

# Distributed Message Service for RabbitMQ

## User Guide

**Issue** 01  
**Date** 2024-11-27



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

---

# Contents

---

|   |           |
|---|-----------|
| <b>1 Process of Using RabbitMQ.....</b>                                 | <b>1</b>  |
| <b>2 Permissions Management.....</b>                                    | <b>3</b>  |
| 2.1 Creating an IAM User and Granting DMS for RabbitMQ Permissions..... | 3         |
| <b>3 Buying a RabbitMQ Instance.....</b>                                | <b>7</b>  |
| <b>4 Configuring Virtual Hosts.....</b>                                 | <b>12</b> |
| 4.1 Creating a RabbitMQ Virtual Host.....                               | 12        |
| 4.2 Creating a RabbitMQ Exchange.....                                   | 14        |
| 4.3 Binding a RabbitMQ Exchange.....                                    | 16        |
| 4.4 Creating a RabbitMQ Queue.....                                      | 17        |
| 4.5 Binding a RabbitMQ Queue.....                                       | 19        |
| 4.6 Managing RabbitMQ Virtual Hosts.....                                | 20        |
| 4.6.1 Viewing a RabbitMQ Virtual Host.....                              | 20        |
| 4.6.2 Deleting RabbitMQ Virtual Hosts.....                              | 21        |
| 4.7 Managing RabbitMQ Exchanges.....                                    | 23        |
| 4.7.1 Unbinding a RabbitMQ Exchange.....                                | 23        |
| 4.7.2 Deleting RabbitMQ Exchanges.....                                  | 23        |
| 4.8 Managing RabbitMQ Queues.....                                       | 24        |
| 4.8.1 Clearing Messages in a RabbitMQ Queue.....                        | 24        |
| 4.8.2 Unbinding a RabbitMQ Queue.....                                   | 26        |
| 4.8.3 Configuring Queue Mirroring.....                                  | 27        |
| 4.8.4 Configuring Lazy Queues.....                                      | 29        |
| 4.8.5 Configuring RabbitMQ Quorum Queues.....                           | 30        |
| 4.8.6 Configuring a Single Active Consumer.....                         | 35        |
| 4.8.7 Deleting RabbitMQ Queues.....                                     | 37        |
| <b>5 Accessing a RabbitMQ Instance.....</b>                             | <b>41</b> |
| 5.1 Configuring RabbitMQ Network Connections.....                       | 41        |
| 5.1.1 RabbitMQ Network Connection Requirements.....                     | 41        |
| 5.1.2 Configuring RabbitMQ Public Access.....                           | 42        |
| 5.2 Configuring Heartbeats on the RabbitMQ Client.....                  | 44        |
| 5.3 Accessing RabbitMQ on a Client (SSL Disabled).....                  | 46        |
| 5.4 Accessing RabbitMQ on a Client (SSL Enabled).....                   | 48        |

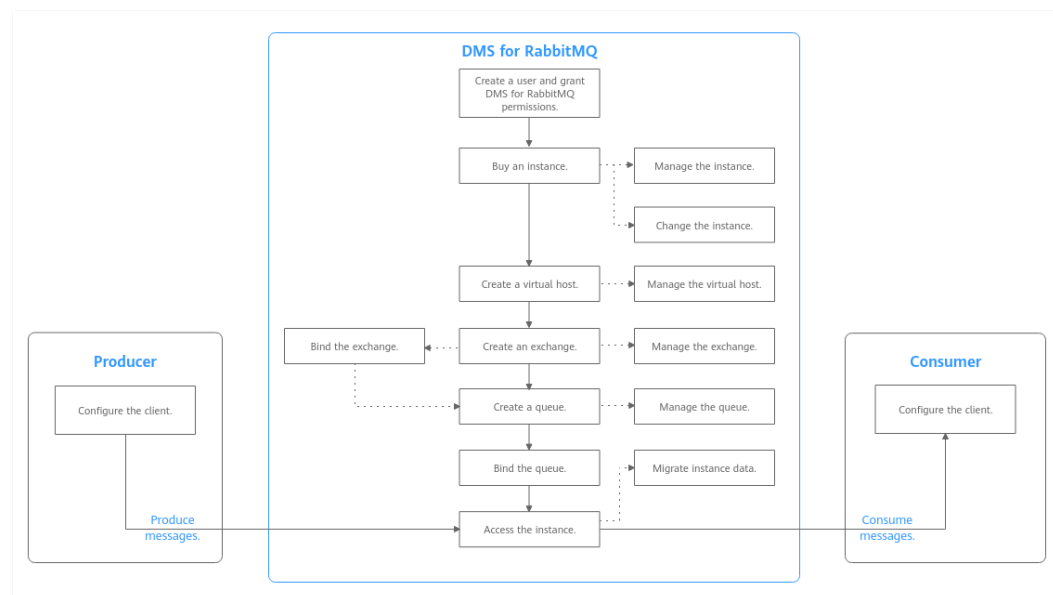
|  |           |
|--|-----------|
| <b>6 Managing Messages</b>   | <b>51</b> |
| 6.1 Configuring RabbitMQ Dead Letter Messages                      | 51        |
| 6.2 Configuring RabbitMQ Message Acknowledgment                    | 52        |
| 6.3 Configuring RabbitMQ Message Prefetch                          | 53        |
| <b>7 Advanced Features</b>   | <b>55</b> |
| 7.1 Configuring RabbitMQ Persistence                               | 55        |
| 7.2 Configuring RabbitMQ TTL                                       | 58        |
| <b>8 Managing Instances</b>  | <b>60</b> |
| 8.1 Viewing and Modifying Basic Information of a RabbitMQ Instance | 60        |
| 8.2 Viewing RabbitMQ Client Connection Addresses                   | 63        |
| 8.3 Managing RabbitMQ Instance Tags                                | 64        |
| 8.4 Resetting the RabbitMQ Instance Password                       | 65        |
| 8.5 Enabling RabbitMQ Plug-ins                                     | 66        |
| 8.6 Using the rabbitmq_tracing Plug-in                             | 68        |
| 8.7 Exporting the RabbitMQ Instance List                           | 73        |
| 8.8 Restarting a RabbitMQ Instance                                 | 74        |
| 8.9 Deleting a RabbitMQ Instance                                   | 75        |
| 8.10 Logging In to RabbitMQ Management UI                          | 76        |
| <b>9 Modifying RabbitMQ Instance Specifications</b>                | <b>78</b> |
| <b>10 Migrating RabbitMQ Services</b>                              | <b>80</b> |
| <b>11 Applying for Increasing RabbitMQ Quotas</b>                  | <b>85</b> |
| <b>12 Viewing Metrics and Configuring Alarms</b>                   | <b>87</b> |
| 12.1 Viewing RabbitMQ Metrics                                      | 87        |
| 12.2 RabbitMQ Metrics  | 88        |
| 12.3 Configuring RabbitMQ Alarms                                   | 92        |
| <b>13 Viewing RabbitMQ Audit Logs</b>                              | <b>97</b> |

# 1 Process of Using RabbitMQ

Based on the open-source RabbitMQ, Distributed Message Service (DMS) for RabbitMQ provides messaging services with rich messaging features, flexible routing, alarms, monitoring, and high availability functions. It is applicable to flash sales, flow control, and system decoupling scenarios.

The following figure shows the process of using a RabbitMQ instance to produce and consume messages.

**Figure 1-1** Process of using RabbitMQ



- 1. Creating an IAM User and Granting DMS for RabbitMQ Permissions**  
Create IAM users and grant them only the DMS for RabbitMQ permissions required to perform a given task based on their job responsibilities.
- 2. Buying a RabbitMQ Instance**  
RabbitMQ instances are tenant-exclusive, and physically isolated in deployment.
- 3. Creating a RabbitMQ Virtual Host**

To connect a producer or consumer to a RabbitMQ instance, you must specify a virtual host.

4. **Creating a RabbitMQ Exchange**

Producers send messages to exchanges first, rather than directly to queues. Exchanges route messages to queues based on routing keys.

5. **Creating a RabbitMQ Queue**

Queues store messages. Each message is sent to one or multiple queues.

6. **Binding a RabbitMQ Queue**

Exchanges route messages to queues based on routing keys.

7. **Accessing a RabbitMQ Instance**

The client access RabbitMQ instances over a private or public network, and produces and consumes messages.

# 2 Permissions Management

---

## 2.1 Creating an IAM User and Granting DMS for RabbitMQ Permissions

Use [Identity and Access Management \(IAM\)](#) to implement fine-grained permissions control over your Distributed Message Service (DMS) for RabbitMQ resources. With IAM, you can:

- Create IAM users for personnel based on your enterprise's organizational structure. Each IAM user has their own identity credentials for accessing DMS for RabbitMQ resources.
- Grant users only the permissions required to perform a given task based on their job responsibilities.
- Entrust a HUAWEI ID or a cloud service to perform efficient O&M on your DMS for RabbitMQ resources.

If your HUAWEI ID meets your permissions requirements, you can skip this section.

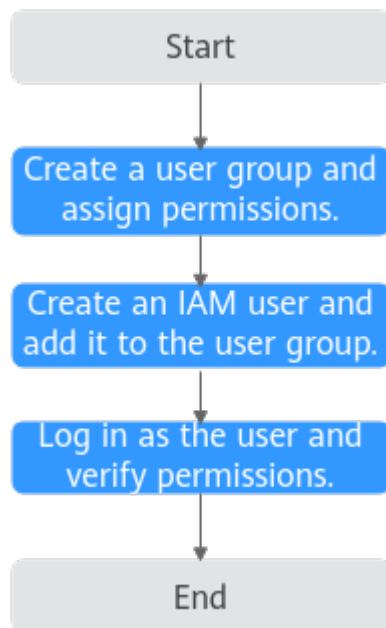
This section describes the procedure for granting user permissions. [Figure 2-1](#) shows the process flow.

### Prerequisites

Learn about the permissions (see [System-defined roles and policies supported by DMS for RabbitMQ](#)) supported by DMS for RabbitMQ and choose policies according to your requirements. For the system policies of other services, see [System Permissions](#).

## Process Flow

Figure 2-1 Process of granting DMS for RabbitMQ permissions



1. For the following example, **create a user group on the IAM console**, and assign the **DMS ReadOnlyAccess** policy to the group.
2. **Create an IAM user.**
3. **Log in as the IAM user** and verify permissions.  
In the authorized region, perform the following operations:
  - Choose **Service List > Distributed Message Service (for RabbitMQ)**. Then click **Buy Instance** on the console of DMS for RabbitMQ. If a message appears indicating that you have insufficient permissions to perform the operation, the **DMS ReadOnlyAccess** policy is in effect.
  - Choose **Service List > Elastic Volume Service**. If a message appears indicating that you have insufficient permissions to access the service, the **DMS ReadOnlyAccess** policy is in effect.
  - Choose **Service List > Distributed Message Service (for RabbitMQ)**. The RabbitMQ console is displayed. If a list of RabbitMQ instances are displayed, the **DMS ReadOnlyAccess** policy is in effect.

## Example Custom Policies

You can create custom policies to supplement the system-defined policies of DMS for RabbitMQ. For details about actions supported in custom policies, see [Permissions and Supported Actions](#).

To create a custom policy, choose either visual editor or JSON.

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Create a JSON policy or edit an existing one.



For details, see [Creating a Custom Policy](#). The following lists examples of common DMS for RabbitMQ custom policies.

 **NOTE**

DMS for RabbitMQ permissions policies are based on DMS. Therefore, when assigning permissions, select DMS permissions policies.

- Example 1: Grant permission to create and delete instances.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dms:instance:create",
        "dms:instance:delete"
      ]
    }
  ]
}
```

- Example 2: Grant permission to deny instance deletion.

A policy with only "Deny" permissions must be used together with other policies. If the permissions granted to an IAM user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

Assume that you want to grant the permissions of the **DMS FullAccess** policy to a user but want to prevent them from deleting instances. You can create a custom policy for denying instance deletion, and attach this policy together with the **DMS FullAccess** policy to the user. As an explicit deny in any policy overrides any allows, the user can perform all operations on DMS for RabbitMQ excepting deleting instances.

Example policy denying instance deletion:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "dms:instance:delete"
      ]
    }
  ]
}
```

## DMS for RabbitMQ Resources

A resource is an object that exists within a service. DMS for RabbitMQ resources include **rabbitmq**. To select these resources, specify their paths.

**Table 2-1** DMS for RabbitMQ resources and their paths

| Resource | Resource Name | Path   |
|----------|---------------|--|
| rabbitmq | Instance      | <p>[Format]<br/>DMS:*:*: rabbitmq: <i>instance ID</i></p> <p>[Note]<br/>For instance resources, IAM automatically generates the prefix (<b>DMS:*:*:rabbitmq:</b>) of the resource path.<br/>For the path of a specific instance, add the <i>instance ID</i> to the end. You can also use an asterisk * to indicate any instance. For example:<br/><b>DMS:*:*:rabbitmq:*</b> indicates any RabbitMQ instance.</p> |

## DMS for RabbitMQ Request Conditions

Request conditions are useful in determining when a custom policy is in effect. A request condition consists of condition keys and operators. Condition keys are either global or service-level and are used in the Condition element of a policy statement. **Global condition keys** (starting with **g:**) are available for operations of all services, while service-specific condition keys (starting with a service name such as **dms:**) are available only for operations of specific services. An operator must be used together with a condition key to form a complete condition statement.

DMS for RabbitMQ has a group of predefined condition keys that can be used in IAM. For example, to define an "Allow" permission, use the condition key **dms:ssl** to filter instances by SSL configurations. The following table lists the DMS for RabbitMQ predefined condition keys.

**Table 2-2** Predefined condition keys of DMS for RabbitMQ

| Condition Key | Operator | Description                      |
|---------------|----------|----------------------------------|
| dms:publicIP  | Bool     | Whether public access is enabled |
| dms:ssl       | Bool     | Whether SSL is enabled           |

# 3 Buying a RabbitMQ Instance

RabbitMQ is an open-source service using the advanced message queuing protocol (AMQP). RabbitMQ stores and forwards messages in a distributed system.

RabbitMQ instances are tenant-exclusive, and physically isolated in deployment. You can customize the computing capabilities and storage space of a RabbitMQ instance as required.

## Preparing Instance Dependencies

Dependency resources listed in [Table 3-1](#) have been prepared.

**Table 3-1** RabbitMQ instance dependencies

| Resource Name  | Requirement   | Reference  |
|----------------|---|--|
| VPC and subnet | You need to configure a VPC and subnet for the RabbitMQ instance as required. You can use the current account's existing VPC and subnet, or create new ones.<br>Note: VPCs must be created in the same region as the RabbitMQ instance. | For details on how to create a VPC and subnet, see <a href="#">Creating a VPC</a> . If you need to create and use a new subnet in an existing VPC, see <a href="#">Creating a Subnet for the VPC</a> . |
| Security group | Different RabbitMQ instances can use the same or different security groups.<br>Before accessing a RabbitMQ instance, configure security groups based on the access mode. For details, see <a href="#">Table 5-2</a> .                   | For details on how to create a security group, see <a href="#">Creating a Security Group</a> . For details on how to add rules to a security group, see <a href="#">Adding a Security Group Rule</a> . |

| Resource Name | Requirement   | Reference  |
|---------------|---|--|
| EIP           | <p>To access a RabbitMQ instance on a client over a public network, create EIPs in advance.</p> <p>Note the following when creating EIPs:</p> <ul style="list-style-type: none"> <li>The EIPs must be created in the same region as the RabbitMQ instance.</li> <li><b>The RabbitMQ console cannot identify IPv6 EIPs.</b></li> </ul> | <p>For details about how to create an EIP, see <a href="#">Assigning an EIP</a>.</p> |

## Buying a RabbitMQ Instance

**Step 1** Go to the [Buy Instance page](#).

**Step 2** Specify **Billing Mode**.

**Step 3** Select a region.

DMS for RabbitMQ instances in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access.

