ALIBABA CLOUD

Alibaba Cloud

Alibaba Cloud Message Queue

Quick Start

Document Version: 20200921

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

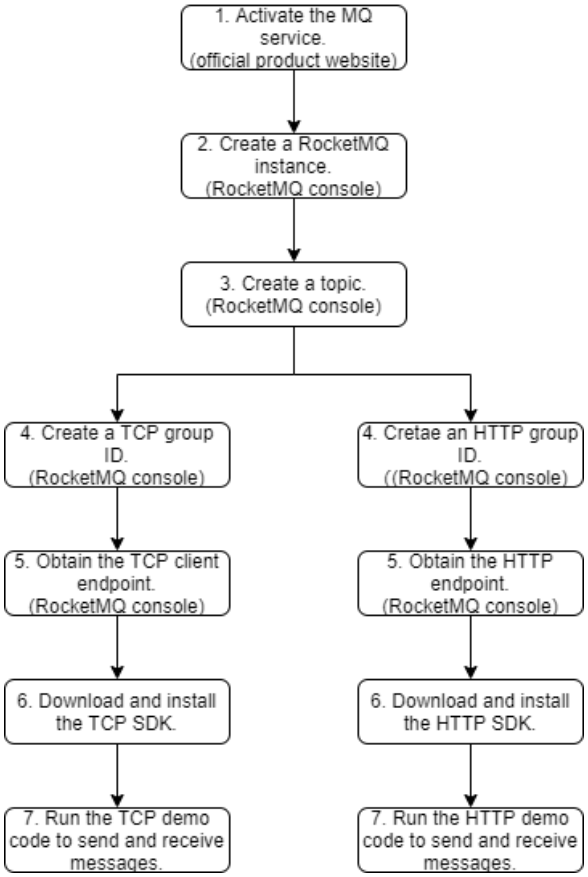
Table of Contents

1.Overview

Message Queue for Apache RocketMQ provides multi-language client SDKs for sending and subscribing to messages over TCP and HTTP. This topic describes how to use the multi-language SDKs to send and subscribe to messages over TCP and HTTP.

Procedure

Depending on the protocol you select, the procedure is as follows.



Instructions

- A TCP client SDK and an HTTP client SDK provided by Message Queue for Apache RocketMQ are different from each other. Therefore, the group IDs of TCP-based instances cannot be used for HTTP-based instances and vice versa.
- If a Message Queue for Apache RocketMQ has a TCP endpoint and an HTTP endpoint, you must obtain a TCP SDK and an HTTP SDK separately to use the endpoints. SDKs of TCP-based instances cannot be used for HTTP-based instances and vice versa.
- If your application uses Message Queue for Apache RocketMQ across domains, we recommend that you use HTTP.

What to do next

- Use TCP SDKs to send and subscribe to messages
- Use HTTP SDKs to send and subscribe to messages

2.Call TCP SDKs to send and subscribe to messages

2.1. Activate Message Queue for Apache RocketMQ

Before you use
Message Queue for Apache RocketMQ
, you must activate it on the Alibaba Cloud official website.

Prerequisites

You have created an Alibaba Cloud account and passed real-name verification.

Procedure

1. Go to the
   Message Queue for Apache RocketMQ
   product details page.
2. On the page that appears, click **Log In** in the upper-right corner.
3. On the **Sign In** page, enter your Alibaba Cloud account and password, and click **Sign In**.
4. On the product details page, click **Buy Now**.
   You are redirected to the
   Message Queue for Apache RocketMQ
   console.
5. In the **Note** dialog box, click **activate Message Queue first >>**.
6. On the **Confirm Order** page, read the order content and service agreement, select **I agree with the Message Queue Service Agreement**, and then click **Activate Now**.

What's next

You can click **Console** to create resources. For more information, see Create resources.

2.2. Create resources

Before you call a TCP SDK to send and subscribe to messages, you must create resources in the Message Queue for Apache RocketMQ
console. You need to enter the resource information when calling the SDK.

Prerequisites

Activate Message Queue for Apache RocketMQ

Context

Message Queue for Apache RocketMQ
provides SDKs for multiple programming languages to send and subscribe to messages over TCP and HTTP. To use
Message Queue for Apache RocketMQ
, you must create an instance, a topic, and a group ID. Due to client differences, a group ID cannot be used for both TCP clients and HTTP clients. Therefore, you must create group IDs for

TCP clients and HTTP clients separately.

Network access restrictions

When you use

Message Queue for Apache RocketMQ

, note the following network access restrictions:

- A topic can be accessed only by a producer or consumer with a group ID created on the same instance and the same region as the topic. For example, when a topic is created on instance A in the **China (Hangzhou)** region, the topic can be accessed only by the producer and consumer with a group ID created on instance A in the **China (Hangzhou)** region.
- If you want to use
  Message Queue for Apache RocketMQ
  over Internet, you can create both the topic and group ID on an instance in the **Internet** region. Producers and consumers can be deployed on a local server or an ECS instance in any region, provided that the local server or ECS instance can access the Internet.
  For more information about regions, see .

