

TDMQ for RabbitMQ

Quick Start



Copyright Notice

©2013–2024 Tencent Cloud. All rights reserved.

The complete copyright of this document, including all text, data, images, and other content, is solely and exclusively owned by Tencent Cloud Computing (Beijing) Co., Ltd. ("Tencent Cloud"); Without prior explicit written permission from Tencent Cloud, no entity shall reproduce, modify, use, plagiarize, or disseminate the entire or partial content of this document in any form. Such actions constitute an infringement of Tencent Cloud's copyright, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

Trademark Notice



This trademark and its related service trademarks are owned by Tencent Cloud Computing (Beijing) Co., Ltd. and its affiliated companies("Tencent Cloud"). The trademarks of third parties mentioned in this document are the property of their respective owners under the applicable laws. Without the written permission of Tencent Cloud and the relevant trademark rights owners, no entity shall use, reproduce, modify, disseminate, or copy the trademarks as mentioned above in any way. Any such actions will constitute an infringement of Tencent Cloud's and the relevant owners' trademark rights, and Tencent Cloud will take legal measures to pursue liability under the applicable laws.

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Quick Start

Using SDK to Send/Receive Message

Quick Start

Using SDK to Send/Receive Message

Last updated: 2024-08-02 12:16:20

Operation scenarios

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

Prerequisites

- [You have created the required resources.](#)
- [You have installed JDK 1.8 or later.](#)
- [You have installed Maven 2.5 or later.](#)
- [Download the demo](#)

Operation step

Step 1. Install the Java dependency library

Add the following dependencies to pom.xml:

```
<!-- in your <dependencies> block -->
<dependency>
  <groupId>com.rabbitmq</groupId>
  <artifactId>amqp-client</artifactId>
  <version>5.13.0</version>
</dependency>
```

Step 2. Produce a message

Compile and run MessageProducer.java.

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.tencent.tdmq.demo.cloud.Constant;
```

```
/**
 * Message producer
 */
public class MessageProducer {

/**
 * Exchange name
 */
    private static final String EXCHANGE_NAME = "exchange_name";

    public static void main(String[] args) throws Exception {
        // Connection factory
        ConnectionFactory factory = new ConnectionFactory();
        // Set the service address (replace with the access point address
        copied in the console)
        factory.setUri("amqp://***");
        // Set virtual hosts (copy the complete vhost name from the open-
        source RabbitMQ console)
        factory.setVirtualHost(VHOST_NAME);
        // Set the username (user name in the permission configuration of
        the vhost in the open-source RabbitMQ console)
        factory.setUsername(USERNAME);
        // Set the password (key for the corresponding user)
        factory.setPassword("*****");
        // Get the connection address and establish the channel
        try (Connection connection = factory.newConnection(); Channel
        channel = connection.createChannel()) {
            // Bind the message exchange (EXCHANGE_NAME must exist in the
            TDMQ for RabbitMQ console, and the exchange type must be the same as
            that in the console)
            channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
            for (int i = 0; i < 10; i++) {
                String message = "this is rabbitmq message " + i;
                // Publish a message to the exchange, which will
                automatically deliver the message to the corresponding queue
                channel.basicPublish(EXCHANGE_NAME, "", null,
                message.getBytes());
                System.out.println(" [producer] Sent '" + message + "'");
            }
        } catch (Exception e) {
```

```
e.printStackTrace();  
  
}  
  
}  
  
}
```

P a r a m e t e r	Description										
E X C H A N G E - N A M E	Exchange name, which can be obtained from the exchange list in the console.										
f a c t o r y - s e t U r i	<div>Cluster access address, obtained from the Client Access section on the cluster basic information page.</div> <div><div>客户端接入 网络信息 Web 控制台访问地址 用 Prometheus 监控实例</div><div>客户端接入 ①添加路</div><table><tr><th>接入类型</th><th>访问策略</th><th>公网带宽</th><th>网络</th><th>操作</th></tr><tr><td>VPC 网络</td><td>-</td><td>-</td><td>vpc-n subnet- amqp:// 2</td><td>删除</td></tr></table></div>	接入类型	访问策略	公网带宽	网络	操作	VPC 网络	-	-	vpc-n subnet- amqp:// 2	删除
接入类型	访问策略	公网带宽	网络	操作							
VPC 网络	-	-	vpc-n subnet- amqp:// 2	删除							
f a c	Vhost name, obtained from the Vhost list in the console.										

t o r y. s e t V i r t u a l H o s t	
f a c t o r y. s e t U s e r n a m e	Username, as created in the console.
f a c t o r y. s e t P	User password, as set when creating the user in the console.

Step 3. Consume the message

Compile and run MessageConsumer.java.