**Step 4** Select a **Project**.

Projects isolate compute, storage, and network resources across geographical regions. For each region, a preset project is available.

**Step 5** Select an **AZ**.

An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.

Select one, three, or more AZs as required. The AZs cannot be changed once the instance is created.

**Step 6** Enter an **Instance Name**.

You can customize a name that complies with the rules: 4–64 characters; starts with a letter; can contain only letters, digits, hyphens (-), and underscores (\_).

**Step 7** Select an **Enterprise Project**.

This parameter is for enterprise users. An enterprise project manages cloud resources. The enterprise project management service unifies cloud resources in projects, and resources and members in a project. The default project is **default**.

**Step 8** Configure the following instance parameters:

- Version:** Currently, only **3.8.35** is supported.
- Architecture:** Select **Single-node** or **Cluster**.

- **Single-node:** There is only one RabbitMQ broker.
  - **Cluster:** There are multiple RabbitMQ brokers, achieving highly reliable message storage.
3. **CPU Architecture:** Retain the default value.
  4. **Broker Flavor:** Select a flavor as required. Learn more about [Specifications](#).

 **NOTE**

To ensure service stability and reliability, DMS for RabbitMQ sets the default memory high watermark to 40%. Publishers will be blocked if the memory usage exceeds 40%. To avoid reaching the high watermark, retrieve messages stacked in the queue in time.

5. **Brokers:** Select the required number of brokers.
6. **Storage space per broker:** Select the disk type and size.  
For details about how to select a disk type, see [Disk Types and Performance](#).
  - For a single-node instance, the value range is 100–30,000 GB.
  - For a cluster instance, the value range is Number of brokers × 100 GB to Number of brokers × 30,000 GB.

7. **VPC:** Select the created VPC and subnet.

A VPC provides an isolated virtual network for your RabbitMQ instances. You can configure and manage the network as required.

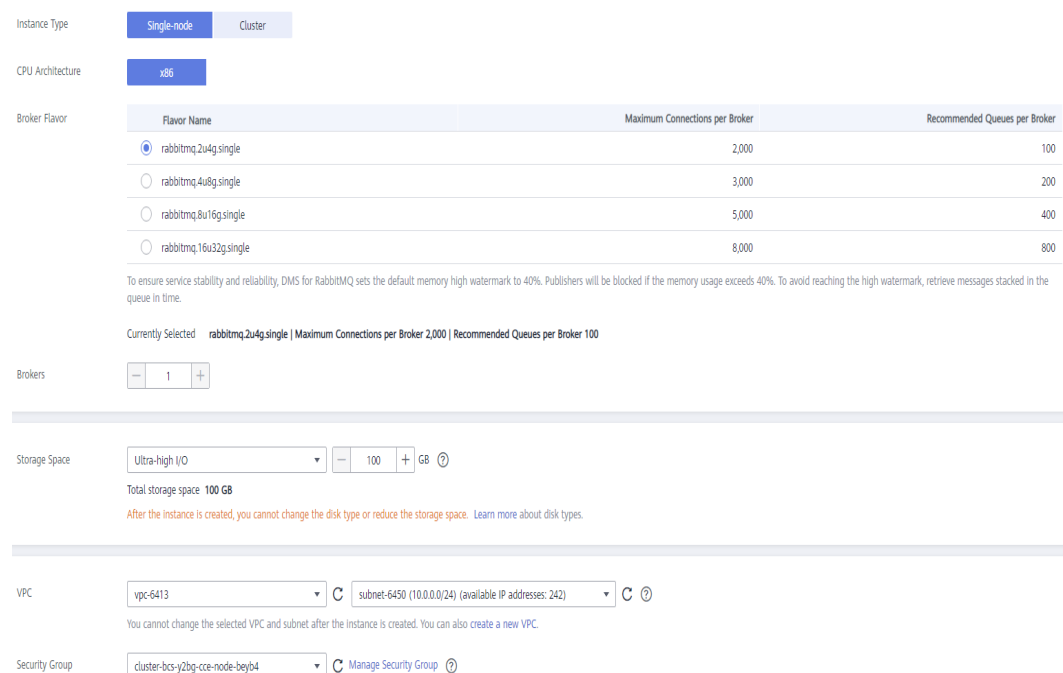
 **NOTE**

The VPC and subnet cannot be changed once the instance is created.

8. **Security Group:** Select a security group.

A security group is a set of rules for accessing a RabbitMQ instance. Click **Manage Security Group**. On the console that is displayed, view or create security groups.

**Figure 3-1** Configuring the instance parameters



The screenshot shows the configuration interface for a RabbitMQ instance. It includes the following sections:

- Instance Type:** Radio buttons for 'Single-node' (selected) and 'Cluster'.
- CPU Architecture:** A dropdown menu showing 'x86'.
- Broker Flavor:** A table with columns for Flavor Name, Maximum Connections per Broker, and Recommended Queues per Broker.
 

| Flavor Name   | Maximum Connections per Broker | Recommended Queues per Broker |
|---|--------------------------------|-------------------------------|
| <input checked="" type="radio"/> rabbitmq.2u4g.single | 2,000                          | 100                           |
| <input type="radio"/> rabbitmq.4u8g.single            | 3,000                          | 200                           |
| <input type="radio"/> rabbitmq.8u16g.single           | 5,000                          | 400                           |
| <input type="radio"/> rabbitmq.16u32g.single          | 8,000                          | 800                           |
- Brokers:** A numeric input field with a minus sign, '1', and a plus sign.
- Storage Space:** A dropdown menu for 'Ultra-high I/O', a numeric input field for '100', and a unit selector for 'GB'. Below it, it shows 'Total storage space 100 GB' and a warning: 'After the instance is created, you cannot change the disk type or reduce the storage space. Learn more about disk types.'
- VPC:** Two dropdown menus: 'vpc-64f3' and 'subnet-6450 (10.0.0.0/24) (available IP addresses: 242)'. A warning below states: 'You cannot change the selected VPC and subnet after the instance is created. You can also create a new VPC.'
- Security Group:** A dropdown menu showing 'cluster-bcs-y2bg-cce-node-beyb4' and a 'Manage Security Group' link.

**Step 9** Enter the username and password used for connecting to the RabbitMQ instance.

A username should contain 4 to 64 characters, start with a letter, and contain only letters, digits, hyphens (-), and underscores (\_).

A password must meet the following requirements:

- Contains 8 to 32 characters.
- Contains at least three types of the following characters: uppercase letters, lowercase letters, digits, and special characters `~!@#\$%^&\*()-\_+=\| [{}];:","<.>? and spaces, and cannot start with a hyphen (-).
- Cannot be the username spelled forwards or backwards.

**Step 10** Click **Advanced Settings** to configure more parameters.

1. Configure **Public Access**.

Public access can be enabled or disabled.

A RabbitMQ instance with public access enabled can be accessed by using an EIP. After you enable public access, **Elastic IP Address** is displayed. Select an EIP or click **Create Elastic IP** to view or buy EIPs.

 **NOTE**

- In comparison with intra-VPC access, enabling public access increases access latency and might lead to packet loss and jitter. Therefore, you are advised to enable public access only during the service development and testing phases.
- If you manually unbind or delete an EIP on the VPC console, the public access function of the corresponding RabbitMQ instance is automatically disabled.

2. Configure **SSL**.

This parameter indicates whether SSL authentication is enabled when a client is accessing an instance. If **SSL** is enabled, data will be encrypted before transmission for enhanced security.

**Once the instance is created, SSL cannot be enabled or disabled.**

3. Specify **Tags**.

Tags are used to identify cloud resources. When you have multiple cloud resources of the same type, you can use tags to classify them based on usage, owner, or environment.

- If you have predefined tags, select a predefined pair of tag key and value. You can click **View predefined tags** to go to the Tag Management Service (TMS) console and view or create tags.
- You can also create new tags by entering **Tag key** and **Tag value**.

Up to 20 tags can be added to each RabbitMQ instance. For details about the requirements on tags, see [Managing RabbitMQ Instance Tags](#).

4. Enter a description of the instance.

**Step 11** Click **Buy Now**.

**Step 12** Confirm the instance information, read and agree to the *Huawei Cloud Customer Agreement*, and click **Submit**.

**Step 13** Return to the instance list and check whether the instance has been created.

It takes 3 to 15 minutes to create an instance. During this period, the instance status is **Creating**.

- If the instance is created successfully, its status changes to **Running**.
- If the instance is in the **Creation failed** state, delete it by referring to [Deleting a RabbitMQ Instance](#) and try purchasing another one. If the instance purchase fails again, contact customer service.

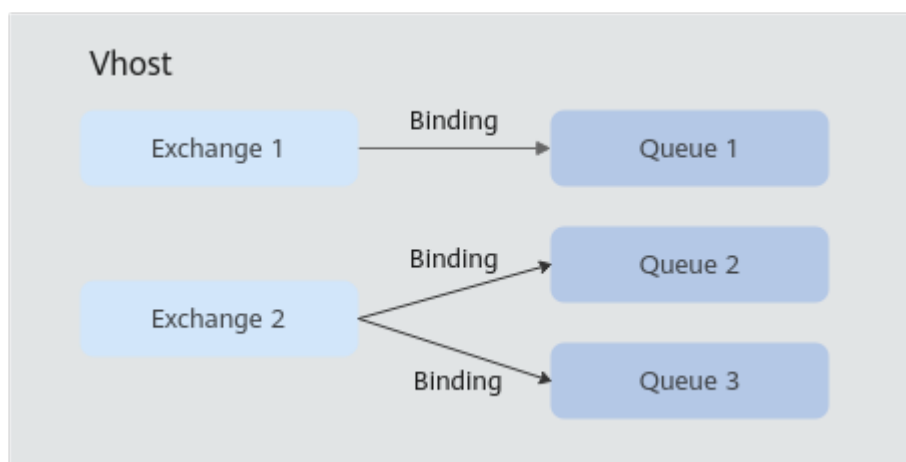
----End

# 4 Configuring Virtual Hosts

## 4.1 Creating a RabbitMQ Virtual Host

Each virtual host serves as an independent RabbitMQ server. Virtual hosts provide logical separation of exchanges, queues, and bindings. Different applications run on different virtual hosts without interfering with each other. An instance can have multiple virtual hosts, and a virtual host can have multiple exchanges and queues. To connect a producer or consumer to a RabbitMQ instance, you must specify a virtual host. For details, see [Virtual Hosts](#) on the official RabbitMQ website.

Figure 4-1 Virtual host architecture



Methods of creating a virtual host:

- [Creating a RabbitMQ Host \(Console\)](#)
- [Creating a RabbitMQ Virtual Host \(Management UI\)](#)


**NOTE**

After an instance is created, a virtual host named / is automatically created.




## Creating a RabbitMQ Host (Console)

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click **Create Virtual Host**.

**Step 7** Enter a virtual host name and click **OK**.

Once a virtual host is created, its name is fixed and it is displayed in the virtual host list.

**Tracing** indicates whether message tracing is enabled. If it is enabled, you can trace the message forwarding path.

----End

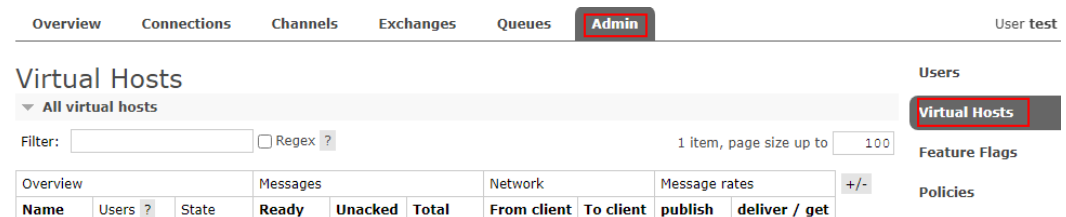
## Creating a RabbitMQ Virtual Host (Management UI)

**Step 1** Log in to the [RabbitMQ management UI](#).

**Step 2** On the top navigation bar, choose **Admin**.

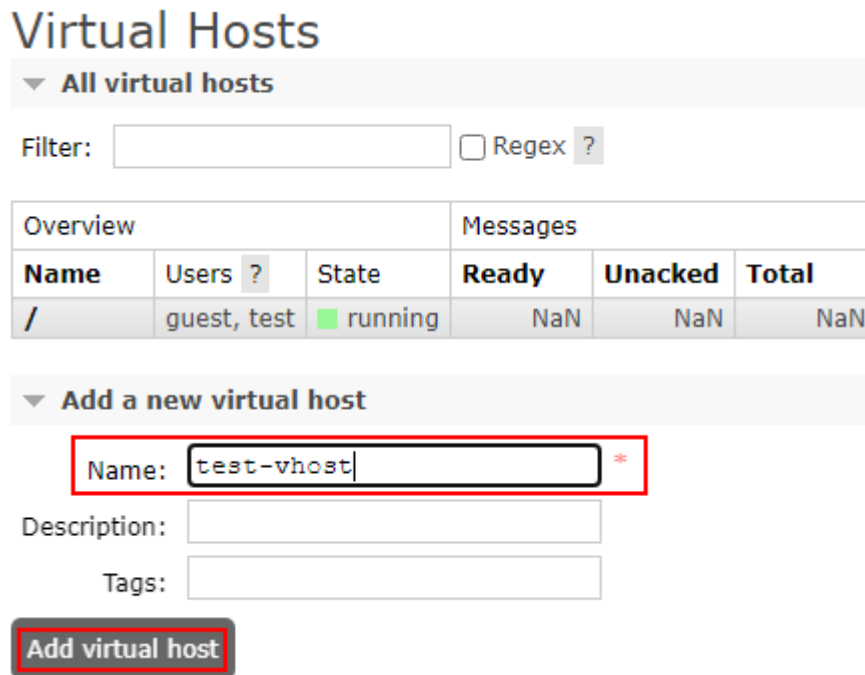
**Step 3** In the navigation tree on the right, choose **Virtual Hosts**.

**Figure 4-2** Virtual hosts



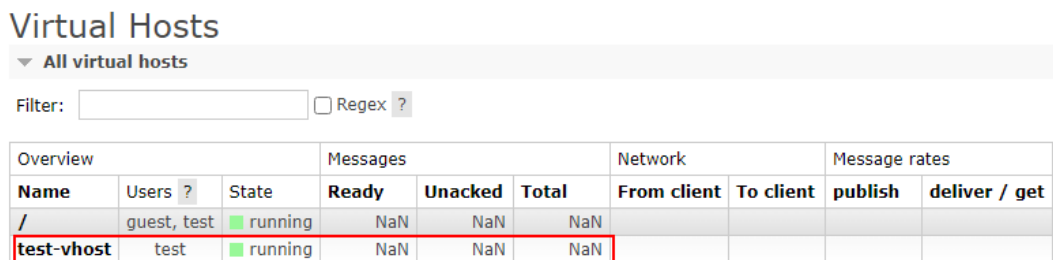
**Step 4** In the **Add a new virtual host** area, enter the virtual host name and click **Add virtual host**.