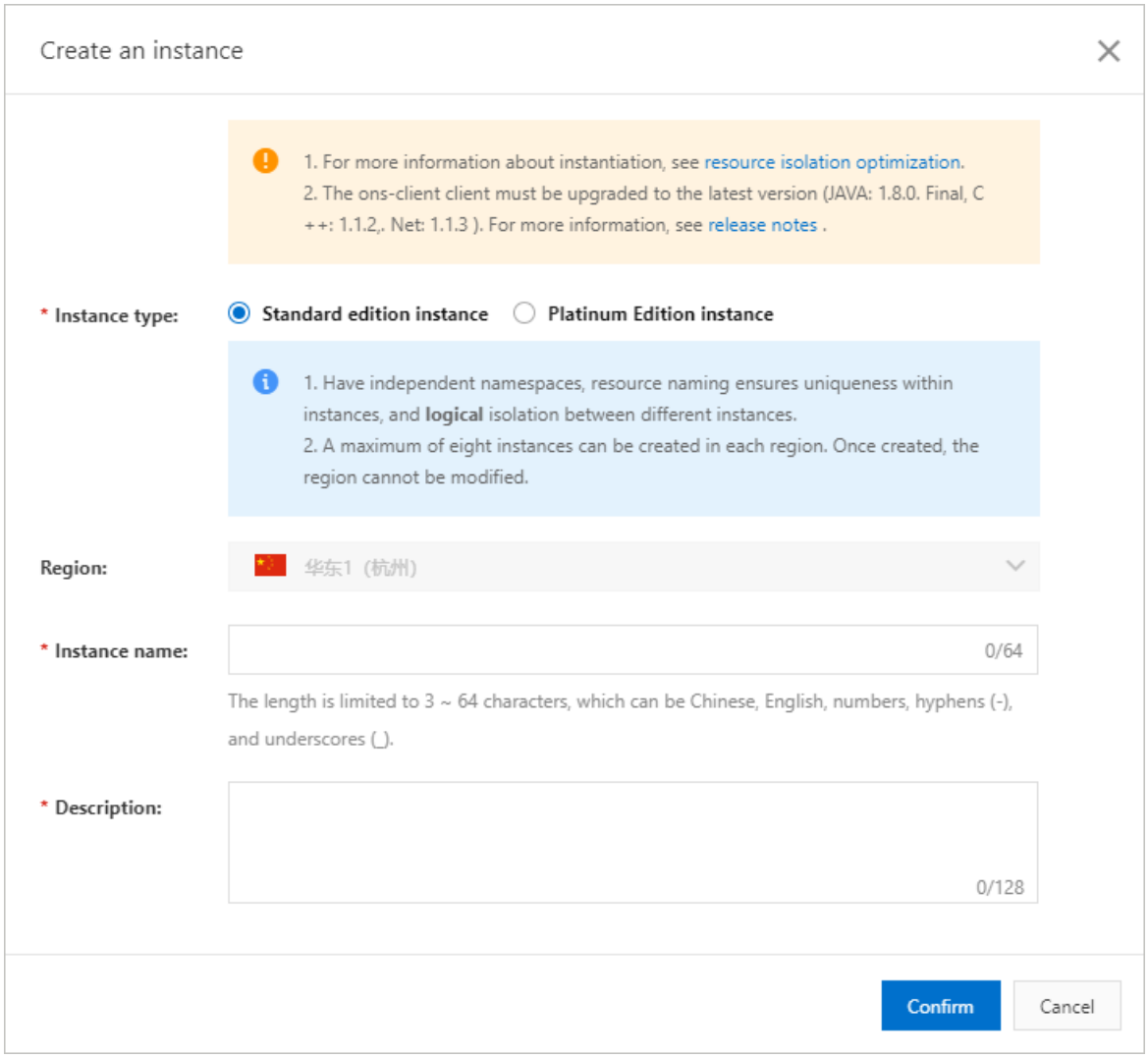
Create an instance

As a virtual machine (VM) resource of

Message Queue for Apache RocketMQ

, an instance stores the topics and group IDs of messages.

1. Log on to the
   Message Queue for Apache RocketMQ
   console.
2. In the top navigation bar, select a region, such as **China (Hangzhou)**.
3. In the left-side navigation pane, click **Instances**.
4. On the **Instances** page, click +.
5. In the **Create Instance** dialog box, set **Instance Type**, **Instance Name**, and **Description**, and click **OK**.



For more information about the billing of Standard Edition and Platinum Edition instances, see .

After the instance is created, it appears on the Instances page.

