

# **TDMQ for RocketMQ**

## **Getting Started**

### **Product Documentation**



## Copyright Notice

©2013-2023 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

Sending/Receiving Messages over TCP

Resource Creation and Preparation

Downloading and Running Demo

# 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



- Log in to the [TDMQ console](#), enter the **Cluster** page, and select the target region.
- Click **Create Cluster** and enter the cluster name and description to create a cluster.

| <input type="checkbox"/> Cluster ID/Name | Topics                    | Group Count                | Cluster Description | Resource Tag | Operation  |
|--|---------------------------|----------------------------|---------------------|--------------|--|
| <input type="checkbox"/> rocketmq-test   | Used: 1<br>Capacity: 1000 | Used: 1<br>Capacity: 10000 |                     | 3            | <a href="#">Access Address</a> <a href="#">Edit</a> <a href="#">Delete</a> |



- 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.

## API Call Address

### VPC Access Address

rocketmq- kvmmz8.rocketmq.ap-  
sh.qcloud.tencenttdmq.com:5022 

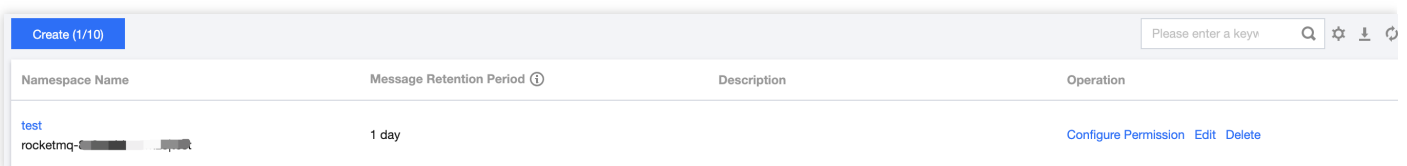
### Public Network Access Address



rocketmq- kvmmz8.rocketmq.ap-  
sh.public.tencenttdmq.com:9876 

OK

## 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.



| Namespace Name  | Message Retention Period  | Description | Operation  |
|---|--|-------------|--|
| test<br>rocketmq-  | 1 day  |             | <a href="#">Configure Permission</a> <a href="#">Edit</a> <a href="#">Delete</a> |

## 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.

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

## Create ✕

Role bigdata ▼

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

Permission  Message production  
 Message consumption

For more permission type information, see [here](#) ↗

Save
Cancel

## Step 4. Create a topic

- On the **Namespace** list page, select the **Topic** tab at the top to enter the **Topic** list page.
- Select the namespace created in [step 3](#) and click **Create** to create a topic.

| Topic Name | Monitori... | Type ▾  | Subscribed Groups | Description | Operation                                   |
|------------|-------------|---------|-------------------|-------------|---|
| test2      |             | General | 0                 |             | <a href="#">Edit</a> <a href="#">Delete</a> |

## Step 5. Create a group

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

| Group Name | Consumer Info ↕                      | Consumption Mode | Description | Operation   |
|------------|--------------------------------------|------------------|-------------|---|
| ▶ t...t    | Online Consumer 0 TPS 0 Total Heap 0 | Unknown          |             | <a href="#">Consumer Details</a> <a href="#">Reset Offset</a> <a href="#">Edit</a> <a href="#">Delete</a> |