Figure 4-3 Creating a virtual host (management UI)



After the creation is successful, the new virtual host is displayed in the **All virtual hosts** area.

Figure 4-4 Virtual host list (management UI)



----End

## 4.2 Creating a RabbitMQ Exchange

Exchanges receive and assign messages. Producers send messages to exchanges first, rather than directly to queues. Exchanges route messages to one or more queues based on routing keys. If there are no matching queues, the messages are discarded.


This section describes how to create an exchange on the console. For RabbitMQ 3.x.x instances, seven exchanges are created by default after virtual host creation. These exchanges include (AMQP default), amq.direct, amq.fanout, amq.headers, amq.match, amq.rabbitmq.trace, and amq.topic.

## Prerequisites

A [virtual host](#) has been created.


## Creating a RabbitMQ Exchange

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Exchange** tab page, click **Create Exchange**. The **Create Exchange** dialog box is displayed.

**Step 8** Configure the exchange name and other parameters by referring to [Table 4-1](#).

**Table 4-1** Exchange parameters

| Parameter | Description  |
|-----------|--|
| Name      | <p>When creating an exchange, you can modify the automatically generated exchange name. Naming rules: 3–128 characters and only letters, digits, percent signs (%), vertical bars ( ), hyphens (-), underscores (_), slashes (/), or backslashes (\).</p> <p>The name cannot be changed once the exchange is created.</p>  |
| Type      | <p>Select a routing type. For details, see .</p> <ul style="list-style-type: none"> <li>• <b>direct</b>: Exchanges route messages to matching queues based on the routing keys.</li> <li>• <b>fanout</b>: Exchanges route messages to all bound queues.</li> <li>• <b>topic</b>: Exchanges route messages to queues based on routing key wildcard matching.</li> <li>• <b>headers</b>: Exchanges are related to the message headers. Routing keys are not used. Exchanges route messages based on matching between key-value pairs in the message headers and the binding (a key-value pair).</li> </ul> |

| Parameter   | Description  |
|-------------|--|
| Auto-Delete | Indicates whether to enable automatic exchange deletion. <ul style="list-style-type: none"> <li>Enabled: The exchange will be automatically deleted when the last bound queue unbound from the exchange.</li> <li>Disabled: The exchange will not be deleted when the last bound queue unbound from the exchange.</li> </ul> |
| Persistence | Indicates whether to enable exchange persistence. <ul style="list-style-type: none"> <li>Enabled: The exchange survives server restart.</li> <li>Disabled: The exchange will be deleted after server restarts and needs to be recreated.</li> </ul>  |
| Internal    | Indicates whether exchanges are for internal use. <ul style="list-style-type: none"> <li>Yes: An exchange can only bind another exchange instead of a queue.</li> <li>No: Exchanges can bind exchanges and queues.</li> </ul>  |

**Step 9** Click **OK**.

View the created exchange on the **Exchange** tab page.

----End

## 4.3 Binding a RabbitMQ Exchange

Binding an exchange is to relate an exchange to another exchange or queue. In this way, producers send messages to exchanges and exchanges route these messages to related exchanges or queues.

This section describes how to bind exchanges on the console. An exchange can be bound with a target exchange or a queue can be bound with a source exchange. An exchange can be bound with multiple target exchanges. A queue can be bound with multiple source exchanges.

### Notes and Constraints

- In RabbitMQ 3.x.x, the exchange (**AMQP default**) cannot be bound with any exchange.
- Internal** exchanges can only be bound with exchanges and not queues.

### Prerequisites

[An exchange](#) has been created.


### Binding an Exchange to a Target Exchange

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

- Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.
- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange. The **Bind Exchange** page is displayed.
- Step 8** Click **Add Binding**. The **Add Binding** dialog box is displayed.
- Step 9** Set the parameters by referring to [Table 4-2](#).

**Table 4-2** Binding parameters

| Parameter   | Description  |
|-------------|--|
| Type        | Select the binding type: Select <b>Exchange</b> .  |
| Target      | Select a target exchange to be bound.  |
| Routing Key | <p>Enter a key string to inform the exchange of which target exchanges to deliver messages to.</p> <ul style="list-style-type: none"> <li>• This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to target exchanges with the routing keys matched. If this parameter is not set, exchanges route messages to all the bound target exchanges.</li> <li>• For fanout exchanges and header exchanges, skip this parameter.</li> </ul> |

- Step 10** Click **OK**.

On the **Bindings** page, view the bound exchange.

----End

## 4.4 Creating a RabbitMQ Queue

Queues store messages. Each message is sent to one or multiple queues. Producers produce and send messages to queues, and consumers pull messages from queues for consumption.

Multiple consumers can subscribe to one queue. In this case, messages in the queue are distributed to the consumers, and no consumer can have all messages.


This section describes how to create a queue on the console.

## Prerequisites

A [virtual host](#) has been created.

## Creating a RabbitMQ Queue

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Queue** tab page, click **Create Queue**. The **Create Queue** dialog box is displayed.

**Step 8** Configure the queue name and other parameters by referring to [Table 4-3](#).

**Table 4-3** Queue parameters

| Parameter               | Description   |
|-------------------------|---|
| Name                    | When creating a queue, you can modify the automatically generated queue name. Naming rules: 3–128 characters and only letters, digits, percent signs (%), vertical bars ( ), hyphens (-), underscores (_), slashes (/), or backslashes (\). The name cannot be changed once the queue is created.       |
| Persistence             | Indicates whether to enable queue persistence. <ul style="list-style-type: none"> <li>Enabled: The queue survives after server restart.</li> <li>Disabled: The queue will be deleted after server restart and needs to be recreated.</li> </ul>   |
| Auto-Delete             | Indicates whether to enable automatic queue deletion. <ul style="list-style-type: none"> <li>Yes: The queue will be automatically deleted when the last consumer unsubscribes from the queue.</li> <li>No: The queue will not be deleted when the last consumer unsubscribes from the queue.</li> </ul> |
| Dead Letter Exchange    | Select an exchange for dead letter messages.  |
| Dead Letter Routing Key | Enter a dead letter message routing key. The dead letter exchange sends dead letter messages to the queue with a binding key that corresponds to this routing key.  |

| Parameter    | Description  |
|--------------|--|
| Time to Live | Indicates how long messages can remain, in ms. If the time to live passed and messages are still not consumed, they become dead letter messages and are sent to the dead letter exchange.  |
| Lazy Queue   | Only available for RabbitMQ 3.x.x instances.<br>Enter <b>lazy</b> to make the queue lazy.<br>Lazy queues store as many messages to the disk as possible. Messages are loaded to the memory only when they are being consumed. This reduces memory consumption. |

**Step 9** Click **OK**.

View the created queue on the **Queue** tab page.

----End

## 4.5 Binding a RabbitMQ Queue

Binding a queue is to relate an exchange to a queue. In this way, producers send messages to exchanges and exchanges route these messages to related queues.

This section describes how to bind queues for an exchange on the console. Exchanges with queues bound can route and store messages to the queues. An exchange can be bound with multiple queues.

### Notes and Constraints


- In RabbitMQ 3.x.x, the exchange (**AMQP default**) cannot be bound with any queue.
- **Internal** exchanges can only be bound with exchanges and not queues.

### Prerequisites

- **An exchange** has been created.
- **A queue** has been created.

### Binding a Queue to an Exchange

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange. The **Bind Exchange** page is displayed.
- Step 8** Click **Add Binding**. The **Add Binding** dialog box is displayed.
- Step 9** Set the parameters by referring to [Table 4-4](#).

**Table 4-4** Binding parameters

| Parameter   | Description  |
|-------------|--|
| Type        | Select the binding type: To bind a queue, select <b>Queue</b> .  |
| Target      | Select a target queue to be bound.   |
| Routing Key | <p>Enter a key string to inform the exchange of which queues to deliver messages to.</p> <ul style="list-style-type: none"> <li>This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to queues with the routing keys matched. If this parameter is not set, exchanges route messages to all the bound queues.</li> <li>For fanout exchanges and header exchanges, skip this parameter.</li> </ul> |


- Step 10** Click **OK**.
- On the **Bindings** page, view the bound queue.
- End

## 4.6 Managing RabbitMQ Virtual Hosts

### 4.6.1 Viewing a RabbitMQ Virtual Host

After a virtual host is successfully created, you can view its exchanges and queues on the console.


#### Viewing a RabbitMQ Virtual Host

- Step 1** Log in to the console.
- Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.



- Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.
- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** View the sum of exchanges or queues in the top area and their details on the **Exchange** and **Queue** tab pages.
- End





## 4.6.2 Deleting RabbitMQ Virtual Hosts

This section describes how to delete virtual hosts. **Deleting a virtual host removes all its resources including exchanges and queues permanently.**

Methods of deleting virtual hosts:

- [Deleting Virtual Hosts \(Console\)](#)
- [Deleting Virtual Hosts \(RabbitMQ Management UI\)](#)

### Deleting Virtual Hosts (Console)

- Step 1** Log in to the console.
- Step 2** In the upper left corner, click  and select a region.
-  **NOTE**
- Select the region where your RabbitMQ instance is.
- Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.
- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Virtual Hosts**.
- Step 6** Delete virtual hosts in any of the following ways:
- Select one or more virtual hosts and click **Delete Virtual Host** in the upper left corner.
  - In the row containing the desired virtual host, click **Delete**.
  - Click a virtual host name. The virtual host details page is displayed. Click **Delete** in the upper right corner.
-  **NOTE**
- The default virtual host created in instance creation cannot be deleted.
  - Deleting a virtual host removes all its resources including exchanges and queues permanently.
- Step 7** In the displayed dialog box, click **OK**.
- End

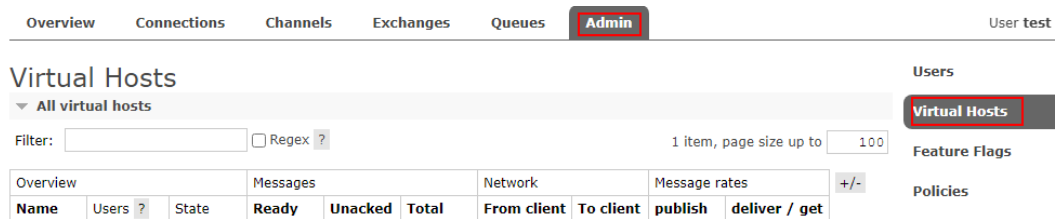
## Deleting Virtual Hosts (RabbitMQ Management UI)

**Step 1** Log in to the [RabbitMQ management UI](#).

**Step 2** On the top navigation bar, choose **Admin**.

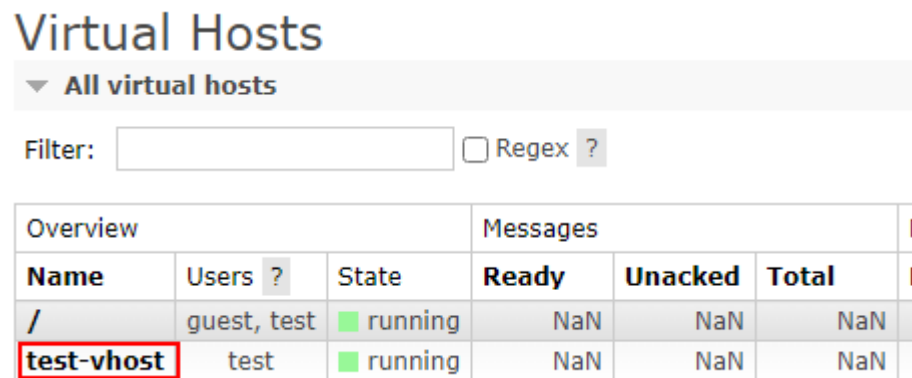
**Step 3** In the navigation tree on the right, choose **Virtual Hosts**.

**Figure 4-5** Virtual Hosts page



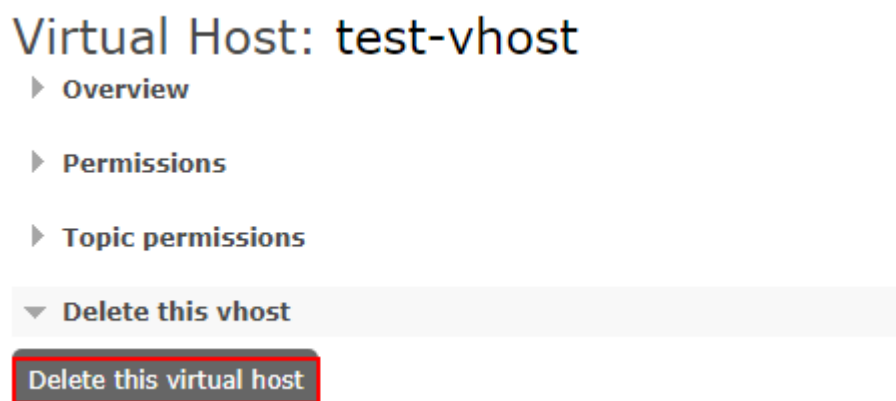
**Step 4** Click the name of the virtual host to be deleted.

**Figure 4-6** Virtual host to be deleted



**Step 5** In the **Delete this vhost** area, click **Delete this virtual host**. A confirmation dialog box is displayed.

**Figure 4-7** Deleting a virtual host



**Step 6** Click **OK**.

----End

## 4.7 Managing RabbitMQ Exchanges

### 4.7.1 Unbinding a RabbitMQ Exchange


This section describes how to unbind exchanges on the console. An exchange can be unbound from a target exchange or a queue can be unbound from a source queue.

#### Prerequisite

- **An exchange** has been created.
- The exchange or queue has been **bound with an exchange**.


#### Unbinding an Exchange from a Target Exchange

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

#### NOTE

Select the region where your RabbitMQ instance is in.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange.

**Step 8** In the row containing the desired exchange, click **Remove Binding**.

#### NOTICE

Removing a binding makes it unavailable permanently. Exercise caution.

**Step 9** Click **Yes**.

----End

### 4.7.2 Deleting RabbitMQ Exchanges

This section describes how to delete exchanges on the console. **Deleting an exchange removes all its configurations including exchange-exchange and exchange-queue bindings permanently.**


In RabbitMQ 3.x.x, the default exchange cannot be deleted.

## Prerequisite

An [exchange](#) has been created.

## Deleting RabbitMQ Exchanges

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

### NOTE

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Exchange** tab page, delete exchanges in either of the following ways:

- Select one or more exchanges and click **Delete Exchange** in the upper left corner.
- In the row containing the desired exchange, click **Delete**.

**Step 8** Click **OK**.

----End

## 4.8 Managing RabbitMQ Queues

### 4.8.1 Clearing Messages in a RabbitMQ Queue

This section describes how to clear all the messages in a queue. Methods of deleting messages:

- [Clearing Messages in a Queue \(Console\)](#)
- [Clearing Messages in a Queue \(RabbitMQ Management UI\)](#)

---

#### NOTICE

All the messages in the queue will be deleted permanently and cannot be restored. Exercise caution.


---

## Prerequisite

A [queue](#) has been created.


## Clearing Messages in a Queue (Console)

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

### NOTE

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Queue** tab page, click **Clear Message** in the row containing the desired queue. The **Clear Message** dialog box is displayed.

