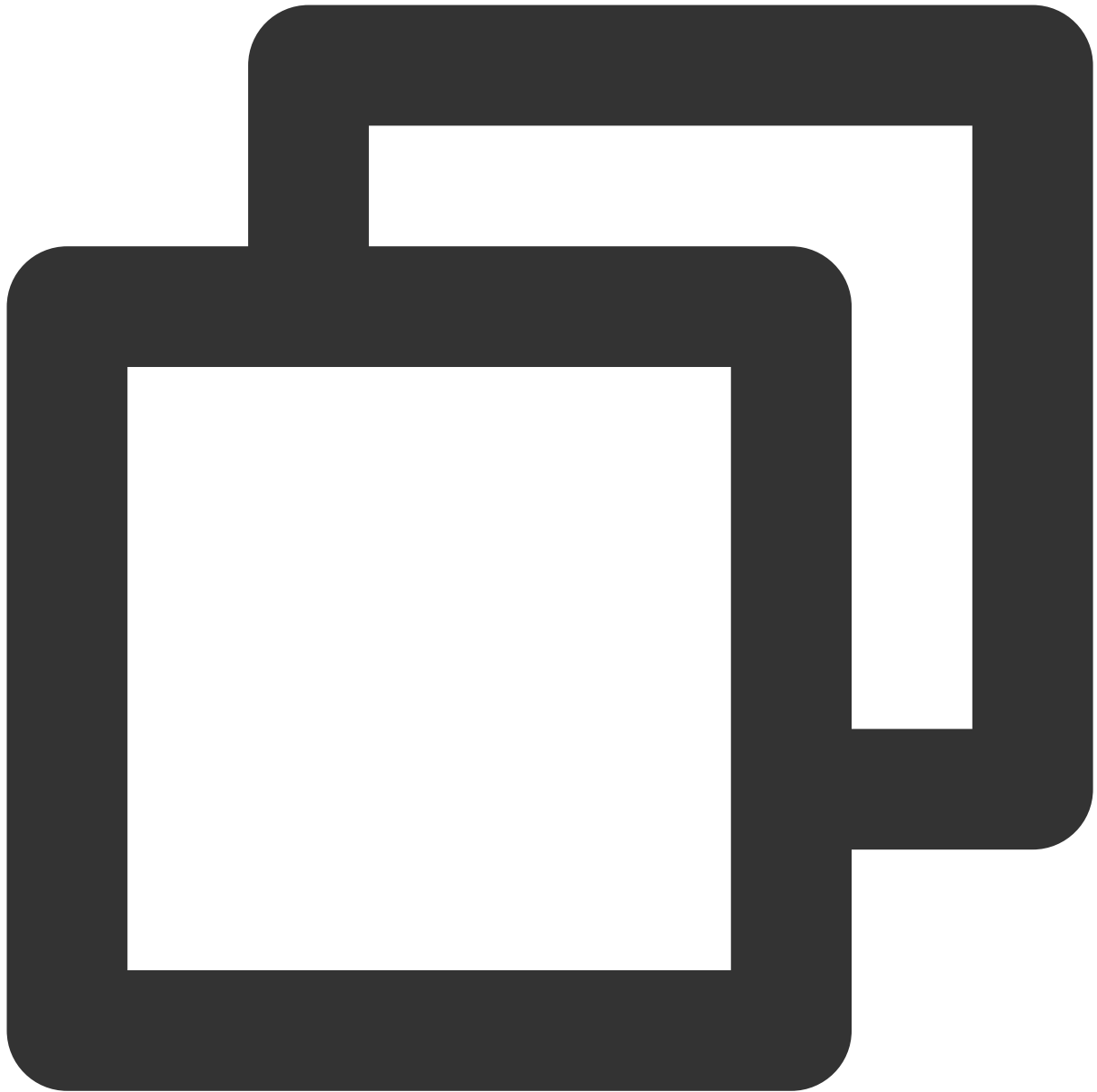
Directions

Step 1. Install the Java dependent 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.

Note

The dependency version must be v4.9.3 or later.