# 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](#).

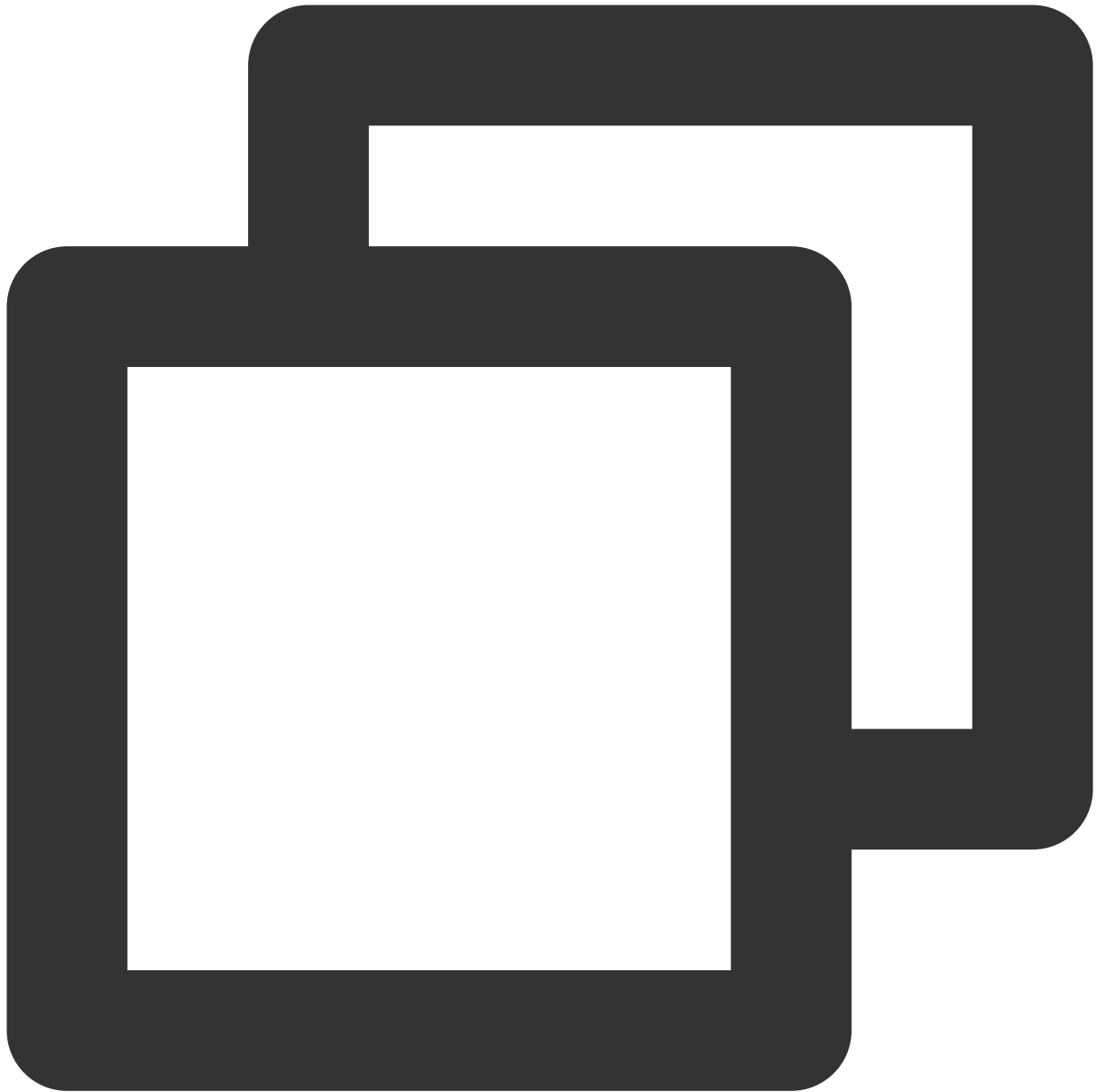
## Directions

### 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>

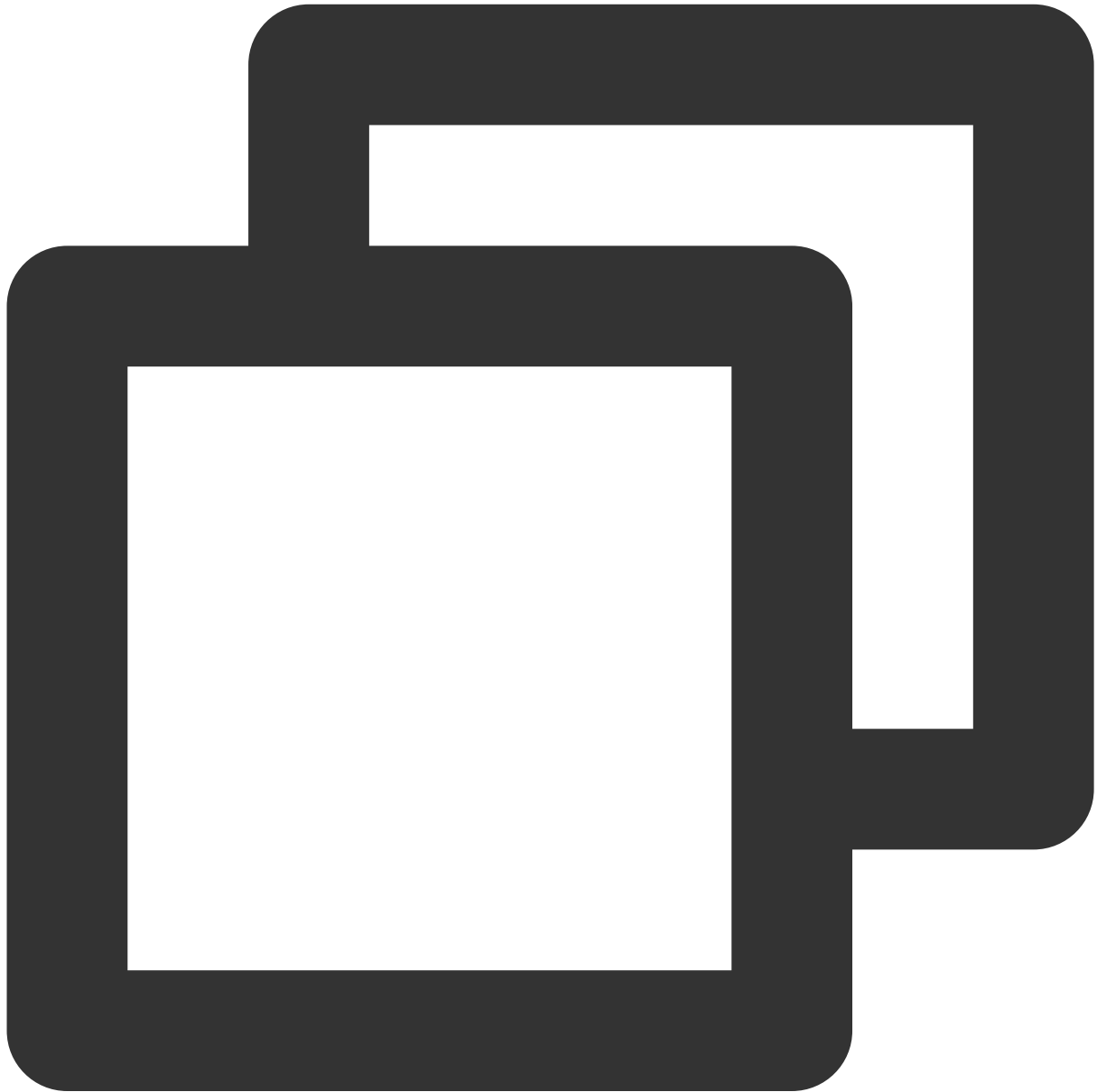
<dependency>
  <groupId>org.apache.rocketmq</groupId>
  <artifactId>rocketmq-acl</artifactId>
  <version>4.6.1</version>
```