**Step 8** Click **OK**.

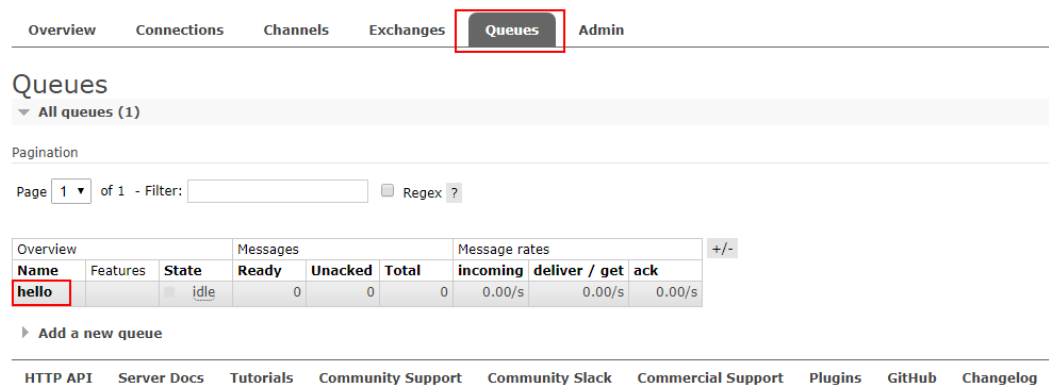
----End

## Clearing Messages in a Queue (RabbitMQ Management UI)

**Step 1** Log in to the [RabbitMQ management UI](#).

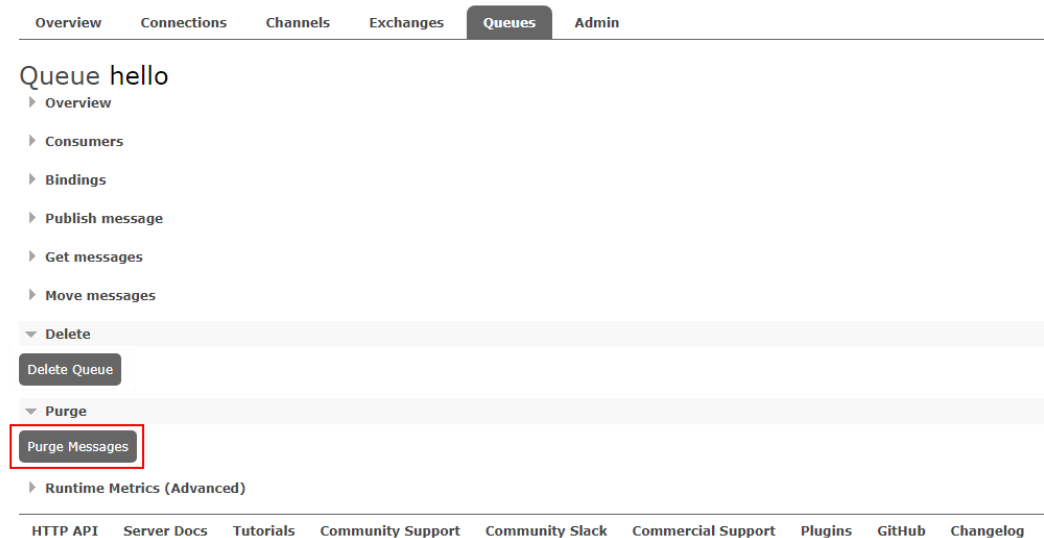
**Step 2** On the **Queues** tab page, click the name of a queue.

**Figure 4-8** Queues



**Step 3** Click **Purge Messages** to remove messages from the queue.

**Figure 4-9** Clearing messages in a queue



----End

## 4.8.2 Unbinding a RabbitMQ Queue


This section describes how to unbind an exchange from a queue on the console. Exchanges can only route and store messages to the bound queues.

### Prerequisites

- **An exchange** has been created.
- **A queue** has been created.
- The exchange has been **bound with the queue**.


### Unbinding an Exchange from a Queue

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

#### NOTE

Select the region where your RabbitMQ instance is in.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange.

**Step 8** In the row containing the queue, click **Remove Binding**.

**NOTICE**

Removing a binding makes it unavailable permanently. Exercise caution.

**Step 9** Click **Yes**.

----End

### 4.8.3 Configuring Queue Mirroring

In a RabbitMQ cluster, queues can be mirrored across multiple nodes. In the event of a node failure, services are still available because the mirrors will take over services.

This section describes how to configure queue mirroring policies for a virtual host on the RabbitMQ management UI. Queues meet the policies are mirrored queues.

#### Prerequisite

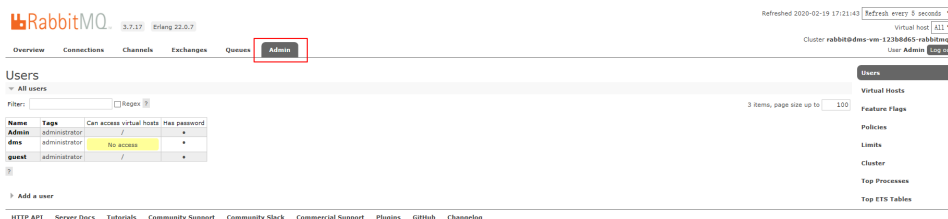
A cluster RabbitMQ instance has been created.

#### Configuring RabbitMQ Queue Mirroring

**Step 1** Log in to the [management UI of a RabbitMQ instance](#).

**Step 2** Click the **Admin** tab.

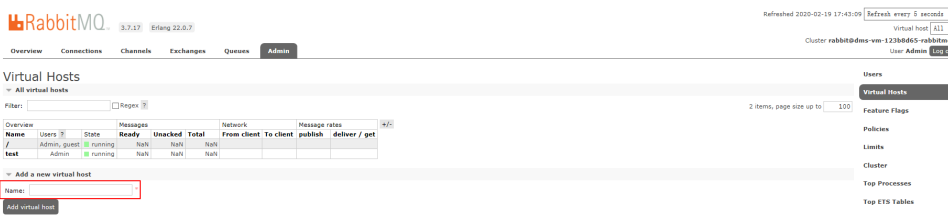
**Figure 4-10** Admin tab page



**Step 3** (Optional) Perform this step only if you need to specify a virtual host. Otherwise, go to [Step 4](#).

In the navigation tree on the right, choose **Virtual Hosts**, specify **Name**, and click **Add virtual host** to create a virtual host.

**Figure 4-11** Creating a virtual host



**Step 4** In the navigation tree on the right, choose **Policies** and set policies for the virtual host.

**Figure 4-12** Setting virtual host policies

**Add / update a policy**

Virtual host:

Name:

Pattern:

Apply to:

Priority:

Definition:

|              |   |           |        |
|--------------|---|-----------|--------|
| ha-sync-mode | = | automatic | String |
| ha-mode      | = | exactly   | String |
| ha-params    | = | 2         | Number |
|              | = |           | String |

HA [HA mode ?](#) | [HA params ?](#) | [HA sync mode ?](#) | [HA mirror promotion on shutdown ?](#)  
[HA mirror promotion on failure ?](#)

Federation [Federation upstream set ?](#) | [Federation upstream ?](#)

Queues [Message TTL](#) | [Auto expire](#) | [Max length](#) | [Max length bytes](#) | [Overflow behaviour](#)  
[Dead letter exchange](#) | [Dead letter routing key](#)  
[Lazy mode](#) | [Master Locator](#)

Exchanges [Alternate exchange ?](#)

**Table 4-5** Policy elements

| Parameter    | Description  |
|--------------|--|
| Virtual Host | Specify the virtual host. To set policies for a specific virtual host, select the virtual host created in <b>Step 3</b> from the <b>Virtual host</b> drop-down list box. If no virtual host has been created, the default value / is used. |
| Name         | The policy name, which can be customized.  |
| Pattern      | Regular expression that defines the pattern for matching queues.   |
| Apply to     | Object to which the policy applies.  |
| Priority     | A larger value indicates a higher priority.  |

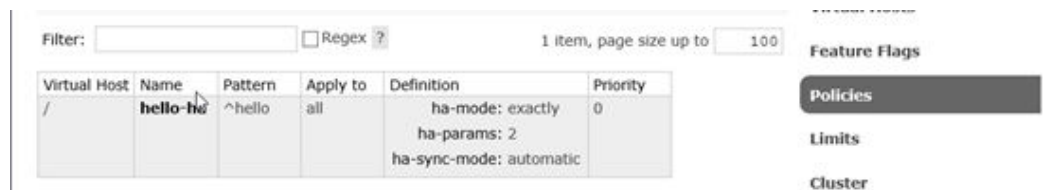


| Parameter  | Description   |
|------------|---|
| Definition | <p>Definition of the mirror, which consists of <b>ha-sync-mode</b>, <b>ha-mode</b>, and <b>ha-params</b>.</p> <ul style="list-style-type: none"> <li>• <b>ha-sync-mode</b>: queue synchronization mode. Options: <b>automatic</b> and <b>manual</b>. <ul style="list-style-type: none"> <li>- <b>automatic</b>: Data is automatically synchronized from the master.</li> <li>- <b>manual</b>: Data is manually synchronized from the master.</li> </ul> </li> <li>• <b>ha-mode</b>: queue mirroring mode. Options: <b>all</b>, <b>exactly</b>, and <b>nodes</b>. <ul style="list-style-type: none"> <li>- <b>all</b>: Mirror the queue across all nodes in the cluster.</li> <li>- <b>exactly</b>: Mirror the queue to a specific number (determined through <b>ha-params</b>) of nodes in the cluster.</li> <li>- <b>nodes</b>: Mirror the queue to specific nodes (determined through <b>ha-params</b>).</li> </ul> </li> <li>• <b>ha-params</b>: This parameter is used in the <b>ha-mode</b> mode.</li> </ul> |

**Step 5** Click **Add policy**.

The following figure shows a successfully added policy.

**Figure 4-13** Virtual host policy



----End

### 4.8.4 Configuring Lazy Queues

By default, messages produced by RabbitMQ producers are stored in the memory. When the memory needs to be released, the messages will be paged out to the disk. Paging takes a long time, during which queues cannot process messages.

If production is too fast (for example during batch processing) or consumers cannot consume messages for a long time due to various reasons (for example when consumers are offline or broke down), a large number of messages will be stacked. Memory usage becomes high and paging is frequently triggered, which may affect message sending and receiving of other queues. In this case, you are advised to use lazy queues.

Lazy queues store as many messages to the disk as possible. Messages are loaded to the memory only when they are being consumed. This reduces memory consumption, but increases I/O and affects single-queue throughput. An important

goal of lazy queues is to support long queues, that is, queues with many messages stored or stacked.

Lazy queues are recommended in the following scenarios:

- Messages may be stacked in queues.
- There is no high requirement on the queue performance (throughput), for example, less than 10,000 TPS.
- Stable production and consumption are desired, without being affected by memory changes.

Lazy queues are not suitable in the following scenarios:

- High RabbitMQ performance is expected.
- Queues are always short (with no messages stacked).
- The queue length limit is configured in a policy.

For more information about lazy queues, see [Lazy Queues](#).

 **NOTE**

Available only for RabbitMQ 3.8.35.

## Configuring a Lazy Queue

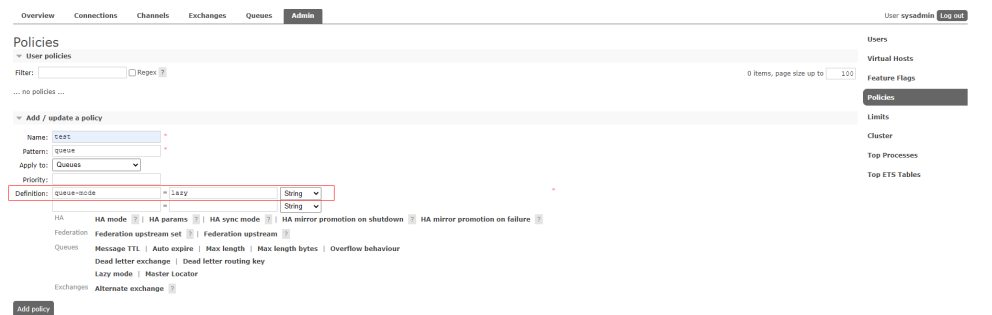
A queue has two modes: **default** and **lazy**. The default mode is **default**. To configure a queue to be **lazy**, you can use the **channel.queueDeclare** argument or a policy. If both these methods are used, the configuration set by the policy takes precedence.

- **The following example shows how to set a lazy queue by using channel.queueDeclare on a Java client.**

```
Map<String, Object> args = new HashMap<String, Object>();
args.put("x-queue-mode", "lazy");
channel.queueDeclare("myqueue", false, false, false, args);
```

- The following figure shows how to use a policy to set a lazy queue on the [RabbitMQ management UI](#).

**Figure 4-14** Using a policy to set a lazy queue



## 4.8.5 Configuring RabbitMQ Quorum Queues

Quorum queues provide the queue replication capability to ensure high data availability and security. Quorum queues can be used to replicate queue data

between RabbitMQ nodes, ensuring that queues can still run when a node breaks down.

Quorum queues can be used in scenarios where queues exist for a long time and there are high requirements on queue fault tolerance and data security and low requirements on latency and queue features. Quorum queues are not recommended if a large number of messages may be stacked, because write significantly increases disk usage.

Messages in quorum queues are preferentially stored in the memory. You are advised to limit the queue length and memory usage of quorum queues. When the number of stacked messages exceeds the threshold, the messages are written to the disk to avoid high memory watermark.

For more information about quorum queues, see [Quorum Queues](#).

 **NOTE**

Available only for RabbitMQ 3.8.35.

## Comparing Quorum Queues and Mirrored Queues

Quorum queues are introduced in RabbitMQ 3.8 to address the technical limitations of mirrored queues. Quorum queues have similar functions as mirrored queues and provide high-availability queues for RabbitMQ.

Mirrored queues are slow at replicating messages.

- A mirrored queue has a queue leader and many mirrors. When a producer sends a message to the queue leader, the leader replicates the message to the mirrors, and sends back confirmation to the producer only after all mirrors have saved the message.
- If a node in a RabbitMQ cluster goes offline due to a fault, all data in the mirrors stored on the node will be lost after the fault is rectified and the node goes online again. In this case, O&M personnel need to determine whether to replicate data from the queue leader to the mirrors. If they choose not to replicate data, messages may be lost. If they choose to replicate data, the queue is blocked during replication and no operation can be performed on the queue. When a large number of messages are stacked, the queue will be unavailable for several minutes, hours, or even longer.

Quorum queues can solve these problems.

- Quorum queues are based on a variant of the Raft consensus algorithm. They deliver a higher message throughput. A quorum queue has a primary replica (queue leader) and many followers. When a producer sends a message to the leader, the leader replicates the message to the followers, and sends back confirmation to the producer only after half of the followers have saved the message. This means that a small number of slow followers do not affect the performance of the entire queue. Similarly, the election of the leader requires the consent of more than half of the followers, which prevents the queue from having two leaders in the event of network partitioning. Therefore, quorum queues attach more importance to consistency than availability.
- After a node in a RabbitMQ cluster goes online after recovering from a fault, the data stored on the node will not be lost. The queue leader directly replicates messages from the position where the followers were interrupted.

The replication process is non-blocking, and the entire queue is not affected by the addition of new followers.

Compared with mirrored queues, quorum queues have fewer features, as shown in [Table 4-6](#). Quorum queues consume more memory and disk space.