```
import com.rabbitmq.client.AMQP;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.DefaultConsumer;
import com.rabbitmq.client.Envelope;
import com.tencent.tdmq.demo.cloud.Constant;

import java.io.IOException;
import java.nio.charset.StandardCharsets;

/**
 * Message consumer
 */
public class MessageConsumer1 {

    /**
     * Queue name
     */
    public static final String QUEUE_NAME = "queue_name";

    /**
     * Exchange name
     */
    private static final String EXCHANGE_NAME = "exchange_name";

    public static void main(String[] args) throws Exception {
        // Connection factory
        ConnectionFactory factory = new ConnectionFactory();
```



```
// Set the service address (replace with the access point address
// copied in the console)
factory.setUri("amqp://***");
// Set virtual hosts (copy the complete vhost name from the open-
// source RabbitMQ console)
factory.setVirtualHost(VHOST_NAME);
// Set the username (user name in the permission configuration of
// the vhost in the open-source RabbitMQ console)
factory.setUsername(USERNAME);
// Set the password (key for the corresponding user)
factory.setPassword("****");
// Get the connection address
Connection connection = factory.newConnection();
// Establish a channel
Channel channel = connection.createChannel();
// Bind the message exchange
channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
// Declare the queue message
channel.queueDeclare(QueueName, true, false, false, null);
// Bind the message exchange (EXCHANGE_NAME must exist in the
// TDMQ for RabbitMQ console, and the exchange type must be the same as
// that in the console)
channel.queueBind(QueueName, EXCHANGE_NAME, "");
System.out.println(" [Consumer1] Waiting for messages.");
// Subscribe to the message
channel.basicConsume(QueueName, false, "ConsumerTag", new
DefaultConsumer(channel) {
    @Override
    public void handleDelivery(String consumerTag, Envelope
envelope,
                                AMQP.BasicProperties properties,
byte[] body)
        throws IOException {
        // Process received messages with business logic.
        System.out.println("Received: " + new String(body,
StandardCharsets.UTF_8) + ", deliveryTag: " + envelope.getDeliveryTag()
+ ", messageId: " + properties.getMessageId());
        channel.basicAck(envelope.getDeliveryTag(), false);
    }
});
}
```

P a r a m e t e r	Description								
Q U E U E - N A M E	Queue name, which can be obtained from the queue list in the console.								
E X C H A N G E - N A M E	Exchange name, which can be obtained from the exchange list in the console.								
f a c t o r y s e t	<div>Cluster access address, obtained from the Client Access section on the cluster basic info page.</div> <div><div><div>客户端接入</div><div>网络信息</div><div>Web 控制台访问地址</div><div>用 Prometheus 监控实例</div></div><div><div>客户端接入 ⓘ</div><table><tr><td>接入类型</td><td>访问策略</td><td>公网带宽</td><td>网络</td></tr><tr><td>VPC 网络</td><td>-</td><td>-</td><td>vpc-n subnet- amqp:// 2</td></tr></table></div></div>	接入类型	访问策略	公网带宽	网络	VPC 网络	-	-	vpc-n subnet- amqp:// 2
接入类型	访问策略	公网带宽	网络						
VPC 网络	-	-	vpc-n subnet- amqp:// 2						

U r i	
f a c t o r y. s e t V i r t u a l H o s t	Vhost name, obtained from the Vhost list in the console.
f a c t o r y. s e t U s e r n a m e	Username, as created in the console.
f a c t o	User password, as set when creating the user in the console.

r
y.
s
e
t
P
a
s
s
w
o
r
d