```
</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
```

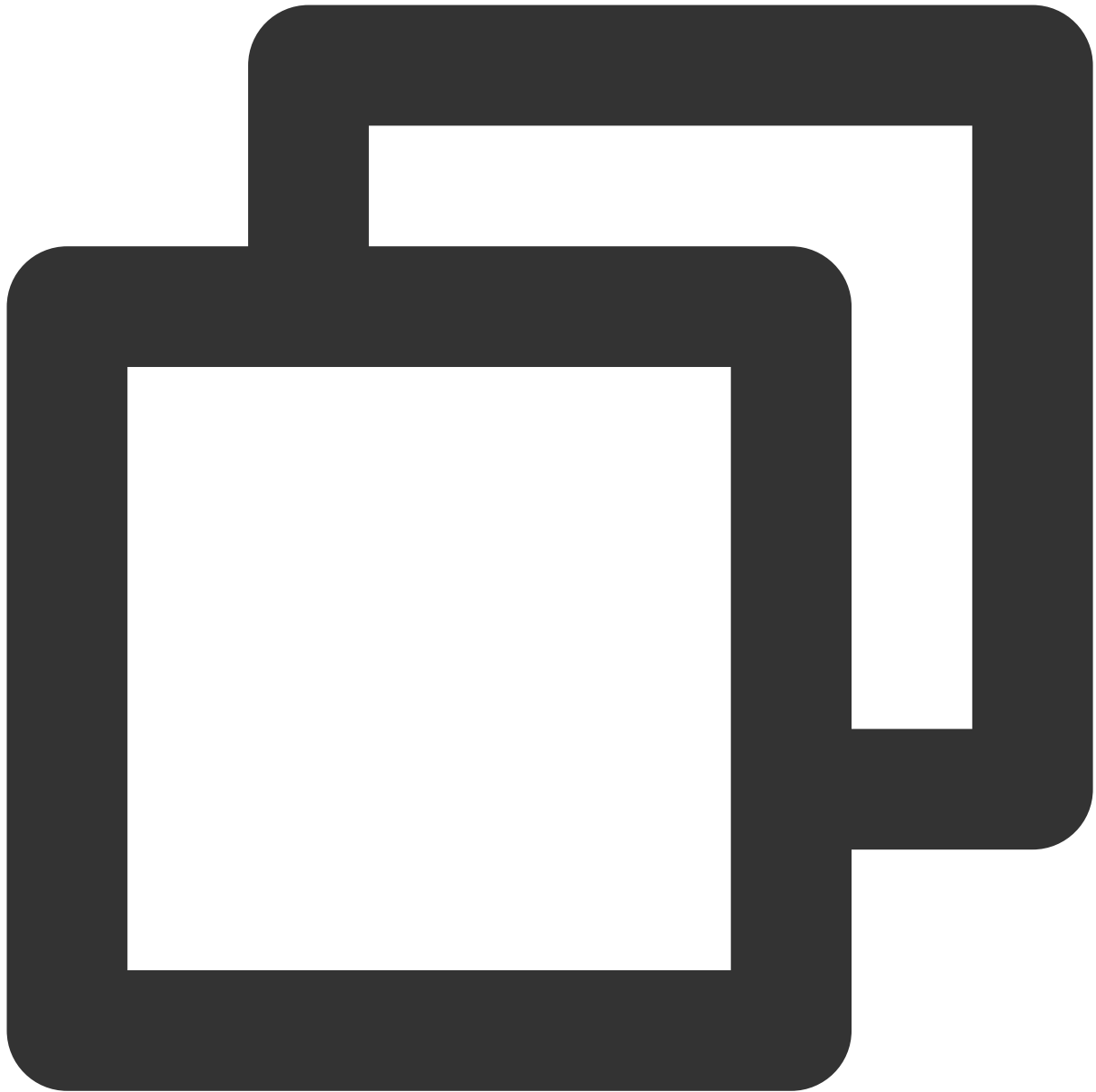
```
);
// Set the NameServer address
producer.setNamesrvAddr(nameserver);
// Start the producer instances
producer.start();
```

| Parameter  | Description   |
|------------|---|
| namespace  | Namespace name, which can be copied under the <b>Namespace</b> tab in the console. Its format is clus               |
| groupName  | Producer group name, which can be copied under the <b>Group</b> tab on the <b>Cluster</b> page in the consol        |