**Table 4-6** Comparing quorum queues and mirrored queues

| Feature  | Mirrored Queues | Quorum Queues   |
|--|-----------------|---|
| Non-durable queues   | Supported       | Not supported   |
| Exclusive queues   | Supported       | Not supported   |
| Message persistence  | Per message     | Always  |
| Queue rebalancing  | Automatic       | Manual  |
| Message TTL  | Supported       | Not supported   |
| Queue TTL  | Supported       | Supported   |
| Queue length limit   | Supported       | Supported (except <b>x-overflow: reject-publish-dlx</b> ) |
| Lazy queues  | Supported       | Supported after the queue length is limited               |
| Message priority   | Supported       | Not supported   |
| Consumption priority                                       | Supported       | Supported   |
| Dead letter exchanges                                      | Supported       | Supported   |
| Dynamic policy   | Supported       | Supported   |
| Poison message (let consumers consume infinitely) handling | Not supported   | Supported   |
| Global QoS prefetch  | Supported       | Not supported   |

## Configuring Quorum Queues

When declaring a queue, set the **x-queue-type** queue argument to **quorum**. This argument can be set only during queue declaration and cannot be set by using a policy.

The default replication factor of a quorum queue is five.

- **The following example shows how to configure a quorum queue on a Java client.**

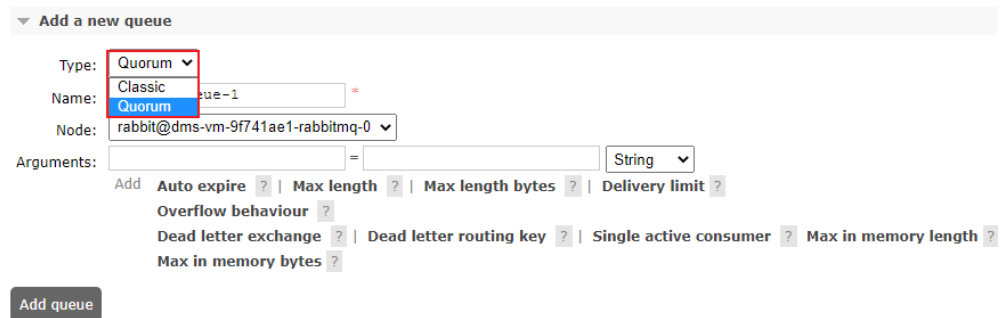
```
ConnectionFactory factory = newConnectionFactory();
factory.setRequestedHeartbeat(30);
factory.setHost(HOST);
factory.setPort(PORT);
factory.setUsername(USERNAME);
```

```
factory.setPassword(PASSWORD);

finalConnection connection = factory.newConnection();
finalChannel channel = connection.createChannel();
// Create the queue parameter map.
Map<String, Object> arguments = newHashMap<>();
arguments.put("x-queue-type", "quorum");
// Declare the quorum queue.
channel.queueDeclare("test-quorum-queue", true, false, false, arguments);
```

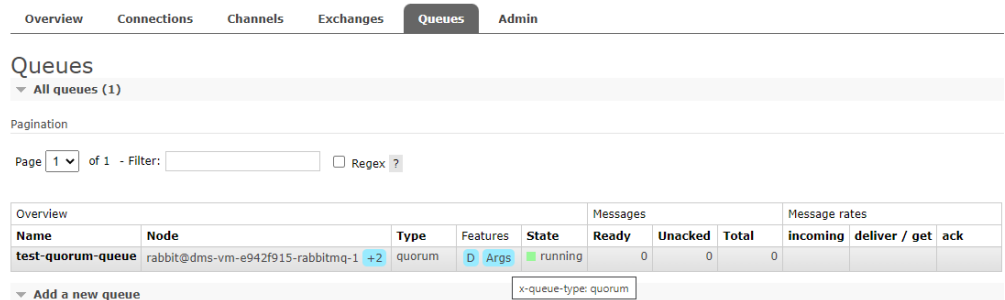
- The following example shows how to configure a quorum queue on the RabbitMQ management UI.

Figure 4-15 Configuring a quorum queue



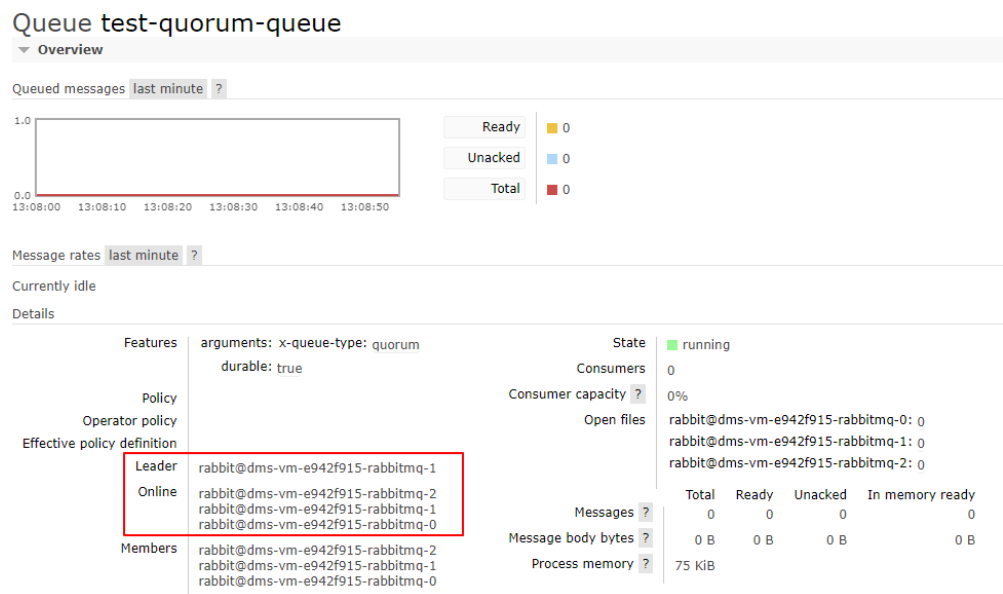
After the configuration is complete, check whether the queue type is **quorum** on the **Queues** page, as shown in Figure 4-16. In the **Node** column, **+2** indicates that the queue has two replicas. Blue indicates that the two replicas have been synchronized. Red indicates that some messages have not been replicated.

Figure 4-16 Checking the queue type



On the **Queues** page, click the name of the desired queue. Check the node where the leader of this quorum queue resides and the node where active followers reside.

**Figure 4-17** Queue details page



## Configuring the Quorum Queue Length

You can configure a policy or queue attributes to limit the length of a quorum queue and the length that can be stored in the memory.

- **x-max-length**: maximum number of messages in the quorum queue. If this limit is exceeded, excess messages will be discarded or sent to the dead letter exchange.
- **x-max-length-bytes**: maximum size (in bytes) of messages in the quorum queue. If this limit is exceeded, excess messages will be discarded or sent to the dead letter exchange.
- **x-max-in-memory-length**: maximum number of messages of the quorum queue that can be stored in memory.
- **x-max-in-memory-bytes**: maximum size (in bytes) of messages of the quorum queue that can be stored in memory.

The following describes how to limit the length of a quorum queue stored in the memory by configuring a policy or the queue attribute.

- By using a policy (recommended)  
The length of a quorum queue is limited by parameter **x-max-in-memory-bytes** in the policy.

**Figure 4-18** Using a policy to set x-max-in-memory-bytes

**Policies**

▼ User policies

Filter:   Regex ?

... no policies ...

▼ Add / update a policy

Name:  \*

Pattern:  \*

Apply to:

Priority:

Definition:  =   \*

=

Queues [All types] [Max length](#) | [Max length bytes](#) | [Overflow behaviour](#) ? | [Auto expire](#)  
[Dead letter exchange](#) | [Dead letter routing key](#)

Queues [Classic] [HA mode](#) ? | [HA params](#) ? | [HA sync mode](#) ?  
[HA mirror promotion on shutdown](#) ? | [HA mirror promotion on failure](#) ?  
[Message TTL](#) | [Lazy mode](#) | [Master Locator](#)

Queues [Quorum] [Max in memory length](#) ? | [Max in memory bytes](#) ? | [Delivery limit](#) ?

Exchanges [Alternate exchange](#) ?

Federation [Federation upstream set](#) ? | [Federation upstream](#) ?

- By setting the queue parameter  
To add a queue, set the **x-max-in-memory-length** parameter to limit the quorum queue length.

**Figure 4-19** Setting x-max-in-memory-length in the queue argument

▼ Add a new queue

Type:

Name:  \*

Node:

Arguments:  =   \*

=

Add [Auto expire](#) ? | [Max length](#) ? | [Max length bytes](#) ? | [Delivery limit](#) ?  
[Overflow behaviour](#) ?  
[Dead letter exchange](#) ? | [Dead letter routing key](#) ? | [Single active consumer](#) ? [Max in memory length](#) ?  
[Max in memory bytes](#) ?

## 4.8.6 Configuring a Single Active Consumer

A queue can have multiple registered consumers, but single active consumer allows only one consumer to consume messages from the queue. Another consumer can consume messages only when the active one is abnormal. Single active consumer can be used when the message consumption sequence must be ensured and high reliability is required.

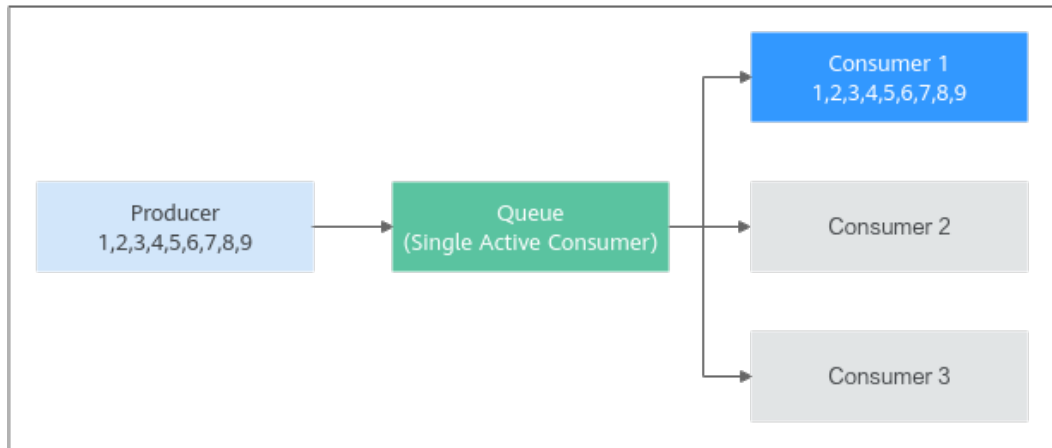
### NOTE

Only available in RabbitMQ 3.8.35.

In [Figure 4-20](#), Producer produces nine messages. Due to the setting of single active consumer, only Consumer 1 can consume messages.

For more information about single active consumer, see [Single Active Consumer](#).

**Figure 4-20** Single active consumer



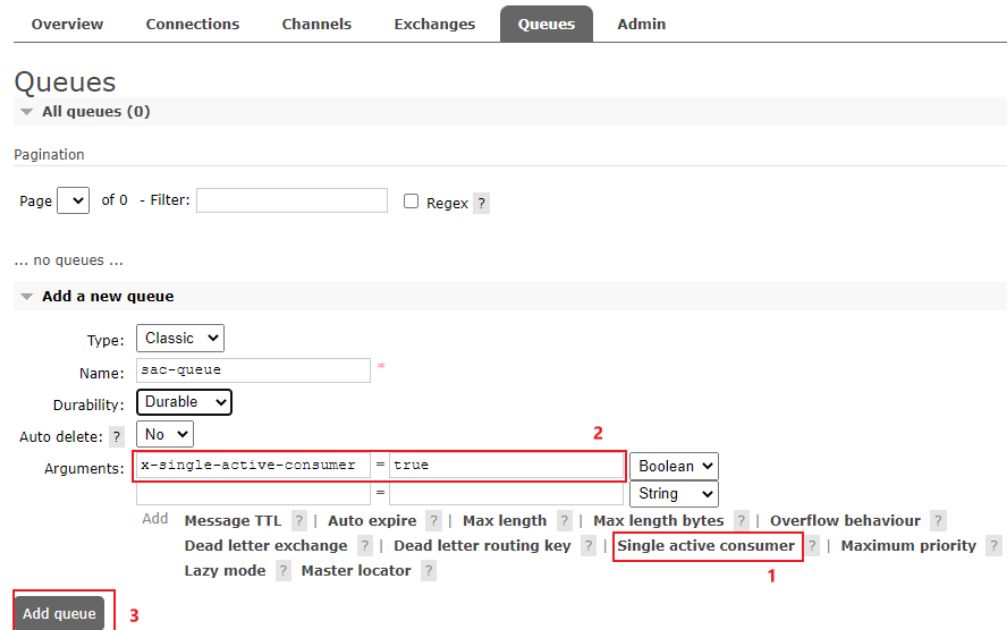
## Configuring a Single Active Consumer

When declaring a queue, you can configure a single active consumer by setting the `x-single-active-consumer` parameter to `true`.

- **The following example shows how to configure single active consumer on a Java client.**

```
Channel ch = ...;
Map<String, Object> arguments = newHashMap<String, Object>();
arguments.put("x-single-active-consumer", true);
ch.queueDeclare("my-queue", false, false, false, arguments);
```
- **The following example shows how to configure single active consumer on the [RabbitMQ management UI](#).**

**Figure 4-21** Configuring single active consumer





After the setting is complete, check whether the queue features contain single active consumer on the **Queues** page. As shown in [Figure 4-22](#), SAC indicates that a single active consumer has been set in the queue.

**Figure 4-22** Viewing queue features

Queues

▼ All queues (1)

Pagination

Page  of 1 - Filter:   Regex ?

| Overview          |         |            |       | Messages                       |         |       | Message rates |               |     | +/- |
|-------------------|---------|------------|-------|--------------------------------|---------|-------|---------------|---------------|-----|-----|
| Name              | Type    | Features   | State | Ready                          | Unacked | Total | incoming      | deliver / get | ack |     |
| sac-queue         | classic | D SAC Args | idle  | 0                              | 0       | 0     |               |               |     |     |
| ▼ Add a new queue |         |            |       | x-single-active-consumer: true |         |       |               |               |     |     |

## 4.8.7 Deleting RabbitMQ Queues

This section describes how to delete queues. **Deleting a queue removes all its configurations including exchange-queue bindings permanently.**

Methods of deleting a queue:


- [Deleting a Queue \(Console\)](#)
- [Deleting a Queue \(RabbitMQ Management UI\)](#)
- [Deleting Queues in Batches \(RabbitMQ Management UI\)](#)

### Prerequisite

[A queue](#) has been created.

### Deleting a Queue (Console)

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click an instance name to go to the instance details page.

**Step 5** In the navigation pane, choose **Virtual Hosts**.

**Step 6** Click a virtual host name.

**Step 7** On the **Queue** tab page, delete queues in either of the following ways:

- Select one or more queues and click **Delete Queue** in the upper left corner.

- In the row containing the desired queue, click **Delete**.