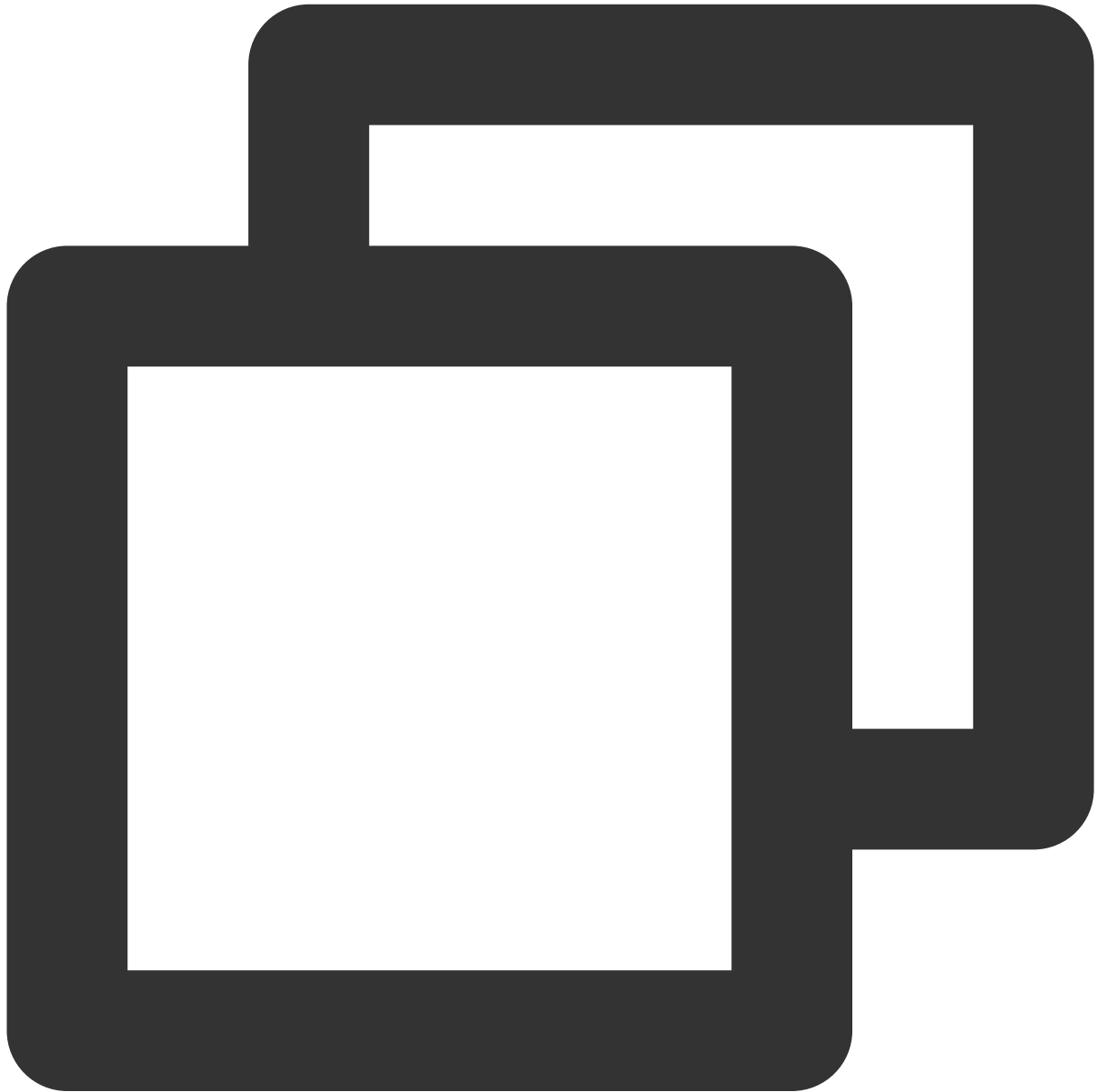
```
<!-- in your <dependencies> block -->
<dependency>
  <groupId>org.apache.rocketmq</groupId>
  <artifactId>rocketmq-client</artifactId>
  <version>4.9.3</version>
</dependency>

<dependency>
  <groupId>org.apache.rocketmq</groupId>
  <artifactId>rocketmq-acl</artifactId>
  <version>4.9.3</version>
```

```
</dependency>
```