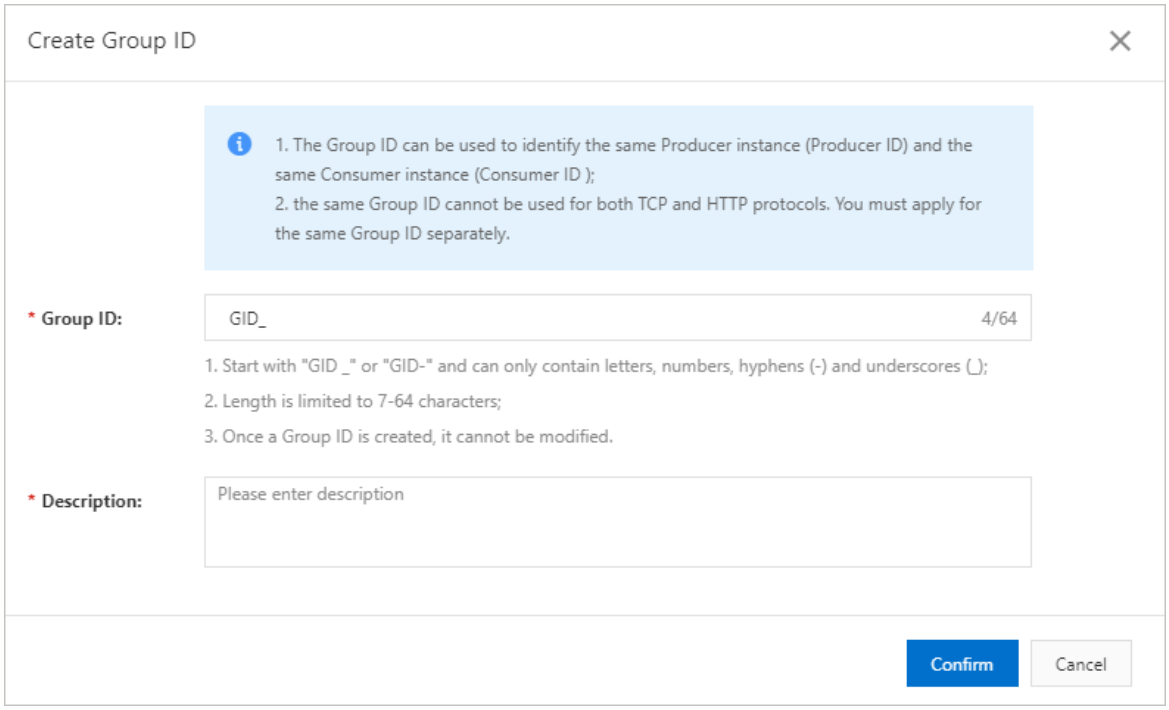
Create a group ID

After you have created an instance and a topic, you need to create a group ID for the message consumer or producer.

- A group ID must be unique within an instance.
- Group IDs and topics implement N:N mapping. A consumer can subscribe to multiple topics and a topic can be subscribed to by multiple consumers. A producer can send messages to multiple topics and a topic can receive messages from multiple producers.

> **Note** A group ID is required for consumers but is optional for producers.

1. On the details page of the target instance, click **Groups** in the left-side navigation pane.
2. On the **Groups** page, choose **TCP > Create Group ID**.
3. In the **Create Group ID** dialog box, set **Group ID** and **Description**, and click **OK**.



After the group ID is created, it appears in the group ID list.

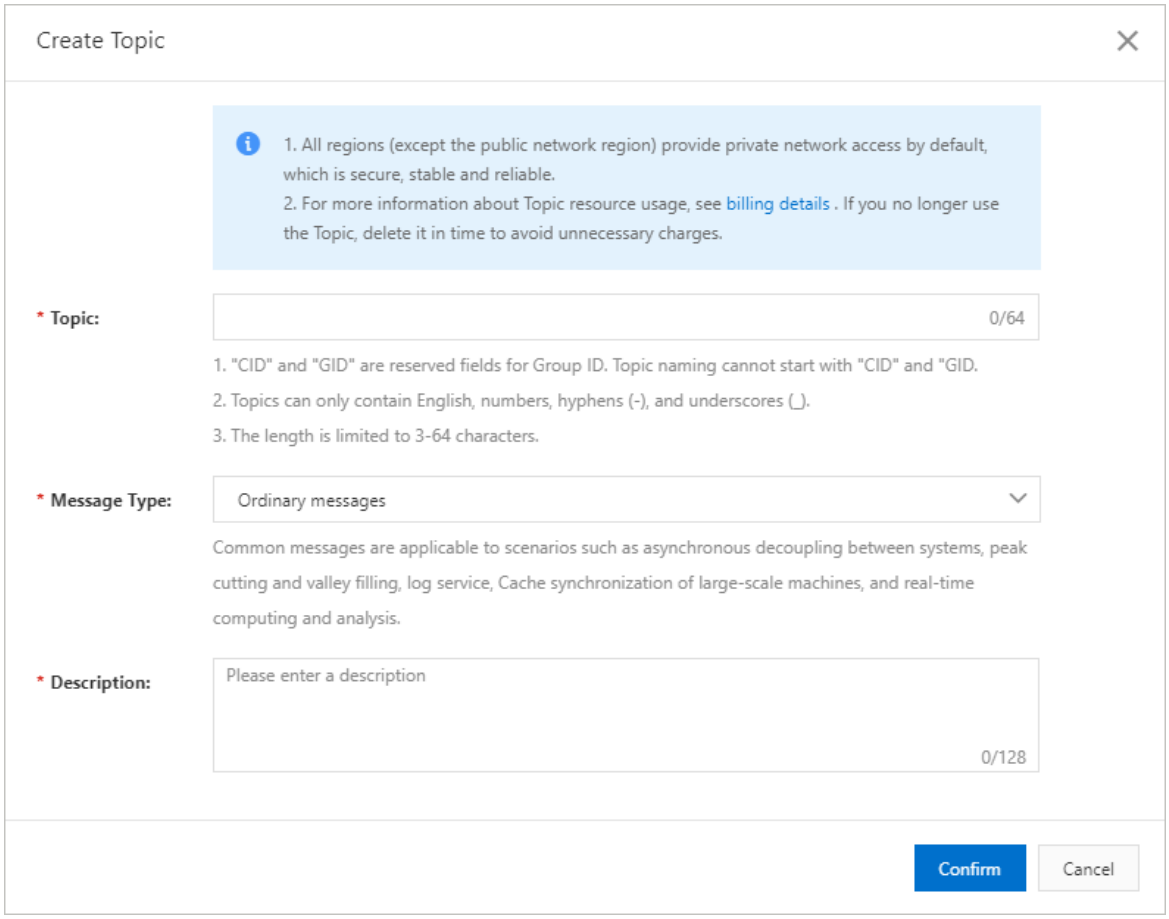
Create a topic

A topic is the first-level identifier for classifying messages in

Message Queue for Apache RocketMQ
. For example, you can create a topic named Topic_Trade for transactional messages. The message producer sends messages to Topic_Trade and the message consumer subscribes to Topic_Trade to consume messages.