| nameserver | Cluster access address, which can be obtained from <b>Access Address</b> in the <b>Operation</b> column or console. |
| secretKey  | Role name, which can be copied on the <a href="#">Role Management</a> page.   |
| accessKey  | Role token, which can be copied in the <b>Token</b> column on the <a href="#">Role Management</a> page.             |

## 2. Send a message

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

Sync sending

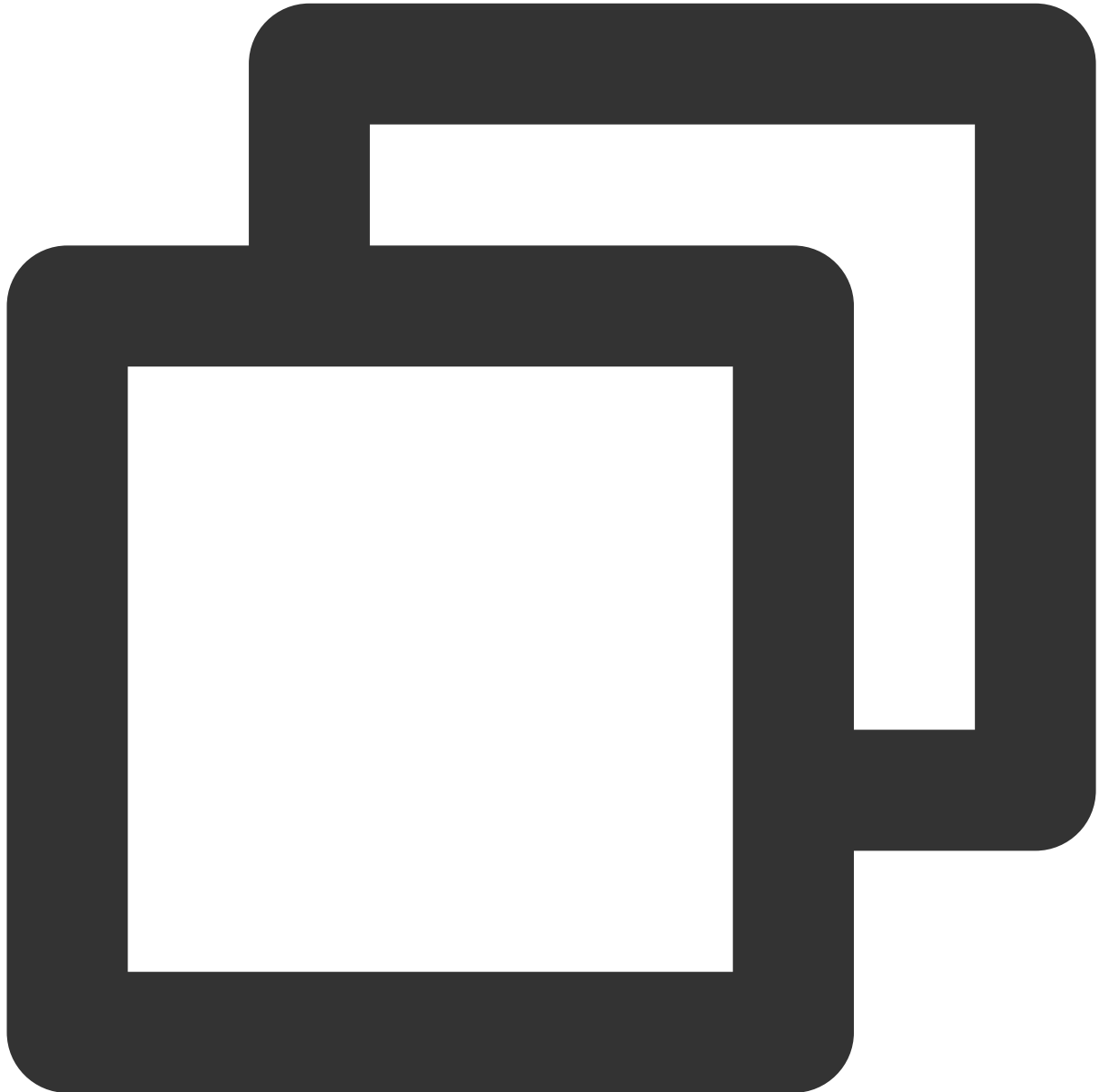