Step 2. Produce messages

1. Create message producers



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

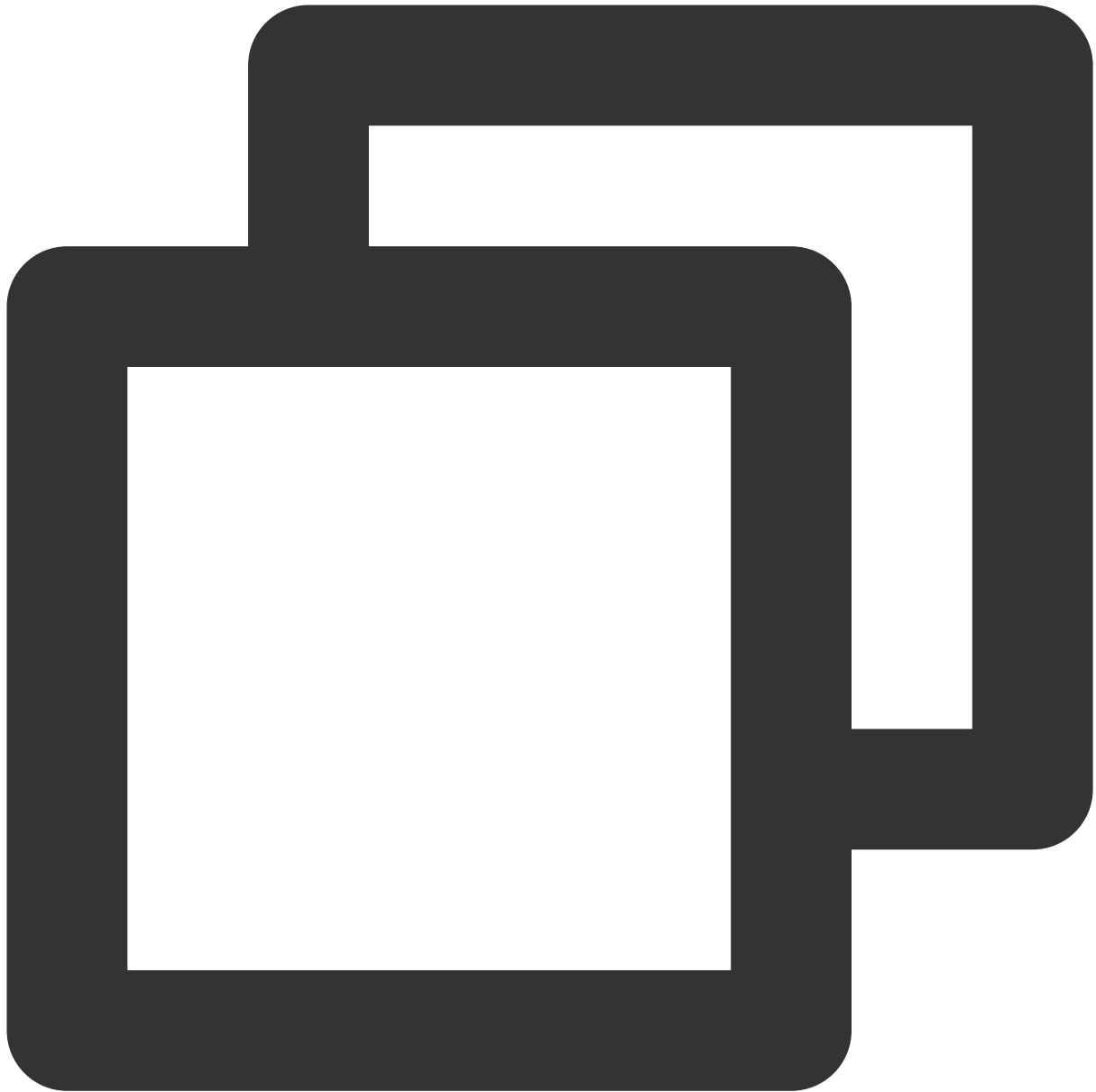
```
// ACL permission
);
// Set the NameServer address
producer.setNamesrvAddr(nameserver);
// Start the producer instances
producer.start();
```

Parameter	Description
namespace	Namespace name, which can be copied on the Namespace page in the console. 
groupName	Producer group name, which can be copied under the Group tab on the Cluster page in the console
nameserver	Cluster access address, which can be copied under the Network module on the cluster's basic info
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 messages