- Topics cannot be used across instances. For example, Topic A created in Instance A cannot be used in Instance B.
- A topic name must be unique within an instance.
- We recommend that you create different topics to send different types of messages. For example, create Topic A for normal messages, Topic B for transactional messages, and Topic C for scheduled and delayed messages.

1. On the details page of the target instance, choose **Topics** > **Create Topic** from the left-side navigation pane.
2. In the **Create Topic** dialog box, set **Topic**, select **Normal Message** for **Message Type**, enter **Description**, and click **OK**.

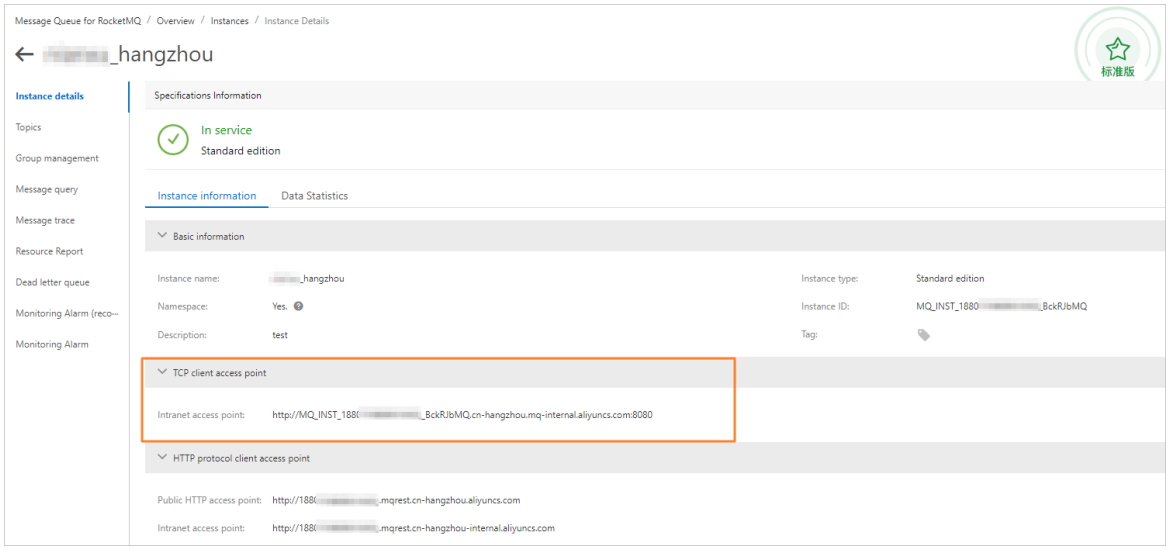


For more information about message types, see Message types.
After the topic is created, it appears in the topic list.

Obtain endpoints

After creating resources in the console, you need to obtain the endpoint of the instance in the console. To access services in an instance or a region when sending or subscribing to messages, you need to configure the endpoints for the producer and consumer.

1. On the details page of the target instance, click **Instance Details** in the left-side navigation pane.
2. On the **Instance Information** tab page that appears by default, find the **TCP Endpoint** section, and click the target TCP endpoint to copy it.



- The endpoint displayed in the console is the endpoint of a specific instance in a region. Different instances in the same region have different endpoints.
- Only the instances in the **Internet** region have public TCP endpoints. Instances in other regions only have internal TCP endpoints.
- TCP endpoints cannot be used across regions. For more information, see Configure the TCP internal endpoint.

> **Note** HTTP and TCP endpoints cannot be used interchangeably.

After completing the preceding preparations, you can run the sample code and use

Message Queue for Apache RocketMQ
to send and subscribe to messages.

What's next

Call TCP SDKs to send and subscribe to messages

2.3. Call TCP SDKs to send and subscribe to messages

After you create all the resources in the console, you can call a TCP SDK to send and subscribe to messages in
Message Queue for Apache RocketMQ
.

Prerequisites

- Create resources
- Create an AccessKey pair

Download and install a TCP SDK

We recommend that you use the following multi-language TCP SDKs provided by

Message Queue for Apache RocketMQ
. Obtain the client SDK in a specific language as needed.

Call TCP SDKs to send messages

After you obtain the client SDK in a specific language, you can run the following sample code to send messages.

On the **Topics** page, find the target topic and click **Send Message** in the **Actions** column.

Java .NET C/C++

Check whether messages are sent

After a message is sent, you can check its sending status in the console by performing the following operations:

1. On the details page of the target instance, choose **Message Query** > **By Topic** from the left-side navigation pane.
2. In the search box, enter the topic to which the message is sent, and click **Search**to check its sending status.
   Stored Atindicates the time when the
   Message Queue for Apache RocketMQ
   broker stores the message. If the message can be queried out, the message has been sent to the Message Queue for Apache RocketMQ broker.

> **Notice** This step demonstrates the situation where
> Message Queue for Apache RocketMQ
> is used for the first time and the consumer has not been started yet. Therefore, no consumption data is displayed in the message status information. To start the consumer and subscribe to messages, go to the next step. For more information about the message status, see Query messages and Query a message trace.

Call TCP SDKs to subscribe to messages

Once a message is sent, you need to start the consumer to subscribe to the message. Run the following sample code as needed to start the consumer and test message subscription. You must set related parameters correctly according to the instructions.

Java .NET C/C++

Check whether the message subscription is successful

After the preceding steps are completed, you can check whether the consumer has been started in the console, that is, whether the message subscription is successful.