```
for (int i = 0; i < 10; i++) {  
    // Create a message instance and set the topic and message content  
    Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " + i).getBytes()  
    // Send the message  
    SendResult sendResult = producer.send(msg);  
    System.out.printf("%s%n",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

| Parameter | Description |
|-----------|-------------|
|           |             |

|            |   |
|------------|---|
| topic_name | Topic name, which can be copied under the <b>Topic</b> tab on the <b>Cluster</b> page in the console. |
| TAG        | A parameter used to set the message tag.  |

### Async sending



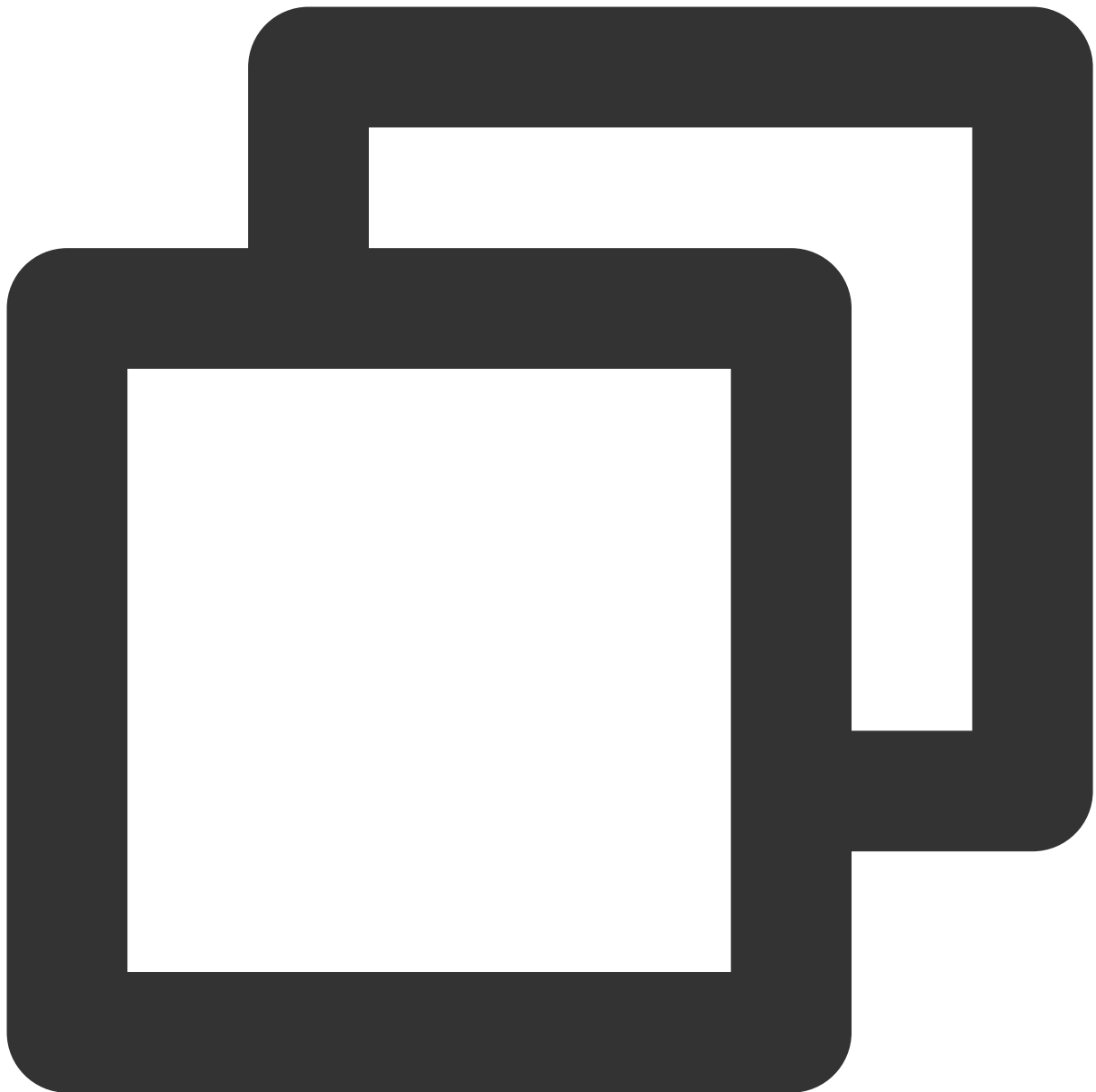
```
// Disable retry upon sending failures
producer.setRetryTimesWhenSendAsyncFailed(0);
// Set the number of messages to be sent
int messageCount = 10;
final CountdownLatch countdownLatch = new CountdownLatch(messageCount);
```

```
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);
            }