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

Sync sending

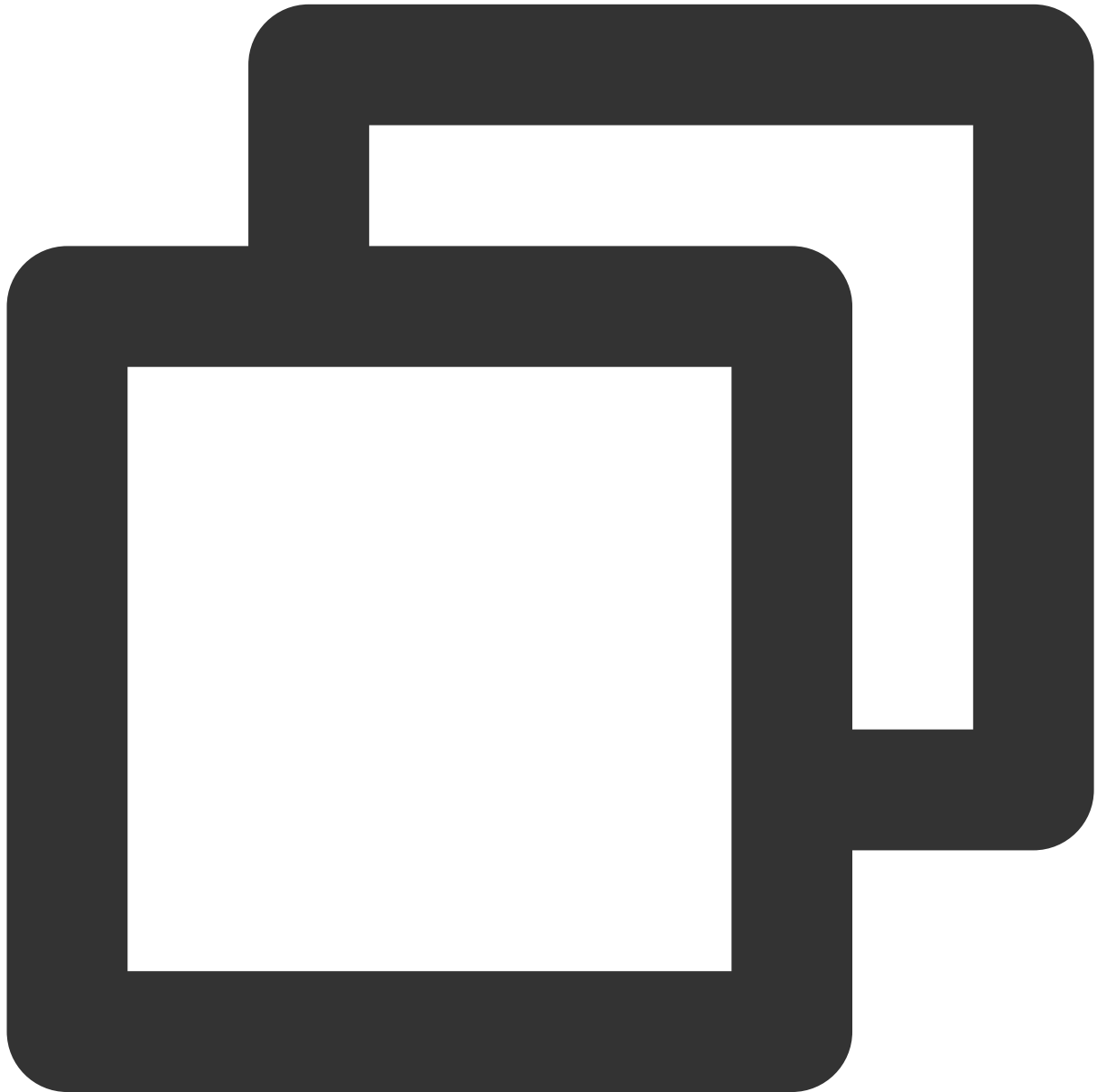


```
for (int i = 0; i < 10; i++) {  
    // Create a message instance and set the topic and message content  
    Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " + i).getBytes(  
    // Send the message  
    SendResult sendResult = producer.send(msg);  
    System.out.printf("%s\n",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

Parameter	Description

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

Async sending



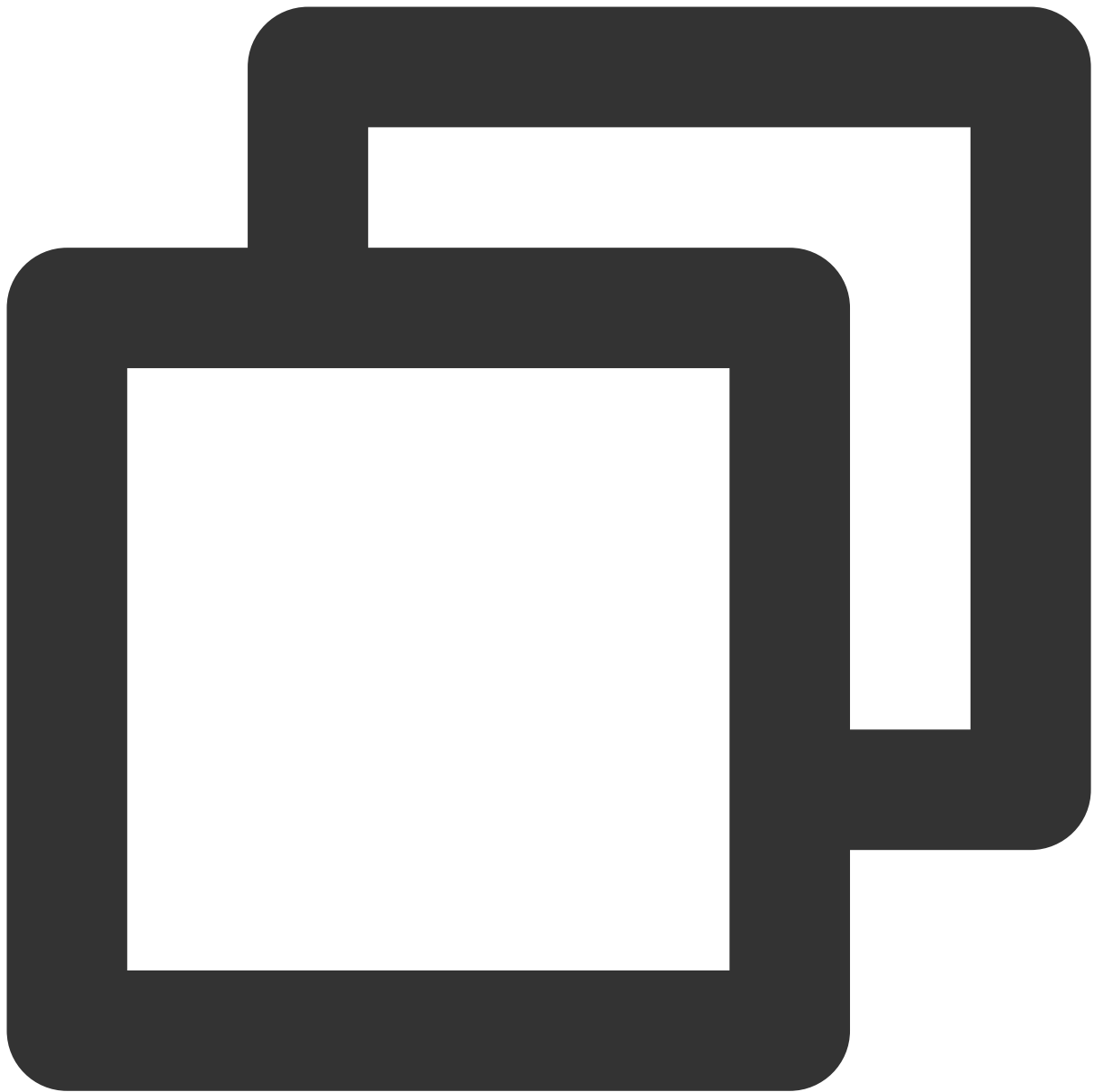
```
// Disable retry upon sending failures
producer.setRetryTimesWhenSendAsyncFailed(0);
// Set the number of messages to be sent
int messageCount = 10;
```