1. In the left-side navigation pane, choose **Groups** > **TCP**.
2. Find the target group ID and click **Subscription** in the **Actions** column.
   If the value of **Online** is **Yes** and the value of Subscription Consistency is Yes, the subscription is successful.

3.Call HTTP SDKs to send and subscribe to messages

3.1. Activate Message Queue for Apache RocketMQ

You can use
Message Queue for Apache RocketMQ
after activating it at the Alibaba Cloud website.

Prerequisites

You have created an Alibaba Cloud account and passed real-name verification.

Procedure

1. Go to the
   Message Queue for Apache RocketMQ
   product page.
2. Click **Log In** in the upper-right corner of the page.
3. On the **Logon** page, enter your Alibaba Cloud account and password, and click **Sign In**.
4. On the product page, click **Buy Now**.
   You are redirected to the
   Message Queue for Apache RocketMQ
   console.
5. In the **Note** dialog box, click **Activate Message Queue>>**.
6. On the **Confirm Order** page, select **I have read and agree to the Message Queue Terms of Service**, and then click **Activate Now**.

What's next

You can click **Console** to create resources. For more information, see Create resources.

3.2. Create resources

Before you call an HTTP SDK to send and subscribe to messages, create resources in the
Message Queue for Apache RocketMQ
console.

Prerequisites

Activate Message Queue for Apache RocketMQ

Context

Message Queue for Apache RocketMQ
provides SDKs for multiple programming languages to send and subscribe to messages over TCP and HTTP. To use
Message Queue for Apache RocketMQ
, you must create an instance, a topic, and a group ID. Due to client differences, a group ID cannot be used for both TCP clients and HTTP clients. Therefore, you must create group IDs for

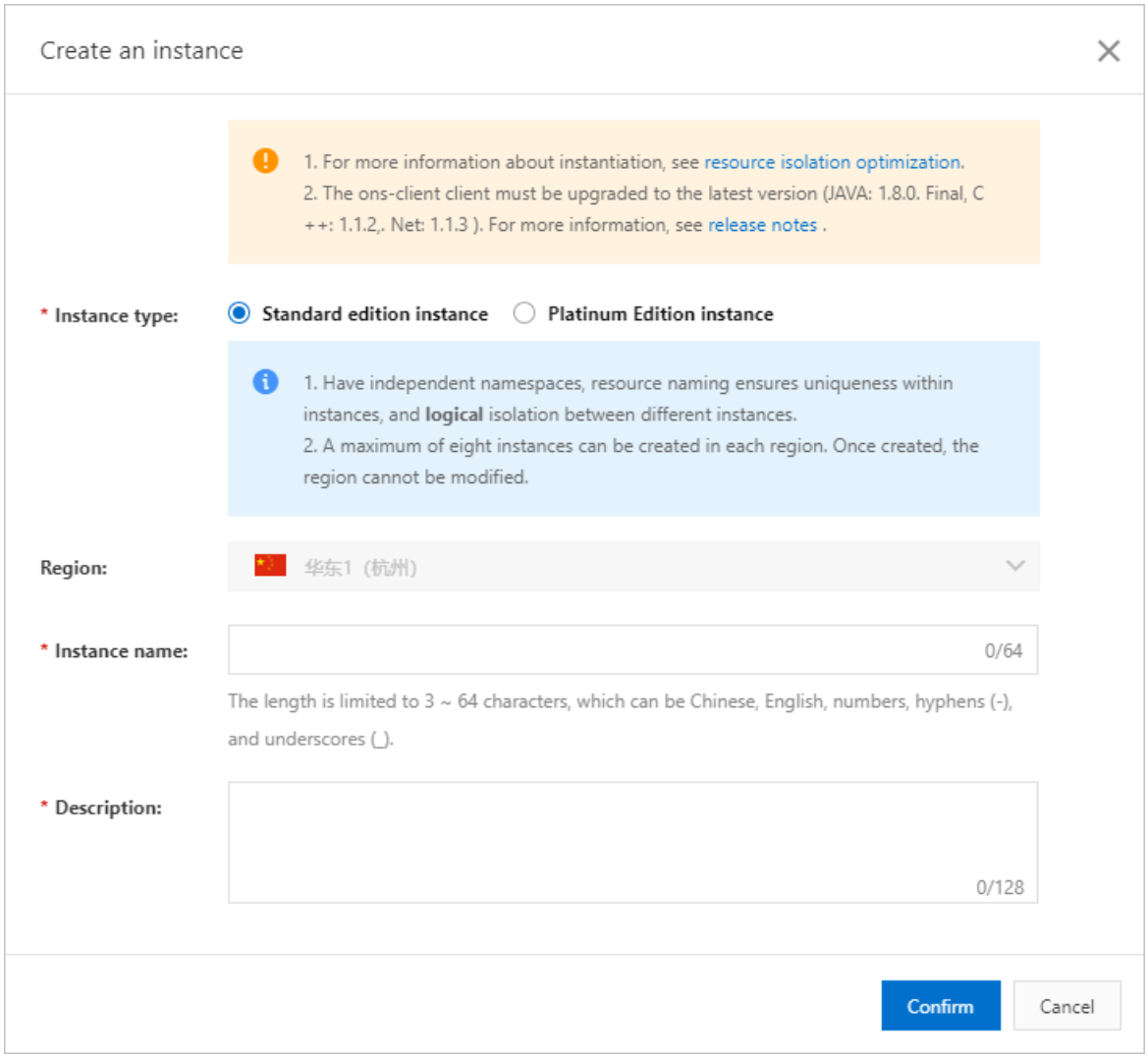
TCP clients and HTTP clients separately.