**Step 8** Click **OK**.

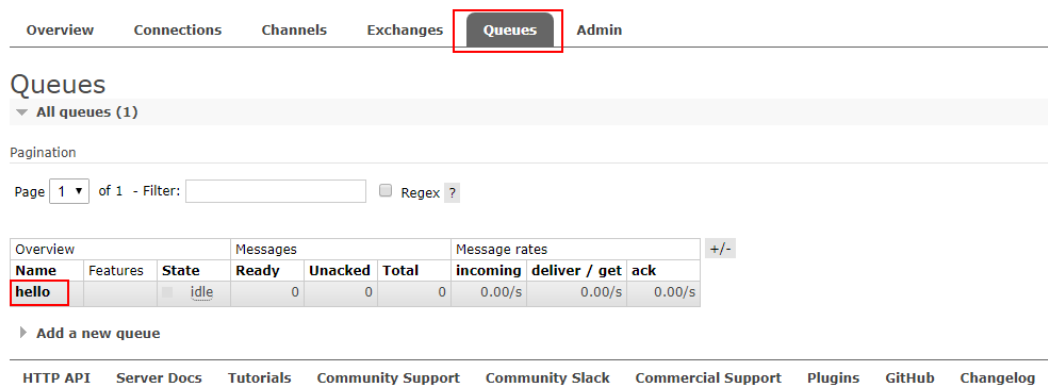
----End

## Deleting a Queue (RabbitMQ Management UI)

**Step 1** [Log in to the RabbitMQ management UI](#).

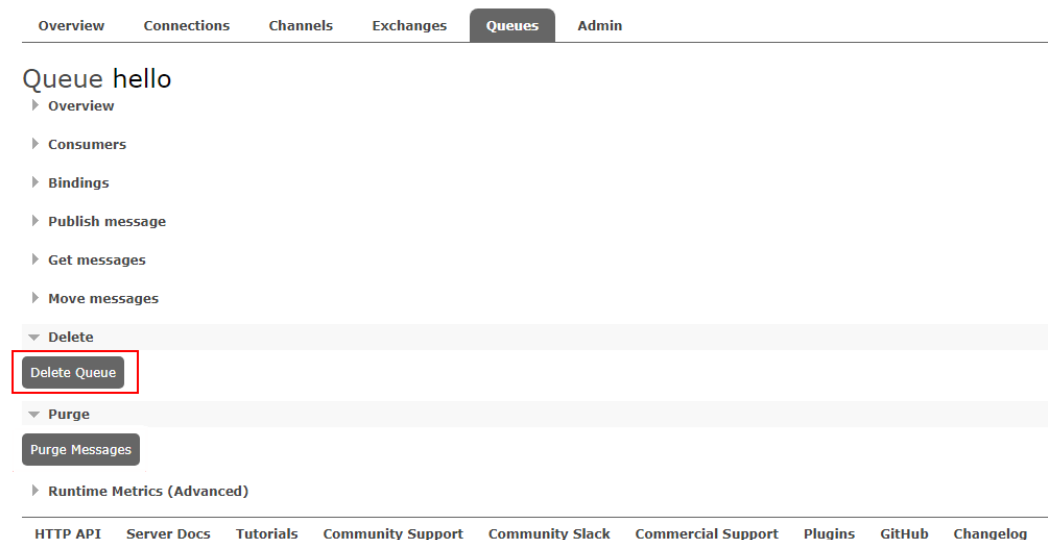
**Step 2** On the **Queues** tab page, click the name of the desired queue.

**Figure 4-23** Queues



**Step 3** Click **Delete Queue**.

**Figure 4-24** Deleting a queue



----End

## Deleting Queues in Batches (RabbitMQ Management UI)

Add a policy to delete multiple queues at a time. The policy has the same prefix as the queues to be deleted, and the queue TTL is 1 ms.

**Step 1** [Log in to the RabbitMQ management UI.](#)

**Step 2** On the **Admin > Policies** page, add a policy.

**Figure 4-25** Adding a policy to delete queues in batches

The screenshot shows the 'Policies' page in the RabbitMQ management UI. The 'Admin' tab is active. Under the 'Add / update a policy' section, the following fields are visible:

- Name:** Delete queues
- Pattern:** .\*
- Apply to:** Queues
- Priority:** (empty)
- Definition:** expires = 1

Below the definition, there are several expandable sections with links:

- HA:** HA mode, HA params, HA sync mode, HA mirror promotion on shutdown
- Federation:** Federation upstream set, Federation upstream
- Queues:** Message TTL, Auto expire, Max length, Max length bytes, Overflow behaviour, Dead letter exchange, Dead letter routing key, Lazy mode, Master Locator
- Exchanges:** Alternate exchange

An 'Add policy' button is located at the bottom left of the form area.

**Table 4-7** Policy elements

| Parameter  | Description  |
|------------|--|
| Name       | The policy name, which can be customized.  |
| Pattern    | Queue matching mode. Enter a queue name. Queues containing this queue name will be matched.<br>If this parameter is set to <code>.*</code> , all queues are matched. If this parameter is set to <code>.*queue-name</code> , all queues whose names contain <b>queue-name</b> are matched. |
| Apply to   | Object to which the policy applies. Select <b>Queues</b> .   |
| Priority   | A larger value indicates a higher priority.  |
| Definition | TTL, in milliseconds. Set <b>expires</b> to <b>1</b> , indicating that the queue expiration time is 1 ms.  |

**Step 3** Click **Add policy**.

On the **Queues** tab page, check whether the queues are successfully deleted.

**Step 4** After the queues are deleted, choose **Admin > Policies**, locate the row containing the policy added in [Step 2](#), and click **Clear** to delete the policy.

If this policy is retained, it will also apply to queues created later, and queues may be deleted by mistake.

**Figure 4-26** Deleting the policy



----End

# 5 Accessing a RabbitMQ Instance

## 5.1 Configuring RabbitMQ Network Connections

### 5.1.1 RabbitMQ Network Connection Requirements

A client can connect to a RabbitMQ instance in public or private networks. Notes before using a private network:

- By default, a client and a RabbitMQ instance are interconnected when they are deployed in a VPC.
- If they are not, you need to interconnect them because of isolation among VPCs.

**Table 5-1** Connection modes

| Mode           | How To Do  | Reference  |
|----------------|--|--|
| Public access  | Enable public access on the RabbitMQ console and configure elastic IPs (EIPs). The client can connect to the RabbitMQ instance through EIPs.   | <a href="#">Configuring RabbitMQ Public Access</a> |
| Private access | By default, a client and a RabbitMQ instance are interconnected when they are deployed in a VPC.   | -  |
|                | When a client and a RabbitMQ instance are deployed in different VPCs of the same region, interconnect two VPCs using a VPC peering connection. | <a href="#">VPC Peering Connection</a>             |

Before connecting a client to a RabbitMQ instance, allow accesses for the following security groups.

 **NOTE**

After a security group is created, its default inbound rule allows communication among ECSs within the security group and its default outbound rule allows all outbound traffic. In this case, you can access a RabbitMQ instance within a VPC, and do not need to add rules according to [Table 5-2](#).

**Table 5-2** Security group rules

| Direction | Type | Protocol | Port  | Source  | Description  |
|-----------|------|----------|-------|---|--|
| Inbound   | IPv4 | TCP      | 5672  | IP address or IP address group of the RabbitMQ client | Accessing a RabbitMQ instance at an IPv4 address on a client (without SSL) |
| Inbound   | IPv4 | TCP      | 5671  | IP address or IP address group of the RabbitMQ client | Accessing a RabbitMQ instance at an IPv4 address on a client (with SSL)    |
| Inbound   | IPv4 | TCP      | 15672 | IP address or IP address group of the RabbitMQ client | Accessing the management UI (without SSL)                                  |
| Inbound   | IPv4 | TCP      | 15671 | IP address or IP address group of the RabbitMQ client | Accessing the management UI (with SSL)                                     |

## 5.1.2 Configuring RabbitMQ Public Access

To access a RabbitMQ instance over a public network, enable public access and configure EIPs for the instance. If you no longer need public access to the instance, you can disable it as required.

### NOTICE


In comparison with intra-VPC access, packet loss and jitter may occur and the access delay increases during public access. Therefore, you are advised to enable public access to RabbitMQ instances only during the service development and testing phase.

## Prerequisite

Public access can only be enabled for instances in the **Running** state.


## Enabling Public Access

**Step 1** Log in to the console.


**Step 2** In the upper left corner, click  and select a region.

### NOTE


Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click the desired instance to view its details.

**Step 5** Click  next to **Public Access**.

**Step 6** Select an EIP from the **Elastic IP Address** drop-down list and click .

If no EIP exists in the **Elastic IP Address** drop-down list box, click **Create Elastic IP** to create an EIP on the page that is displayed. After the EIP is created, return to the RabbitMQ console, click  next to **Elastic IP Address**, and select the new EIP from the drop-down list.

It takes 10s to 30s to enable public access. After public access is enabled, the **Background Tasks** page is displayed. If the task status is **Successful**, public access is enabled successfully.

### NOTE


After enabling public access, configure the following settings:

- If SSL is not enabled, add an inbound rule to the security group, allowing access to ports 5672 and 15672.  
To access the RabbitMQ management plane, visit `http://{public IP address of the RabbitMQ instance}:15672`, and enter your username and password.  
To access the instance through clients, use port 5672.
- If SSL is enabled, add an inbound rule to the security group, allowing access to ports 5671 and 15671.  
To access the RabbitMQ management plane, visit `https://{public IP address of the RabbitMQ instance}:15671`, and enter your username and password.  
To access the instance through clients, use port 5671.

----End


## Disabling Public Access

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click the desired instance to view its details.

**Step 5** Click  next to **Public Access**.

**Step 6** Click .

It takes 10s to 30s to disable public access. After this operation, the **Background Tasks** page is displayed. If the task status is **Successful**, public access is disabled successfully.

----End

## 5.2 Configuring Heartbeats on the RabbitMQ Client

If messages may be consumed more than 90 seconds after they are produced, enable heartbeats on the client and set the heartbeats to shorter than 90 seconds, to prevent the client from being disconnected from a cluster RabbitMQ instance.

### What Is a Heartbeat?

RabbitMQ heartbeats help the application layer detect interrupted connections and unresponsive peers in a timely manner. Heartbeats also prevent some network devices from disconnecting TCP connections where there is no activity for a certain period of time. **To enable heartbeats, specify the heartbeat timeout for connections.**

The heartbeat timeout defines after how long the peer TCP connection is considered closed by the server or client. The server and client negotiate the timeout value. The client must be configured with the value to request heartbeats. The Java, .NET, and Erlang clients maintained by RabbitMQ use the following negotiation logic:

- If the heartbeat timeout set on neither the server nor the client is **0**, the smaller value is used.
- If the heartbeat timeout is set to **0** on either the server or the client, the non-zero value is used.
- If the heartbeat timeout set on both the server and the client is **0**, heartbeats are disabled.

After the heartbeat timeout is configured, the RabbitMQ server and client send AMQP heartbeat frames to each other at an interval of half the heartbeat



timeout. After a client misses two heartbeats, it is considered unreachable and the TCP connection is closed. If the client detects that the server cannot be accessed due to heartbeats, the client needs to reconnect to the server. For more information about heartbeats, see [Detecting Dead TCP Connections with Heartbeats and TCP Keepalives](#).

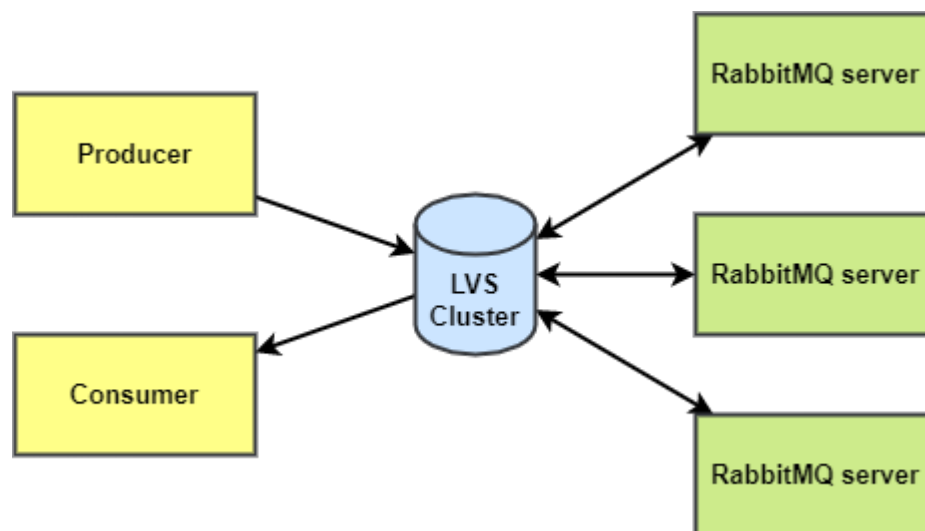
**NOTE**

Some clients (such as C clients) do not have the logic for sending heartbeats. Even if the heartbeat timeout is configured and heartbeats are enabled, heartbeats still cannot be sent. In this case, an extra thread needs to be started to compile the logic for sending heartbeats.

## LVS Heartbeat Timeout

Cluster RabbitMQ instances use Linux Virtual Servers (LVSs) for load balancing, as shown in [Figure 5-1](#). Single-node instances do not have LVSs.

**Figure 5-1** Load balancing of a cluster instance



LVS configures a heartbeat timeout of 90 seconds by default on client connections. If a client does not send a heartbeat (AMQP heartbeat frames or messages) to LVS for 90 seconds, LVS disconnects the client and the client will need to reconnect to LVS.

If messages are consumed more than 90 seconds after they are produced, enable heartbeats on the client and set the heartbeat timeout to shorter than 90 seconds.

## Configuring Heartbeats on a Client

- Java client

Before creating a connection, configure the **ConnectionFactory#setRequestedHeartbeat** parameter. Example:

```
ConnectionFactory cf = new ConnectionFactory();
// The heartbeat timeout is 60 seconds.
cf.setRequestedHeartbeat(60);
```

- .NET client

```
var cf = new ConnectionFactory();
// The heartbeat timeout is 60 seconds.
cf.RequestedHeartbeat = TimeSpan.FromSeconds(60);
```

- **Python Pika**  

```
// The heartbeat is 60 seconds.  
params = pika.ConnectionParameters(host='host', heartbeat=60,  
credentials=pika.PlainCredentials('username', 'passwd'))  
connection = pika.BlockingConnection(params)  
  
while True:  
    channel.basic_publish(exchange="", routing_key='hello', body='Hello World!')  
    print(" [x] Sent 'Hello World!'")  
    # The producer needs to use connection.sleep() to trigger a heartbeat. time.sleep() cannot trigger  
    heartbeats.  
    connection.sleep(200)
```
- **PHP client**  

```
// The heartbeat is 60 seconds.  
$connection = new AMQPStreamConnection(RMQ_HOST, RMQ_PORT, RMQ_USER, RMQ_PASS,  
RMQ_vhost, ['heartbeat'=> 60]);
```

## 5.3 Accessing RabbitMQ on a Client (SSL Disabled)

This section takes the example of a demo of DMS for RabbitMQ to describe how to access a RabbitMQ instance with SSL disabled on a RabbitMQ client for message production and consumption.

### Prerequisites

- A RabbitMQ instance with SSL disabled has been created following the instructions in [Buying a RabbitMQ Instance](#). The username and password entered in the instance creation have been obtained.
- **Instance Address (Private Network)** or **Instance Address (Public Network)** has been recorded.
- The network between the client server and the RabbitMQ instance has been established. For details about network requirements, see [RabbitMQ Network Connection Requirements](#).
- **JDK v1.8.111 or later** has been installed on the client server, and the **JAVA\_HOME** and **PATH** environment variables have been configured as follows:  
Add the following lines to the **.bash\_profile** file in the home directory as an authorized user. In this command, **/opt/java/jdk1.8.0\_151** is the JDK installation path. Change it to the path where you install JDK.

```
export JAVA_HOME=/opt/java/jdk1.8.0_151  
export PATH=$JAVA_HOME/bin:$PATH
```

Run the **source .bash\_profile** command for the modification to take effect.
- In the RabbitMQ instance: A **virtual host**, **exchange**, and **queue** have been created and an **exchange-queue binding** has been configured.