```
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 " + index));
        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.toString());
            }

            @Override
            public void onException(Throwable e) {
                // Logic for message sending failures
                countDownLatch.countDown();
                System.out.printf("%-10d Exception %s %n", index, e.toString());
                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 console.
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

	console.
tag	A parameter used to set the message tag.

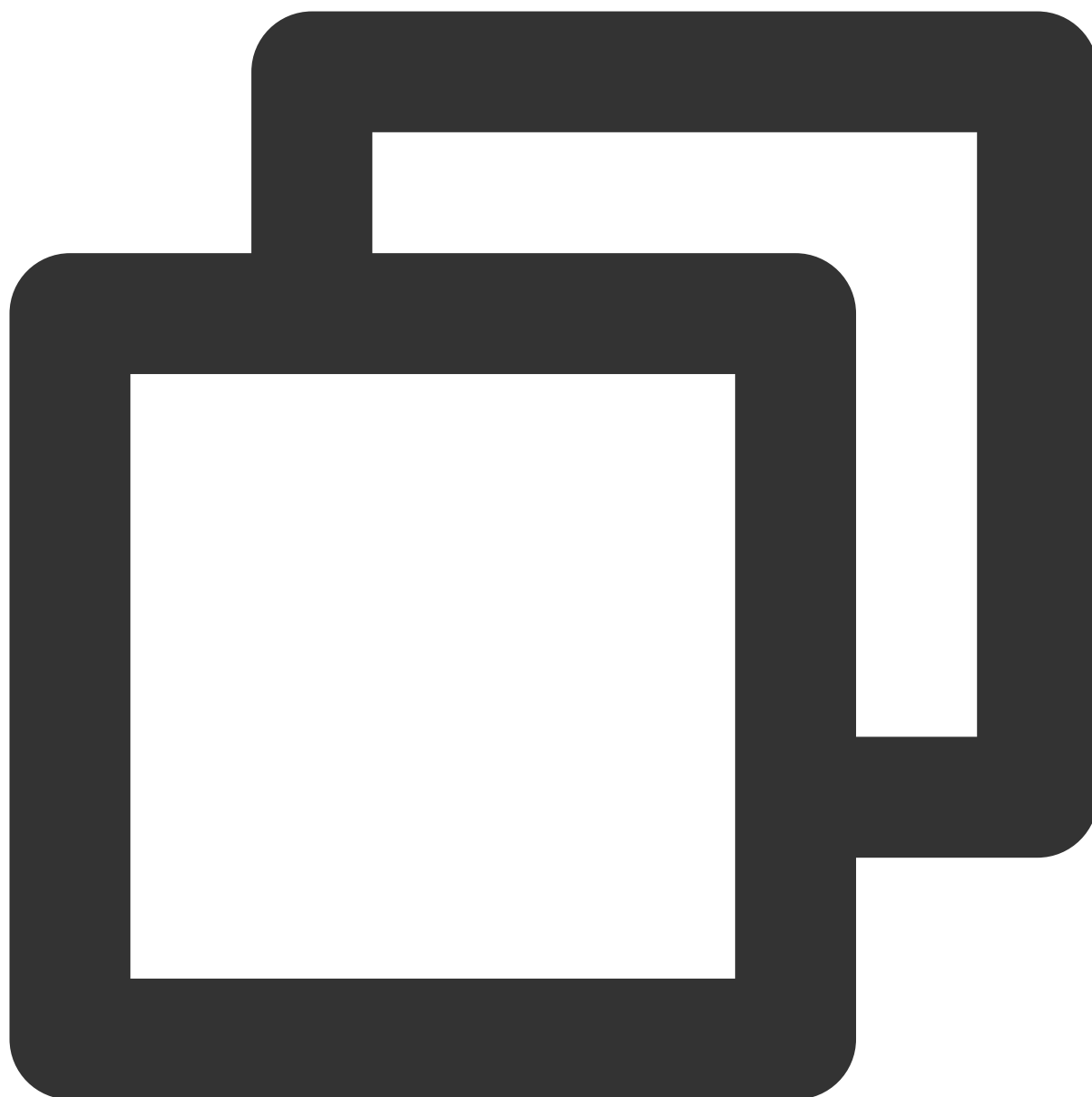
Note

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.

For consumers using the push mode:

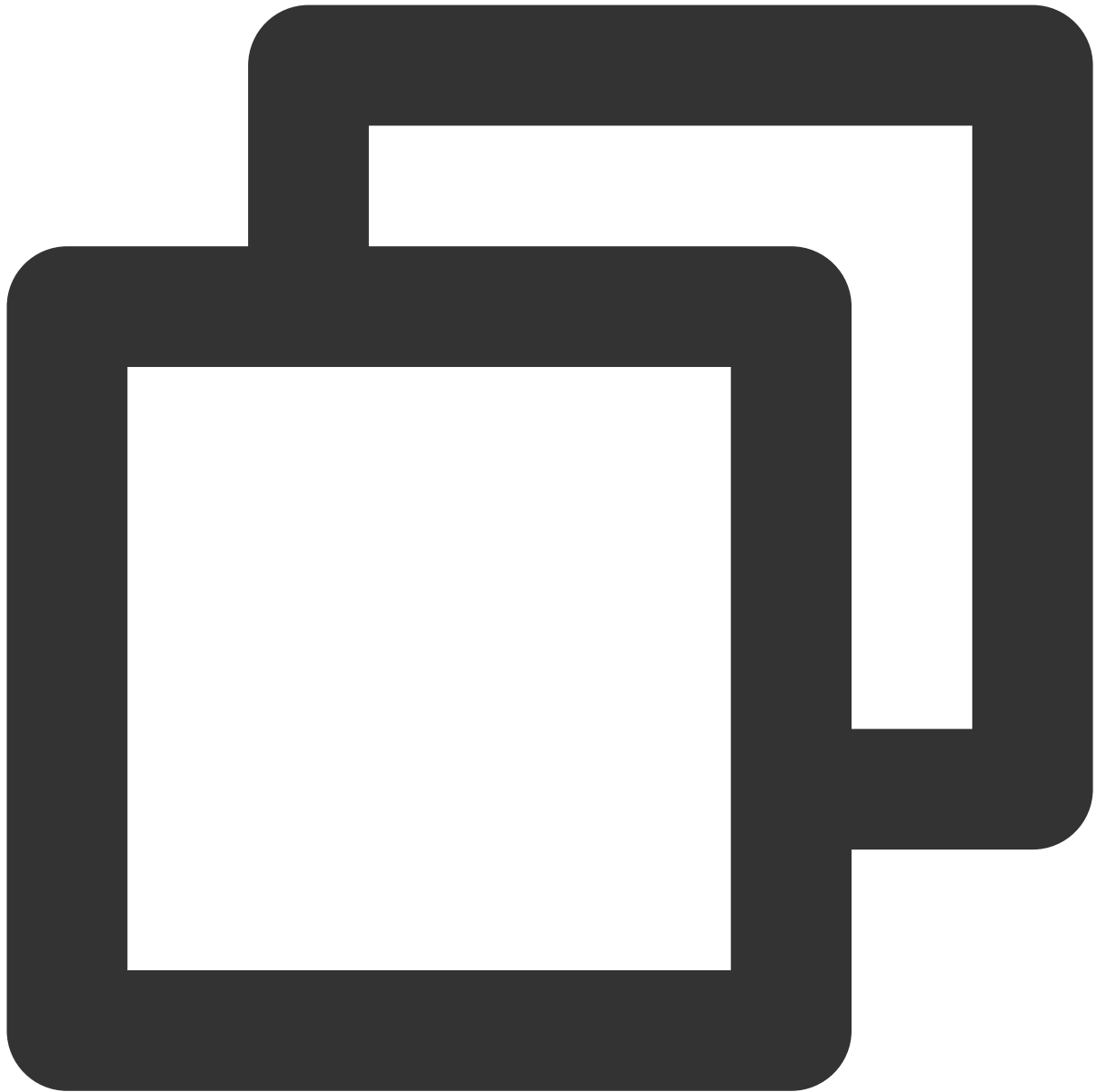


```
// Instantiate the consumer
DefaultMQPushConsumer pushConsumer = new DefaultMQPushConsumer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey))); //ACL per
// Set the NameServer address
pushConsumer.setNamesrvAddr(nameserver);
```