Create an instance

As a virtual machine (VM) resource of

Message Queue for Apache RocketMQ
, an instance stores the topics and group IDs of messages.

1. Log on to the
   Message Queue for Apache RocketMQ
   console.
2. In the top navigation bar, select a region, such as **China (Hangzhou)**.
3. In the left-side navigation pane, click **Instances**.
4. On the **Instances** page, click ＋.
5. In the **Create Instance** dialog box, set **Instance Type**, **Instance Name**, and **Description**, and click **OK**.



For more information about the billing of Standard Edition and Platinum Edition instances, see .
After the instance is created, it appears on the Instances page.

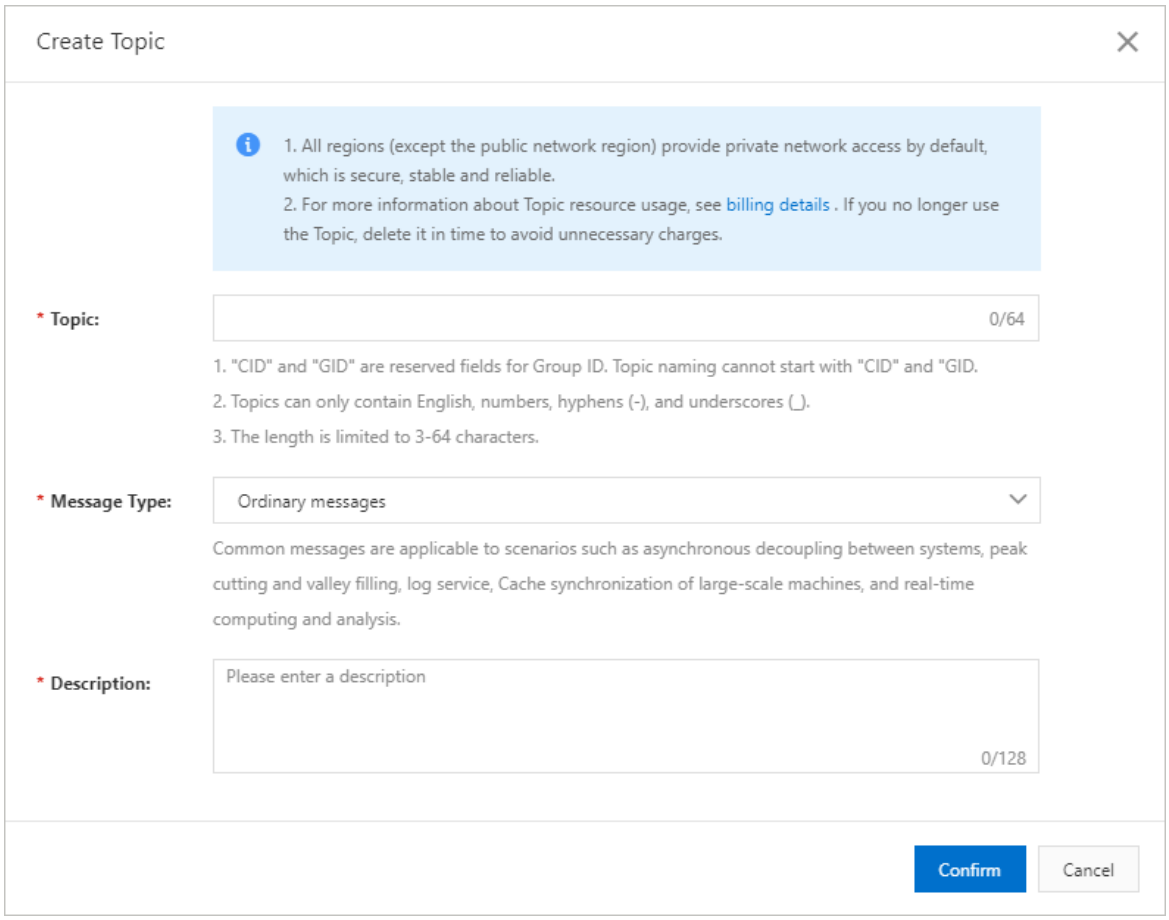
Create a topic

A topic is the first-level identifier for classifying messages in

Message Queue for Apache RocketMQ
. For example, you can create a topic named Topic_Trade for transactional messages. The message producer sends messages to Topic_Trade and the message consumer subscribes to Topic_Trade to consume messages.

- Topics cannot be used across instances. For example, Topic A created in Instance A cannot be used in Instance B.
- A topic name must be unique within an instance.
- We recommend that you create different topics to send different types of messages. For example, create Topic A for normal messages, Topic B for transactional messages, and Topic C for scheduled and delayed messages.

1. On the details page of the target instance, choose **Topics** > **Create Topic** from the left-side navigation pane.
2. In the **Create Topic** dialog box, set **Topic**, select **Normal Message** for **Message Type**, enter **Description**, and click **OK**.



For more information about message types, see Message types.
After the topic is created, it appears in the topic list.