### Accessing the Instance in CLI Mode

**Step 1** Log in to the client server.

**Step 2** Run the following command to download **RabbitMQ-Tutorial.zip** (code package of the sample project):

```
wget https://dms-demo.obs.cn-north-1.myhuaweicloud.com/RabbitMQ-Tutorial.zip
```

**Step 3** Run the following command to decompress **RabbitMQ-Tutorial.zip**:

```
unzip RabbitMQ-Tutorial.zip
```

- Step 4** Run the following command to navigate to the **RabbitMQ-Tutorial** directory, which contains the precompiled JAR file:

```
cd RabbitMQ-Tutorial
```

- Step 5** Create messages using the sample project.

```
java -cp ../rabbitmq-tutorial.jar Send {host} {port} {user} {password}
```

Parameter description:

- **{host}**: connection address obtained in [Prerequisites](#)
- **{port}**: port of the RabbitMQ instance. Enter **5672**.
- **{user}**: username obtained in [Prerequisites](#)
- **{password}**: password obtained in [Prerequisites](#)

Sample message production:

```
[root@ecs-test RabbitMQ-Tutorial]# java -cp ../rabbitmq-tutorial.jar Send 192.168.xx.40 5672 test Zxxxxxxs
[x] Sent 'Hello World!'
[root@ecs-test RabbitMQ-Tutorial]# java -cp ../rabbitmq-tutorial.jar Send 192.168.xx.40 5672 test Zxxxxxxs
[x] Sent 'Hello World!'
```

- Step 6** Retrieve messages using the sample project.

```
java -cp ../rabbitmq-tutorial.jar Recv {host} {port} {user} {password}
```

Parameter description:

- **{host}**: connection address obtained in [Prerequisites](#)
- **{port}**: port of the RabbitMQ instance. Enter **5672**.
- **{user}**: username obtained in [Prerequisites](#)
- **{password}**: password obtained in [Prerequisites](#)

Sample message consumption:

```
[root@ecs-test RabbitMQ-Tutorial]# java -cp ../rabbitmq-tutorial.jar Recv 192.168.xx.40 5672 test Zxxxxxxs
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Hello World!'
[x] Received 'Hello World!'
```

To stop retrieving messages, press **Ctrl+C** to exit.

----End

## Java Sample Code

- Accessing an instance and producing messages
  - **VHOST\_NAME**: name of the virtual host that contains the queue for messages to be sent to
  - **QUEUE\_NAME**: name of the queue for messages to be sent to
  - **Hello World!**: the message to be sent in this sample

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost(host);
factory.setPort(port);
factory.setVirtualHost("VHOST_NAME");

factory.setUsername(user);
factory.setPassword(password);
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();
```

```
channel.queueDeclare(QueueName, false, false, false, null);

String message = "Hello World!";
channel.basicPublish("", QueueName, null, message.getBytes("UTF-8"));
System.out.println(" [x] Sent '" + message + "'");

channel.close();
connection.close();
```

- Accessing an instance and consuming messages
  - **VHOST\_NAME**: name of the virtual host that contains the queue to consume messages
  - **QueueName**: name of the queue to consume messages

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost(host);
factory.setPort(port);
factory.setVirtualHost("VHOST_NAME");
factory.setUsername(user);
factory.setPassword(password);
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();

channel.queueDeclare(QueueName, false, false, false, null);
System.out.println(" [*] Waiting for messages. To exit press CTRL+C");

Consumer consumer = new DefaultConsumer(channel)
{
    @Override
    public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties
properties,
        byte[] body)
        throws IOException
    {
        String message = new String(body, "UTF-8");
        System.out.println(" [x] Received '" + message + "'");
    }
};
channel.basicConsume(QueueName, true, consumer);
```

## 5.4 Accessing RabbitMQ on a Client (SSL Enabled)

This section takes the example of a demo of DMS for RabbitMQ to describe how to access a RabbitMQ instance with SSL enabled on a RabbitMQ client for message production and consumption. If SSL is enabled, data will be encrypted before transmission for enhanced security.

### Prerequisites

- A RabbitMQ instance with SSL enabled has been created following the instructions in [Buying a RabbitMQ Instance](#). The username and password entered in the instance creation have been obtained.
- **Instance Address (Private Network)** or **Instance Address (Public Network)** has been recorded.
- The network between the client server and the RabbitMQ instance has been established. For details about network requirements, see [RabbitMQ Network Connection Requirements](#).
- **JDK v1.8.111 or later** has been installed on the client server, and the **JAVA\_HOME** and **PATH** environment variables have been configured as follows:

Add the following lines to the `.bash_profile` file in the home directory as an authorized user. In this command, `/opt/java/jdk1.8.0_151` is the JDK installation path. Change it to the path where you install JDK.

```
export JAVA_HOME=/opt/java/jdk1.8.0_151
export PATH=$JAVA_HOME/bin:$PATH
```

Run the `source .bash_profile` command for the modification to take effect.

- In the RabbitMQ instance: A **virtual host**, **exchange**, and **queue** have been created and an **exchange-queue binding** has been configured.

## Accessing the Instance in CLI Mode

**Step 1** Log in to the client server.

**Step 2** Run the following command to download **RabbitMQ-Tutorial-SSL.zip** (code package of the sample project):

```
wget https://dms-demo.obs.cn-north-1.myhuaweicloud.com/RabbitMQ-Tutorial-SSL.zip
```

**Step 3** Run the following command to decompress **RabbitMQ-Tutorial-SSL.zip**:

```
unzip RabbitMQ-Tutorial-SSL.zip
```

**Step 4** Run the following command to navigate to the **RabbitMQ-Tutorial-SSL** directory, which contains the precompiled JAR file:

```
cd RabbitMQ-Tutorial-SSL
```

**Step 5** Produce messages using the sample project.

```
java -cp ./rabbitmq-tutorial-sll.jar Send {host} {port} {user} {password}
```

Parameter description:

- **{host}**: connection address obtained in [Prerequisites](#)
- **{port}**: port of the RabbitMQ instance. Enter **5671**.
- **{user}**: username obtained in [Prerequisites](#)
- **{password}**: password obtained in [Prerequisites](#)

Figure 5-2 Sample project for message creation

```
root@ecs-3b6f RabbitMQ-Tutorial-SSL# java -cp ./rabbitmq-tutorial-sll.jar Send 192.168.1.35 5671 root admin123
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[x] Sent 'Hello World!'
root@ecs-3b6f RabbitMQ-Tutorial-SSL# java -cp ./rabbitmq-tutorial-sll.jar Send 192.168.1.35 5671 root admin123
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[x] Sent 'Hello World!'
```

**Step 6** Consume messages using the sample project.

```
java -cp ./rabbitmq-tutorial-sll.jar Recv {host} {port} {user} {password}
```

Parameter description:

- **{host}**: connection address obtained in [Prerequisites](#)
- **{port}**: port of the RabbitMQ instance. Enter **5671**.
- **{user}**: username obtained in [Prerequisites](#)
- **{password}**: password obtained in [Prerequisites](#)

Figure 5-3 Sample project for message retrieval

```
root@ecs-3b6f RabbitMQ-Tutorial-SSL# java -cp ./rabbitmq-tutorial-sll.jar Recv 192.168.1.35 5671 root admin123
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Hello World!'
[x] Received 'Hello World!'
c[root@ecs-3b6f RabbitMQ-Tutorial-SSL]#
```

To stop retrieving messages, press **Ctrl+C** to exit.

----End

## Java Sample Code

- Accessing an instance and producing messages
  - **VHOST\_NAME**: name of the virtual host that contains the queue for messages to be sent to
  - **QUEUE\_NAME**: name of the queue for messages to be sent to
  - **Hello World!**: the message to be sent in this sample

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost(host);
factory.setPort(port);
factory.setVirtualHost("VHOST_NAME");

factory.setUsername(user);
factory.setPassword(password);
factory.useSslProtocol();
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();

channel.queueDeclare(QUEUE_NAME, false, false, false, null);

String message = "Hello World!";
channel.basicPublish("", QUEUE_NAME, null, message.getBytes("UTF-8"));
System.out.println(" [x] Sent '" + message + "'");

channel.close();
connection.close();
```

- Accessing an instance and consuming messages
  - **VHOST\_NAME**: name of the virtual host that contains the queue to consume messages
  - **QUEUE\_NAME**: name of the queue to consume messages

```
ConnectionFactory factory = new ConnectionFactory();
factory.setHost(host);
factory.setPort(port);
factory.setVirtualHost("VHOST_NAME");
factory.setUsername(user);
factory.setPassword(password);
factory.useSslProtocol();
Connection connection = factory.newConnection();
Channel channel = connection.createChannel();

channel.queueDeclare(QUEUE_NAME, false, false, false, null);
System.out.println(" [*] Waiting for messages. To exit press CTRL+C");

Consumer consumer = new DefaultConsumer(channel)
{
    @Override
    public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties
properties,
        byte[] body)
        throws IOException
    {
        String message = new String(body, "UTF-8");
        System.out.println(" [x] Received '" + message + "'");
    }
};
channel.basicConsume(QUEUE_NAME, true, consumer);
```

# 6 Managing Messages

## 6.1 Configuring RabbitMQ Dead Letter Messages

Dead lettering is a message mechanism in RabbitMQ. When a message is consumed, it becomes a dead letter message if any of the following happens:

- **requeue** is set to **false**, and the consumer uses **basic.reject** or **basic.nack** to negatively acknowledge (NACK) the message.
- The message has stayed in the queue for longer than the configured TTL.
- The number of messages in the queue exceeds the maximum queue length.

Such a message will be stored in a dead letter queue, if any, for special treatment. If there is no dead letter queue, the message will be discarded.

For more information about dead lettering, see [Dead Letter Exchanges](#).

### NOTICE

RabbitMQ dead letter messages may compromise performance. Exercise caution.

### Configuring dead letter exchanges and routes using queue parameters

To configure a dead letter exchange for a queue, specify the **x-dead-letter-exchange** and **x-dead-letter-routing-key** parameters when creating the queue. The queue sends the dead letter message to the dead letter exchange based on **x-dead-letter-exchange** and sets the dead letter routing key for the dead letter message based on **x-dead-letter-routing-key**.

The following example shows how to configure a dead letter exchange and routing information on a Java client.

```
channel.exchangeDeclare("some.exchange.name", "direct");  
  
Map<String, Object> args = new HashMap<String, Object>();  
args.put("x-dead-letter-exchange", "some.exchange.name");  
args.put("x-dead-letter-routing-key", "some-routing-key");  
channel.queueDeclare("myqueue", false, false, false, args);
```

## 6.2 Configuring RabbitMQ Message Acknowledgment

RabbitMQ messages are acknowledged by producers and consumers. Acknowledgments by producers ("producer confirms") and consumers are critical to ensure data reliability. If a connection fails, messages being transmitted may be lost and need to be transmitted again. The message acknowledgment mechanism enables the server and client to know when to retransmit messages. A client may acknowledge a message upon receipt of the message, or after it has completely processed the message.

**Producer confirms affect performance and should be disabled if high throughput is required. However, disabling producer confirms leads to lower reliability.**

For details about the message acknowledgment mechanism, see [Consumer Acknowledgment and Publisher Confirms](#).

### Producer Confirms

The server confirms that it has received the message sent from the producer.

The following example shows how to configure publisher confirms on a Java client.

```
try {
    channel.confirmSelect(); // Enable publisher confirms on the channel.
    // Send messages normally.
    channel.basicPublish("exchange", "routingKey", null, "publisher confirm test".getBytes());
    if (!channel.waitForConfirms()) {
        System.out.println("send message failed ");
        // do something else...
    }
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

After the **channel.waitForConfirms** method is called, the system waits for a confirmation from the server. Such synchronous waiting affects performance, but is necessary if the publisher requires at-least-once delivery.

### Consumer Acknowledgment

The server determines whether to delete a message from a queue based on whether the message is successfully received by the consumer.

Consumer acknowledgments are important to ensure data reliability. Consumer applications should take enough time to process important messages before acknowledging the messages. In this way, we do not have to worry about message losses caused by consumer process exceptions (such as program breakdown and restart) during message processing.

Consumer acknowledgment can be enabled by using the **basicConsume** method. In most cases, consumer acknowledgments are enabled on channels.

The following example shows how to configure consumer acknowledgments on a Java client (using **Channel#basicAck** and **basic.ack** for positive acknowledgment):



```
// this example assumes an existing channel instance

boolean autoAck = false;
channel.basicConsume(queueName, autoAck, "a-consumer-tag",
    new DefaultConsumer(channel) {
        @Override
        public void handleDelivery(String consumerTag,
            Envelope envelope,
            AMQP.BasicProperties properties, byte[] body)
            throws IOException
        {
            long deliveryTag = envelope.getDeliveryTag();
            // positively acknowledge a single delivery, the message will
            // be discarded
            channel.basicAck(deliveryTag, false);
        }
    });
```

Unacknowledged messages are cached in the memory. If there are too many unacknowledged messages, the memory usage becomes high. In this case, you can limit the number of messages prefetched by the consumer. For details, see [Configuring RabbitMQ Message Prefetch](#).

## 6.3 Configuring RabbitMQ Message Prefetch

Prefetch limits the number of unacknowledged messages. Once a consumer has more unacknowledged messages than the prefetch limit, the server stops sending messages to the consumer, unless at least one message is acknowledged. Prefetch is essentially a flow control measure on consumers.

Consider the following factors when setting prefetch:

- If the limit is too low, the performance may be affected, because RabbitMQ keeps waiting for the permission to send messages.
- If the limit is too high, a large number of messages may be transmitted to a consumer, leaving other consumers idle. In addition, you also need to consider consumer configurations. When processing messages, consumers save all messages in the memory. A high prefetch limit may affect consumer performance and may even crash the consumer.

For details about prefetch, see [Consumer Prefetch](#).

### Prefetch Suggestions

- If you have only one or a few consumers processing messages, it is recommended that you prefetch multiple messages at a time to keep the client busy. If your processing time and network status are stable, you can obtain an estimated prefetch value by dividing the total round-trip time by the processing time of each message on the client.
- If you have a large number of consumers and the processing time is short, a low prefetch limit is recommended. However, if the limit is too low, consumers will be idle after they have processed a batch of messages but the next batch has not yet arrived. If the limit is too high, a single consumer may be busy while other consumers are idle.
- If you have a large number of consumers and the processing time is long, set the prefetch value to **1** so that messages can be evenly distributed among all consumers.

 **NOTE**

If automatic message acknowledgment has been configured on the client, the prefetch value is invalid, and acknowledged messages are deleted from queues.

## Setting the Prefetch Value

The following example shows how to set the prefetch value to **10** for a single consumer on a Java client.

```
ConnectionFactory factory = new ConnectionFactory();  
  
Connection connection = factory.newConnection();  
Channel channel = connection.createChannel();  
  
// Set the prefetch to 10.  
channel.basicQos(10, false);  
  
QueueingConsumer consumer = new QueueingConsumer(channel);  
channel.basicConsume("my_queue", false, consumer);
```

On a Java client, the default value of **global** is **false**. Therefore, the preceding example can be simply written as **channel.basicQos(10)**.