Parameter	Description

namespace	Namespace name, which can be copied on the Namespace page in the console. 
groupName	Producer group name, which can be copied under the Group tab on the Cluster page in the console
nameserver	Cluster access address, which can be copied under the Network module on the cluster's basic info page
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. 

For consumers using the pull mode:



```
// 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 page in the 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. 

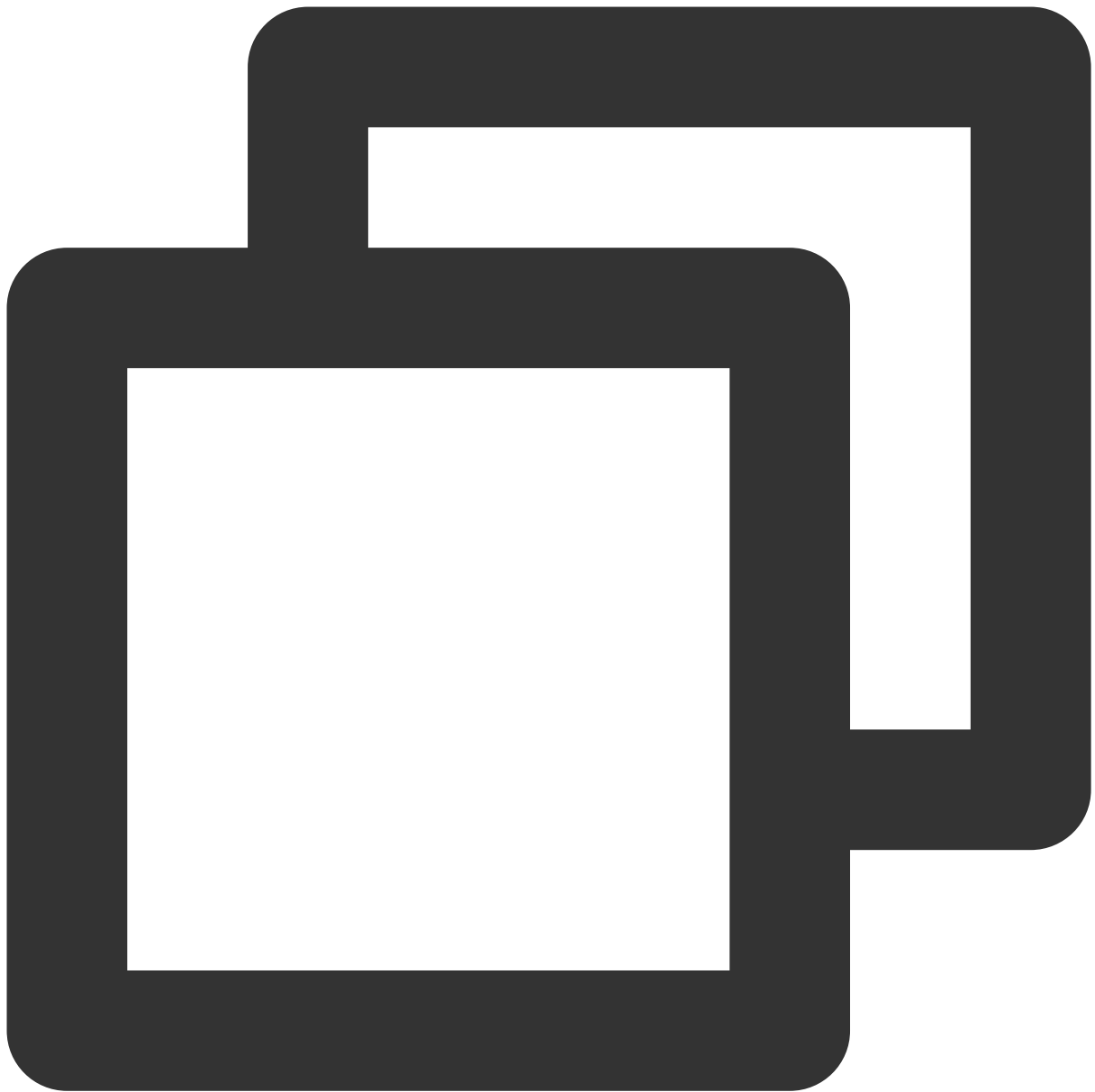
Note

For more consumption mode information, see [Demo](#) or [RocketMQ documentation](#).

2. Subscribe to messages

The subscription modes vary by consumption mode.

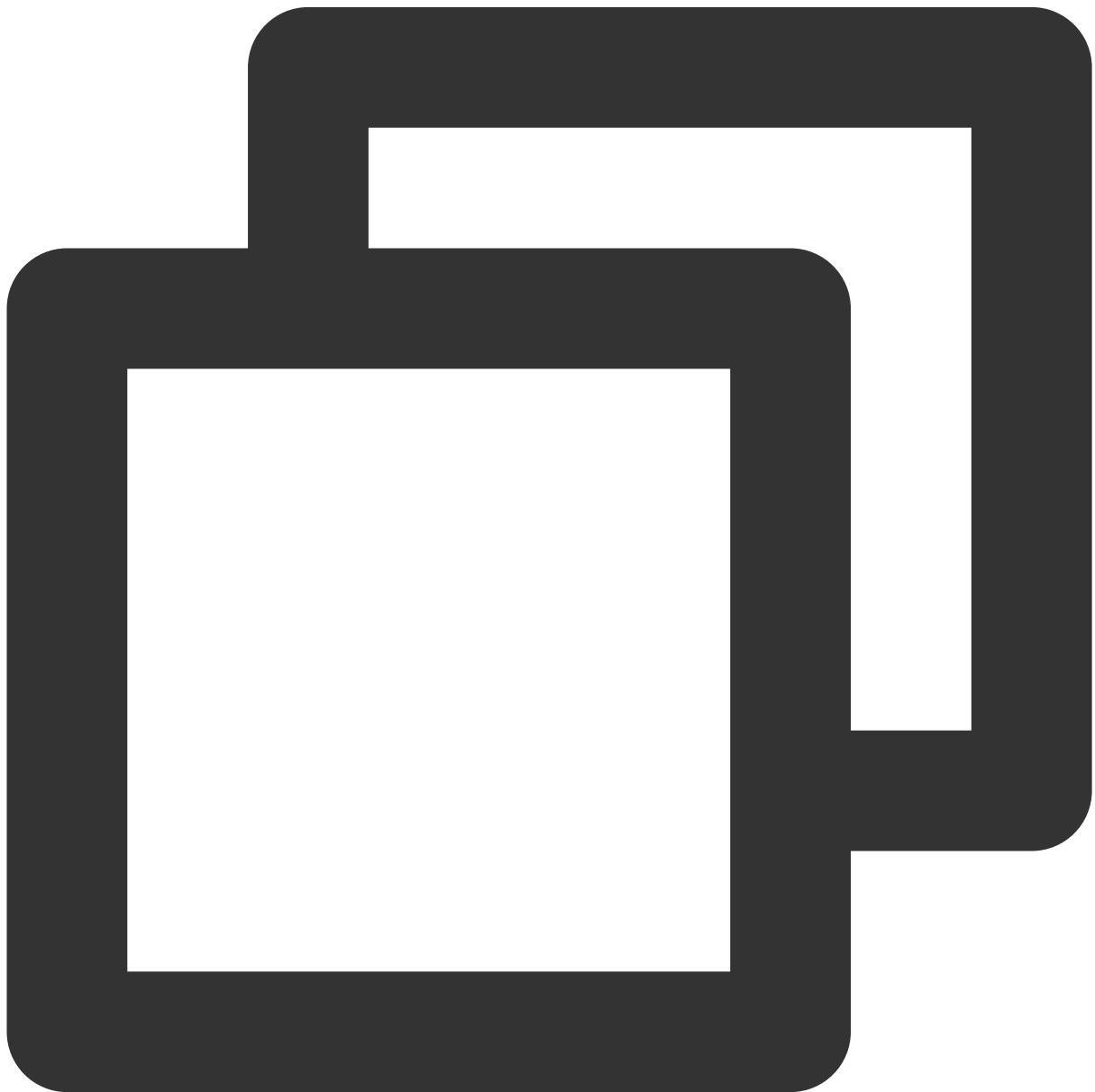
Subscription in push mode



```
// Subscribe to a topic
pushConsumer.subscribe(topic_name, "*");
// Register a callback implementation class to process messages pulled from the broker
pushConsumer.registerMessageListener((MessageListenerConcurrently) (msgs, context)
    // Message processing logic
    System.out.printf("%s Receive New Messages: %s %n", Thread.currentThread().getName(), msgs);
    // Mark the message as being successfully consumed and return the consumption result
    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 to. <code>tag1 tag2 tag3</code> means subscribing to multiple types of tags.

Subscription in pull mode



```
// 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 to. <code>tag1 tag2 tag3</code> 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.

**Note**

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