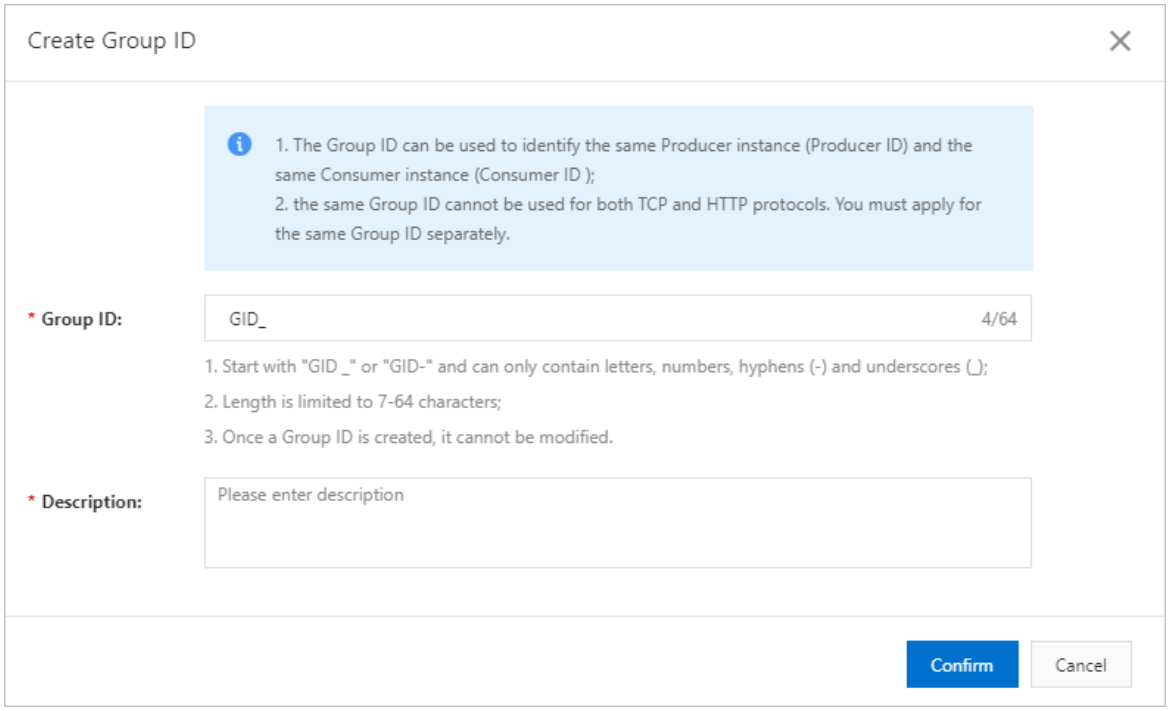
Create a group ID

After you have created an instance and a topic, you need to create a group ID for the message consumer or producer.

- A group ID must be unique within an instance.
- Group IDs and topics implement N:N mapping. A consumer can subscribe to multiple topics and a topic can be subscribed to by multiple consumers. A producer can send messages to multiple topics and a topic can receive messages from multiple producers.

> **Note** A group ID is required for consumers but is optional for producers.

1. On the details page of the target instance, click **Groups** in the left-side navigation pane.
2. On the **Groups** page, choose **HTTP > Create Group ID**.
3. In the **Create Group ID** dialog box, set **Group ID** and **Description**, and click **OK**.

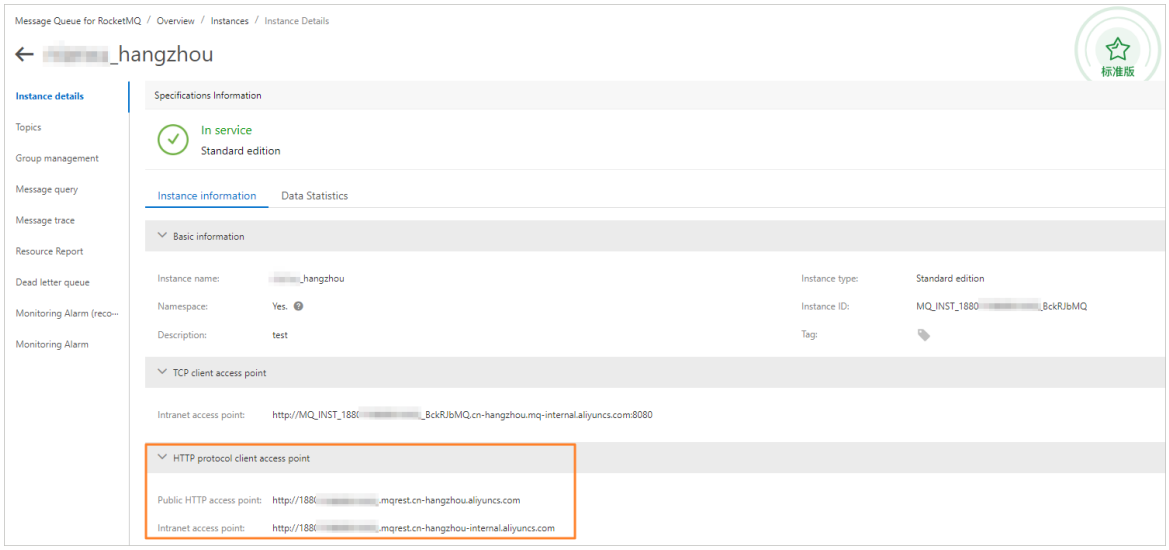


After the group ID is created, it appears in the group ID list.

Obtain endpoints

After creating resources in the console, you need to obtain the endpoint of the instance in the console. To access services in an instance or a region when sending or subscribing to messages, you need to configure the endpoints for the producer and consumer.

1. On the details page of the target instance, click **Instance Details** in the left-side navigation pane.
2. On the **Instance Information** tab that appears by default, go to the **HTTP Endpoint** section and click the endpoint of the target HTTP client to copy it.



The HTTP endpoint displayed in the console is the endpoint of a region, instead of a specific instance. You need to configure an ID for the instance when sending and subscribing to messages.

> **Note** HTTP and TCP endpoints cannot be used interchangeably.