            @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 <b>Topic</b> tab on the <b>Cluster</b> 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 <b>Topic</b> tab on the <b>Cluster</b> page in the console |

---

|     |  |
|-----|--|
| TAG | A parameter used to set the message tag. |
|-----|--|

---

### Notes

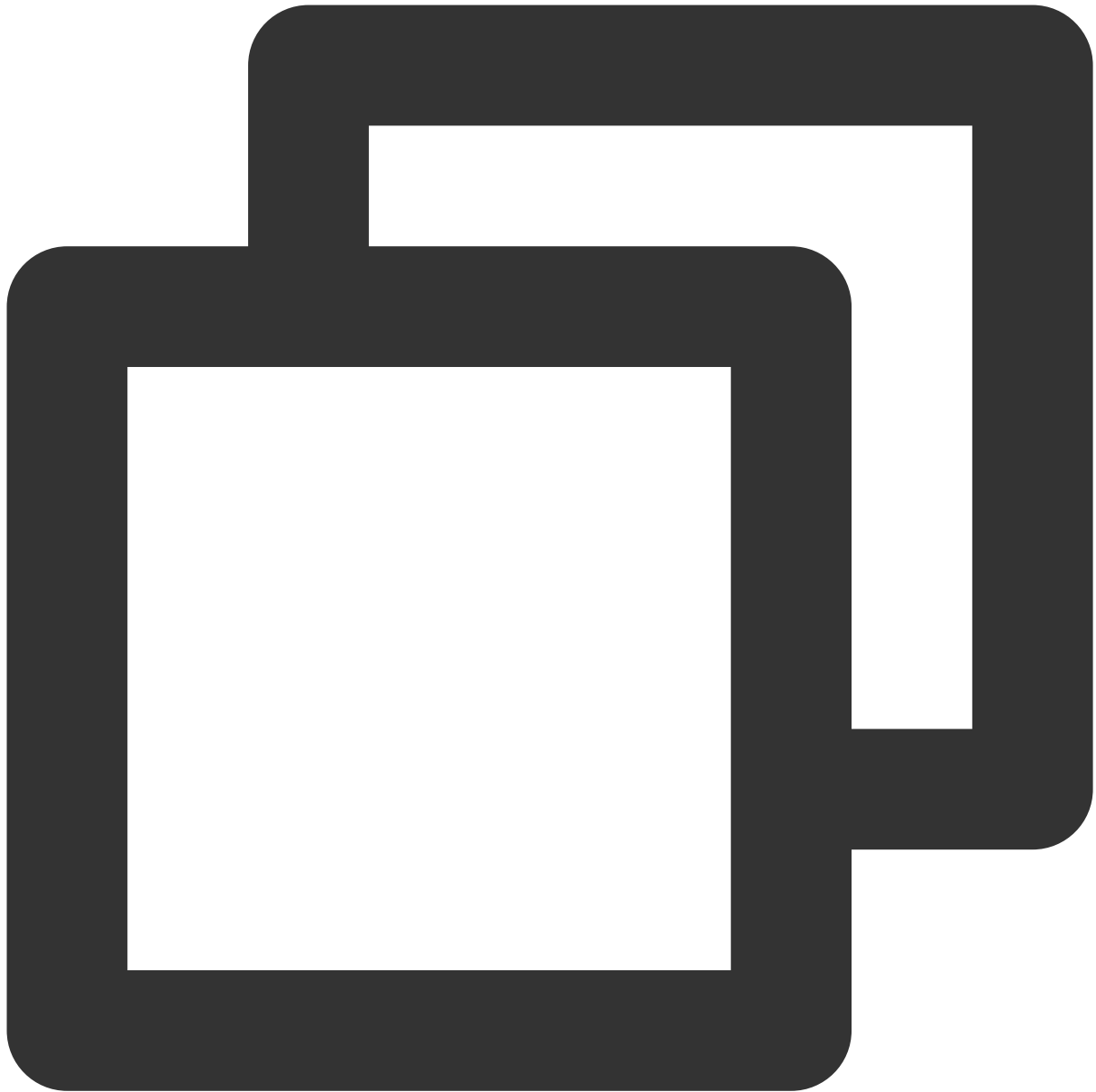
For more information on batch sending or other scenarios, see [Demo](#) or [RocketMQ documentation](#).

## Step 3. Consume messages

### 1. Create a consumer

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


Push consumer



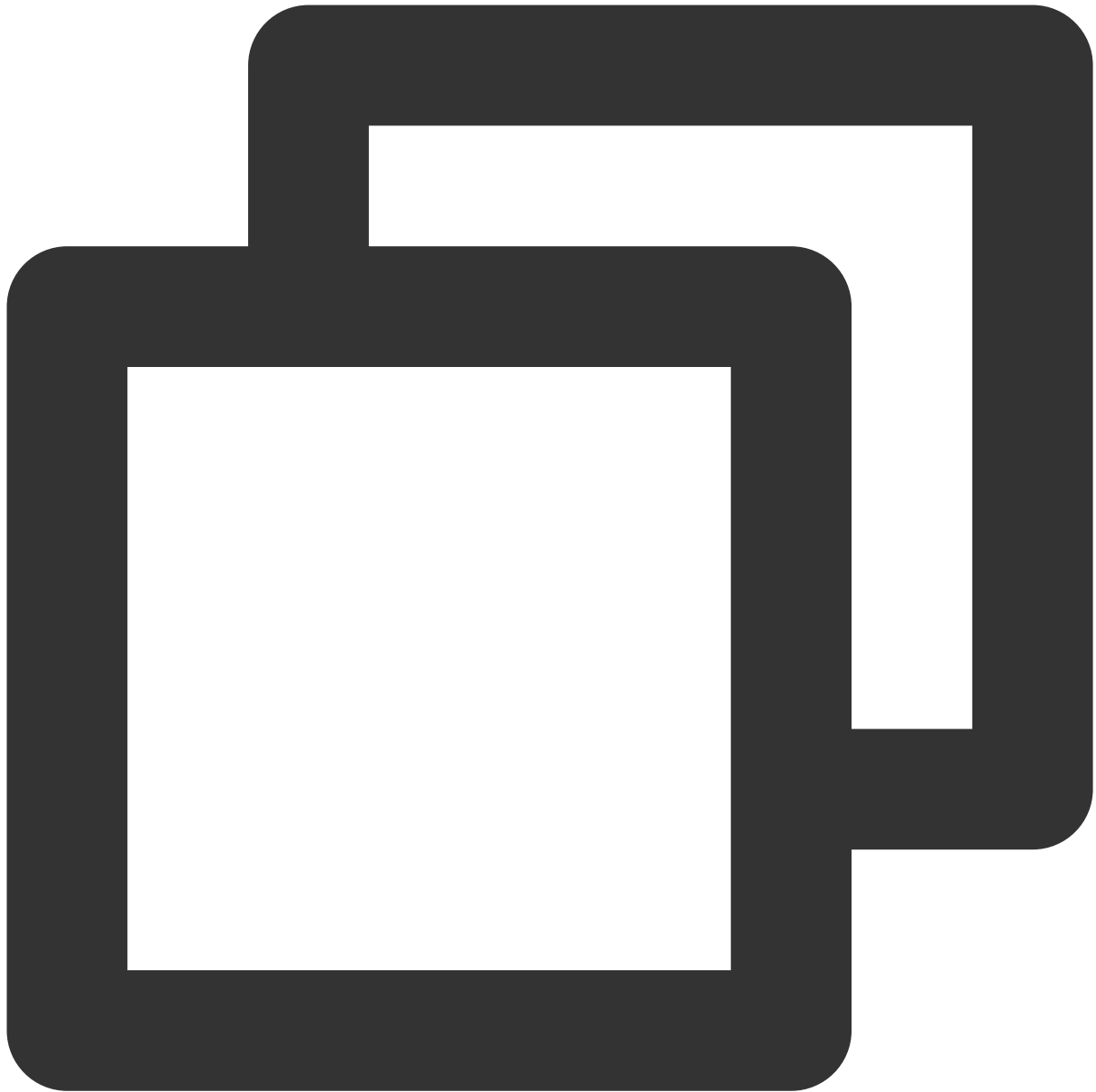
```
// Instantiate the consumer
DefaultMQPushConsumer pushConsumer = new DefaultMQPushConsumer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey))); //ACL per
// Set the NameServer address
pushConsumer.setNamesrvAddr(nameserver);
```

| Parameter | Description |
|-----------|-------------|
|           |             |




|            |  |
|------------|--|