What's next

Call HTTP SDKs to send and subscribe to normal messages

3.3. Call HTTP SDKs to send and subscribe to normal messages

After you create all the resources in the console, you can call an HTTP SDK to send and subscribe to messages in
Message Queue for Apache RocketMQ
.

Prerequisites

- Create resources
- Create an AccessKey pair

Download and install an HTTP SDK

Message Queue for Apache RocketMQ
provides the following multi-language HTTP SDKs. Download and install the client SDK in a specific language as needed.

Java SDK

- Access description
- Release notes

PHP SDK
- Access description
- Release notes

Go SDK
- Access description
- Release notes

Python SDK
- Access description
- Release notes

Node.js SDK
- Access description
- Release notes

C# SDK
- Access description
- Release notes

C++ SDK
- Access description
- Release notes

Call HTTP SDKs to send messages

After you obtain the client SDK in a specific language, you can run the following sample code to send messages.

On the **Topics** page, find the target topic and click **Send Message** in the **Actions** column.

Java | Go | PHP | Python | Node.js | C++ | C#

Check whether messages are sent

After a message is sent, you can check its sending status in the console by performing the following operations:

1. On the details page of the target instance, choose **Message Query** > **By Topic** from the left-side navigation pane.
2. In the search box, enter the topic to which the message is sent, and click **Search**to check its

sending status.
Stored Atindicates the time when the
Message Queue for Apache RocketMQ
broker stores the message. If the message can be queried out, the message has been sent to the Message Queue for Apache RocketMQ broker.

Notice This step demonstrates the situation where Message Queue for Apache RocketMQ is used for the first time and the consumer has not been started yet. Therefore, no consumption data is displayed in the message status information. To start the consumer and subscribe to messages, go to the next step. For more information about the message status, see Query messages and Query a message trace.

Call HTTP SDKs to subscribe to messages

Once a message is sent, you need to start the consumer to subscribe to the message. Run the following sample code as needed to start the consumer and test message subscription. You must set related parameters correctly according to the instructions.

Java | Go | PHP | Python | Node.js | C++ | C#

```
import com.aliyun.mq.http.MQClient;
import com.aliyun.mq.http.MQConsumer;
import com.aliyun.mq.http.common.AckMessageException;
import com.aliyun.mq.http.model.Message;

import java.util.ArrayList;
import java.util.List;

public class Consumer {

  public static void main(String[] args) {
    MQClient mqClient = new MQClient(
        // The domain name used for HTTP access. (The production environment hosted on Alibaba Cloud is used here as an example.)
        "${HTTP_ENDPOINT}",
        // The AccessKey ID that you created in the Alibaba Cloud console for verifying the Alibaba Cloud account.
        "${ACCESS_KEY}",
        // The AccessKey secret that you created in the Alibaba Cloud console for verifying the Alibaba Cloud account.
        "${SECRET_KEY}"
    );

    // The topic of the message.
    final String topic = "${TOPIC}";
    // The group ID (consumer ID) you created in the console.
    final String groupId = "${GROUP_ID}";
    // The instance ID of the topic. This parameter is null by default.
    final String instanceId = "${INSTANCE_ID}";

    final MQConsumer consumer;
    if (instanceId != null && instanceId != "") {
      consumer = mqClient.getConsumer(instanceId, topic, groupId, null);
    }else {
      consumer = mqClient.getConsumer(topic, groupId);
    }

    // Consume messages cyclically in the current thread. We recommend that you use multiple
```

```
threads to concurrently consume messages.
    do {
      List<Message> messages = null;

      try {
        // Consume messages in long-polling mode.
        // In long-polling mode, if no message on the topic is available for consumption, the
request is hung on the server for 3 seconds. If any message is available for consumption within
the duration, a response is immediately sent to the client.
        messages = consumer.consumeMessage(
            3,// A maximum of 3 messages can be consumed at a time. The largest value is 16.
            3// The duration of a long-polling cycle is 3 seconds. The largest value is 30 seconds.
        );
      }catch (Throwable e) {
        e.printStackTrace();
        try {
          Thread.sleep(2000);
        }catch (InterruptedException e1) {
          e1.printStackTrace();
        }
      }
      // No messages.
      if (messages == null || messages.isEmpty()) {
        System.out.println(Thread.currentThread().getName() + ": no new message, continue!
");
        continue;
      }

      // Data consumption logic.
      for (Message message : messages) {
        System.out.println("Receive message: " + message);
      }

      // If the consumption of a message is not confirmed before Message.nextConsumeTime,
the message will be consumed again.
      // A unique timestamp is specified for the handle of a message each time the message is
consumed.
      {
        List<String> handles = new ArrayList<String>();
        for (Message message : messages) {
          handles.add(message.getReceiptHandle());
        }

        try {
          consumer.ackMessage(handles);
        }catch (Throwable e) {
          // The consumption confirmation of some messages may fail due to timeout of the
message handles.
          if (e instanceof AckMessageException) {
            AckMessageException errors = (AckMessageException) e;
            System.out.println("Ack message fail, requestId is:" + errors.getRequestId() + ", fail
handles:");
            if (errors.getErrorMessages() ! = null) {
              for (String errorHandle :errors.getErrorMessages().keySet()) {
                System.out.println("Handle:" + errorHandle + ", ErrorCode:" +
errors.getErrorMessages().get(errorHandle).getErrorCode()
                    + ", ErrorMsg:" +
errors.getErrorMessages().get(errorHandle).getErrorMessage());
              }
            }
            continue;
          }
          e.printStackTrace();
```

```
            }
        }
    }while (true);
  }
}
```