| namespace  | Namespace name, which can be copied under the <b>Namespace</b> tab in the console. Its format is clus  |
| groupName  | Producer group name, which can be copied under the <b>Group</b> tab on the <b>Cluster</b> page in the consol   |
| nameserver | Cluster access address, which can be obtained from <b>Access Address</b> in the <b>Operation</b> column or   |
| secretKey  | Role name, which can be copied on the <a href="#">Role Management</a> page.  |
| accessKey  | Role token, which can be copied in the <b>Token</b> column on the <a href="#">Role Management</a> page.<br> |

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 <b>Namespace</b> tab in the console. Its format is clus  |
| groupName  | Producer group name, which can be copied under the <b>Group</b> tab on the <b>Cluster</b> page in the consol   |
| nameserver | Cluster access address, which can be obtained from <b>Access Address</b> in the <b>Operation</b> column or console.  |
| secretKey  | Role name, which can be copied on the <a href="#">Role Management</a> page.  |
| accessKey  | Role token, which can be copied in the <b>Token</b> column on the <a href="#">Role Management</a> page.<br> |

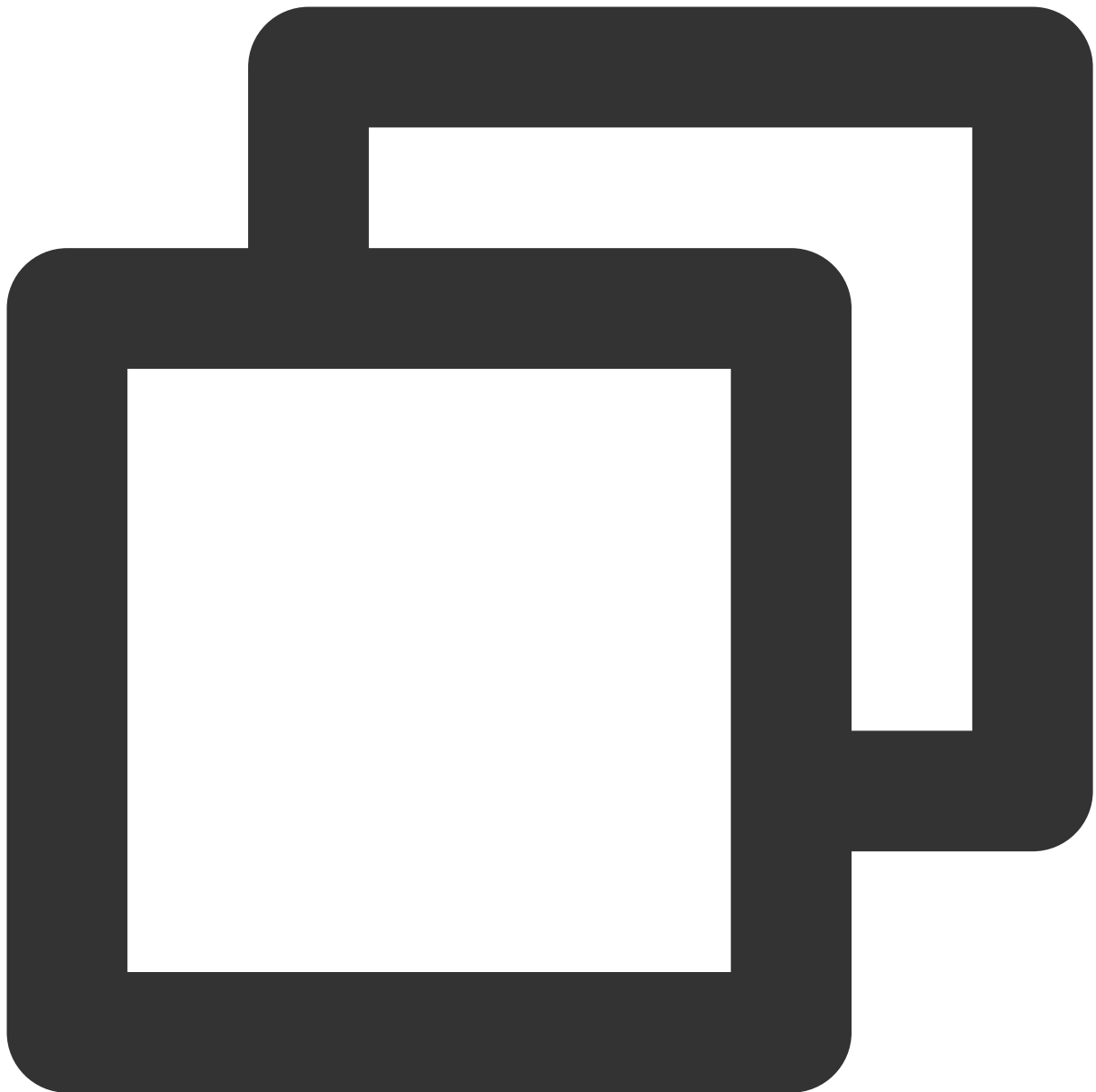
## Notes

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

## 2. Subscribe to messages

The subscription modes vary by consumption mode.

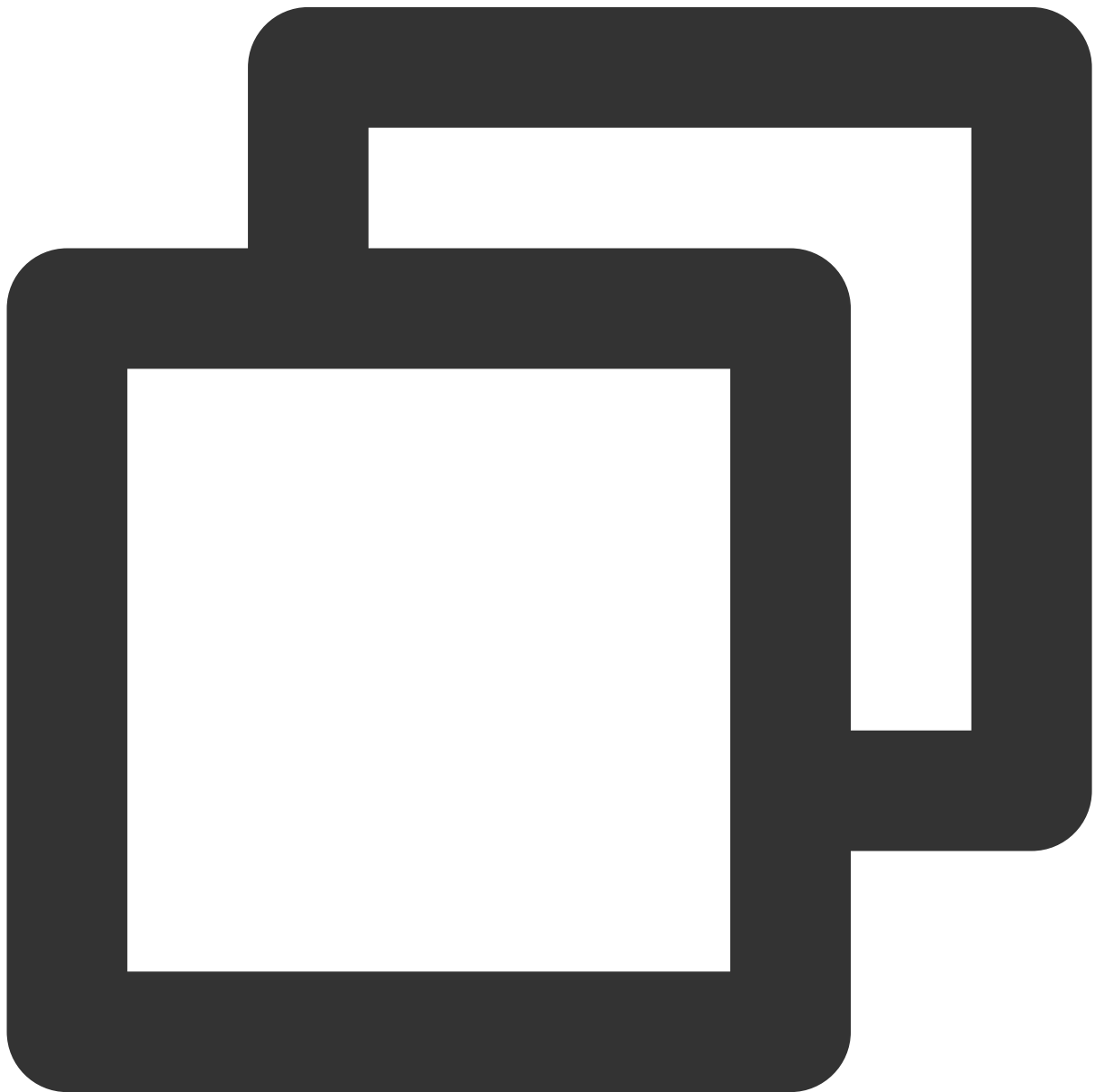
Push subscription



```
// 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 <b>Topic</b> tab on the <b>Cluster</b> page in the console.   |
| "*"        | If the subscription expression is left empty or specified as asterisk (*), all messages are subscribed.<br><code>tag2    tag3</code> 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 <b>Topic</b> tab on the <b>Cluster</b> page in the console.  |
| "*"        | If the subscription expression is left empty or specified as asterisk (*), all messages are subscribed<br><code>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.

**Notes**

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