The values of **global** are described as follows.

- **false**: applied separately to each new consumer on the channel.
- **true**: shared among all consumers on the channel.

# 7 Advanced Features

---

## 7.1 Configuring RabbitMQ Persistence

By default, messages produced by RabbitMQ producers are stored in the memory. When a node breaks down or restarts, messages are lost. RabbitMQ can persist data during such events for exchanges, queues, and messages.

**Persistence means writing messages in the memory to the disk to prevent them from being lost due to exceptions. However, if message persistence is enabled, RabbitMQ performance deteriorates because read and write operations are much slower in disks than in memory.** Different from the lazy queue mechanism, a persisted message is stored in both the disk and memory. It is deleted from the memory only when the memory is insufficient.

### NOTE

- Non-persistent queues and exchanges are lost after a restart.
- Non-persistent messages are lost after a restart. (Messages that are sent to persistent queues or exchanges will not automatically become persistent.)
- A message will be lost if the server restarts before the message persistence is complete.

### Configuring Exchange Persistence

- On the [RabbitMQ management UI](#) Set **durable** to **true** in exchange creation, as shown in [Figure 7-1](#).

**Figure 7-1** Configuring exchange persistence (management UI)

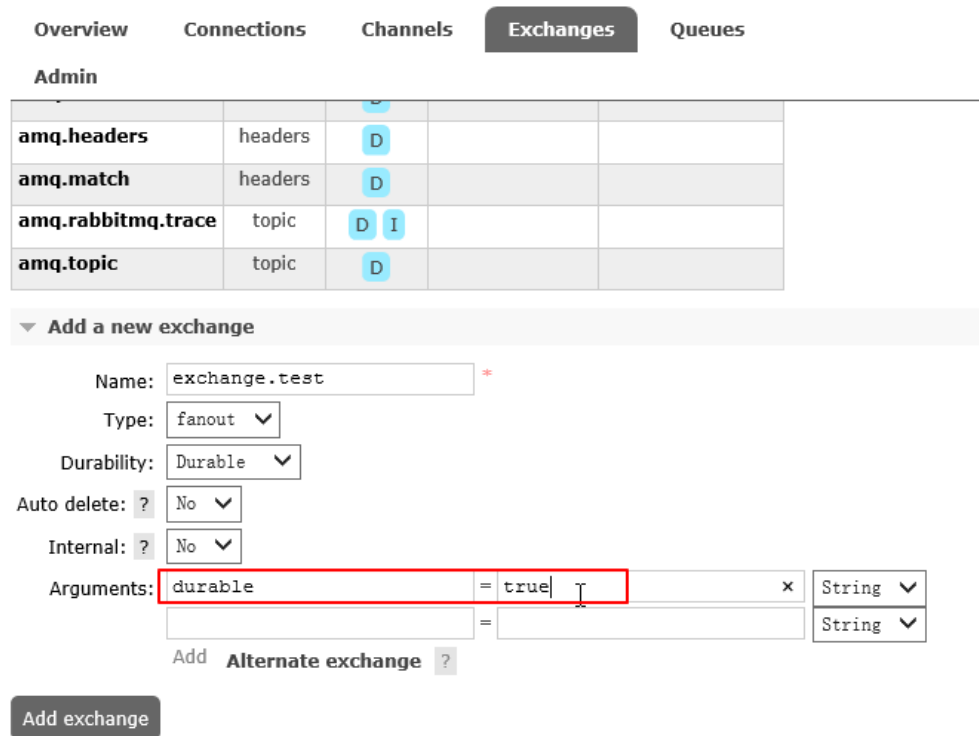
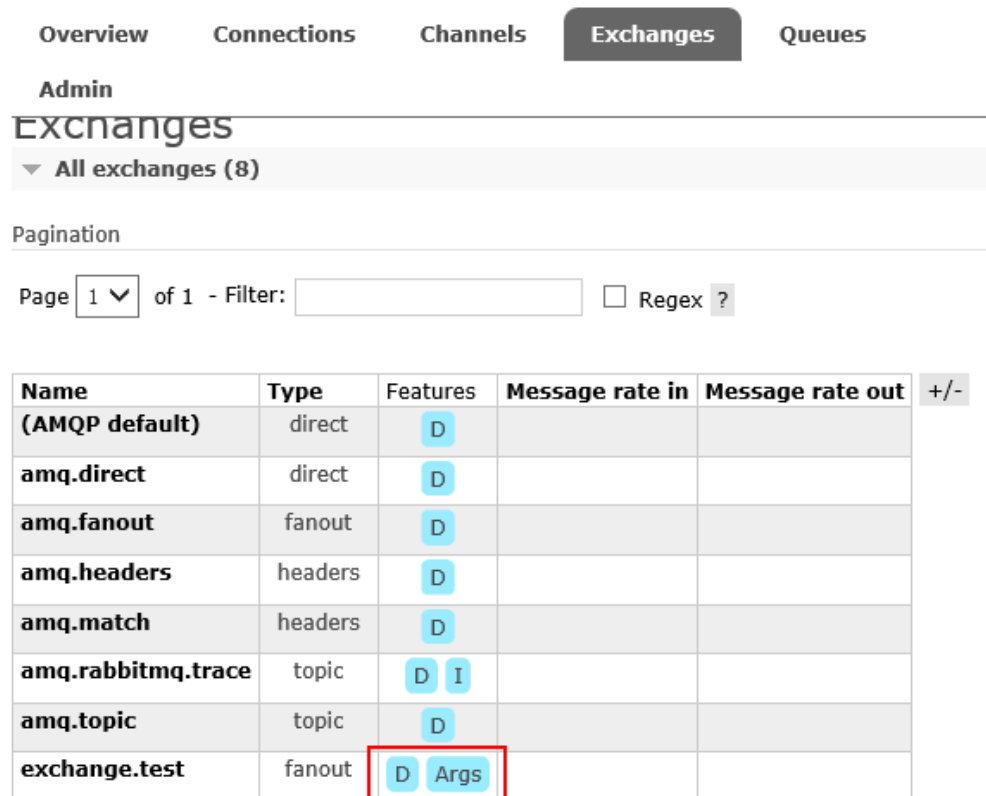


Figure 7-2 shows a successful configuration.

**Figure 7-2** Exchange persistence configured (management UI)

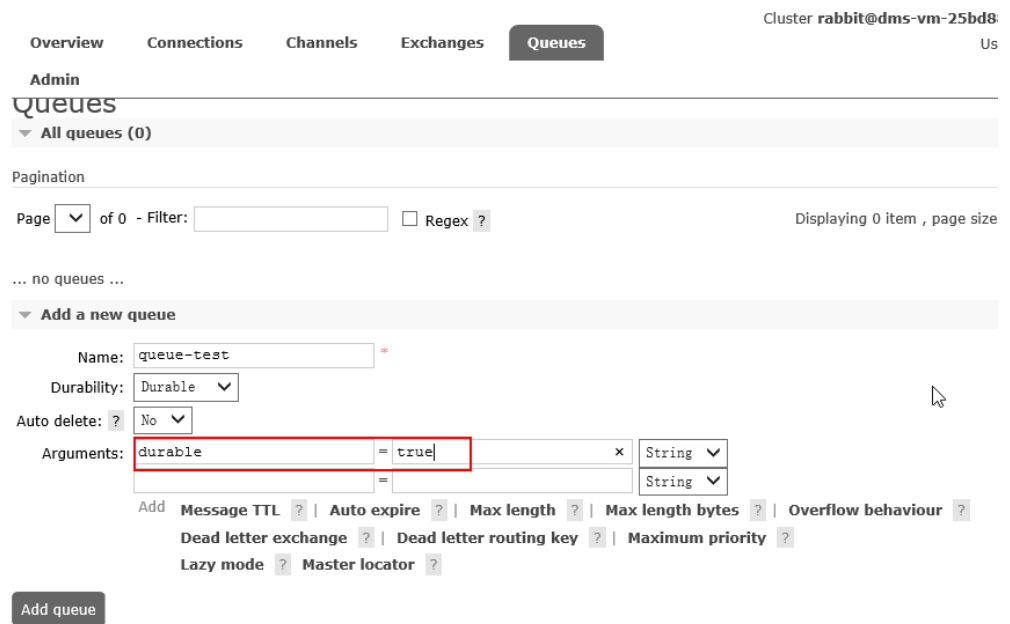


- On the RabbitMQ console  
Configure exchange persistence when creating an exchange, as shown in .  
shows a successful configuration.

## Configuring Queue Persistence

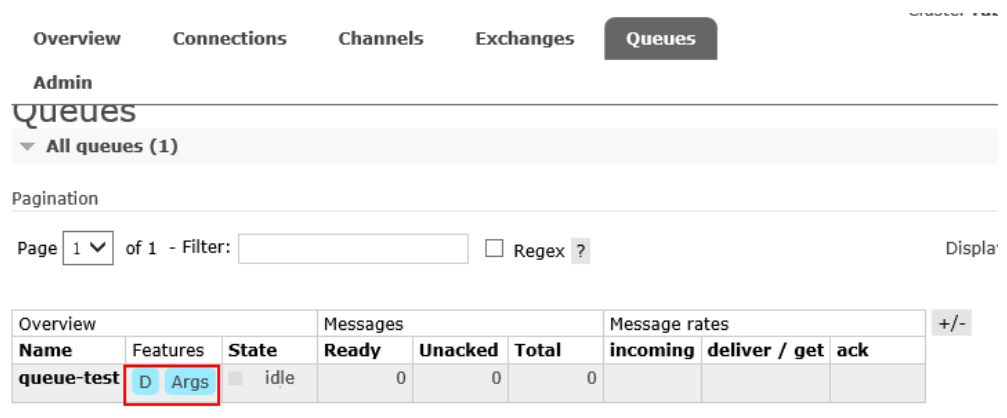
- On the [RabbitMQ management UI](#)  
Set  **durable**  to  **true**  in queue creation, as shown in [Figure 7-3](#).

**Figure 7-3** Configuring queue persistence (management UI)



[Figure 7-4](#) shows a successful configuration.

**Figure 7-4** Queue persistence configured (management UI)



- On the RabbitMQ console  
Configure queue persistence when creating a queue, as shown in .  
shows a successful configuration.

## Configuring Message Persistence

After configuring queue persistence, set **MessageProperties** to **PERSISTENT\_TEXT\_PLAIN** on the client to send persistent messages to the queue.

The following example shows how to configure message persistence on a Java client.

```
import com.rabbitmq.client.MessageProperties;
channel.basicPublish("", "my_queue", MessageProperties.PERSISTENT_TEXT_PLAIN, message.getBytes());
```

## 7.2 Configuring RabbitMQ TTL

TTL (time to live) indicates the expiration time. If a message that has stayed in a queue for longer than the TTL, the message will be discarded. If a dead letter exchange has been configured for the queue, the message will be sent to the dead letter exchange, and then routed to the dead letter queue. For more information about TTL, see [TTL](#).

You can configure TTL for messages and queues. Message TTL can be configured in the following ways:

- Configure a TTL in queue properties: All messages in the queue have the same expiration time.
- Configure a TTL for each message: Each message has a dedicated TTL.

If both methods are used, the smaller TTL value is used.

### NOTICE

TTL is a RabbitMQ feature that must be used with caution because it may adversely affect system performance.

## Configuring Queue TTL

The **x-expires** parameter in the **channel.queueDeclare** argument is used to control how long a queue will remain active after being unused before it is automatically deleted. "Unused" indicates that the queue has no consumer and is not re-declared, and the **Basic.Get** command is not called before expiration. The value of **x-expires** must be an integer other than 0, in milliseconds.

The following example shows how to configure a queue TTL on a Java client.

```
Map<String, Object> args = new HashMap<String, Object>();
args.put("x-expires", 1800000); // Set queue TTL to 1,800,000 ms.
channel.queueDeclare("myqueue", false, false, false, args);
```

## Configuring Message TTL

- In the queue configuration  
Add the **x-message-ttl** parameter to the **channel.queueDeclare** argument. The value must be an integer other than 0, in milliseconds.  
The following example shows how to configure a message TTL in queue properties on a Java client.

```
Map<String,Object> arg = new HashMap<String, Object>();  
arg.put("x-message-ttl",6000); // Set queue TTL to 6,000 ms.  
channel.queueDeclare("normalQueue",true,false,false,arg);
```

- For a dedicated message

Add the **expiration** parameter to the **channel.basicPublish** argument. The value must be an integer other than 0, in milliseconds.

The following example shows how to set a per-message TTL on a Java client.

```
byte[] messageBodyBytes = "Hello, world!".getBytes();  
AMQP.BasicProperties properties = new AMQP.BasicProperties.Builder()  
    .expiration("60000") // Set message TTL to 60,000 ms.  
    .build();  
channel.basicPublish("my-exchange", "routing-key", properties, messageBodyBytes);
```

# 8 Managing Instances

---

## 8.1 Viewing and Modifying Basic Information of a RabbitMQ Instance

This section describes how to view the details, and modify the basic information of a RabbitMQ instance on the console.


After creating a RabbitMQ instance, you can modify some configurations of it as required, including the instance name, description, and security group.

### Prerequisite

You can modify basic information of a RabbitMQ instance when the instance is in the **Running** state.

### Viewing RabbitMQ Instance Details

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

#### NOTE

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** RabbitMQ instances can be queried by filters. Current filters include the tag, status, name, connection address, and ID. Only enterprise users can use enterprise projects for filtering. For RabbitMQ instance statuses, see [Table 8-1](#).



**Table 8-1** RabbitMQ instance status description

| Status          | Description   |
|-----------------|---|
| Creating        | The instance is being created.  |
| Creation failed | The instance failed to be created.  |
| Running         | The instance is running properly.<br>Only instances in the <b>Running</b> state can provide services. |
| Faulty          | The instance is not running properly.   |
| Starting        | The status between <b>Frozen</b> and <b>Running</b> .   |
| Changing        | The instance specifications are being changed.  |
| Change failed   | The instance specifications failed to be changed.   |
| Frozen          | The instance is frozen.   |
| Freezing        | The status between <b>Running</b> and <b>Frozen</b> .   |
| Upgrading       | The instance is being upgraded.   |
| Rolling back    | The instance is being rolled back.  |

**Step 5** Click the name of the RabbitMQ instance and view the instance details.

**Table 8-2** describes the parameters for connecting to an instance. For details about other parameters, see the **Basic Information** tab page of the RabbitMQ instance on the console.


**Table 8-2** Connection parameters

| Parameter                          | Description  |
|------------------------------------|--|
| Instance Address (Private Network) | Address for connecting to the instance when public access is disabled.               |
| Mgmt. UI Address                   | Address for connecting to the instance management UI when public access is disabled. |
| Public Access                      | Whether public access has been enabled.  |
| Instance Address (Public Network)  | Address for connecting to the instance when public access is enabled.                |
| Mgmt. UI Address (Public Network)  | Address for connecting to the instance management UI when public access is enabled.  |

----End


## Modifying Basic Information of a RabbitMQ Instance

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**









Select the region where your RabbitMQ instance is in.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click a RabbitMQ instance name to go to the instance details page.

**Step 5** Modify the following parameters if needed:

**Table 8-3** Modifiable RabbitMQ parameters

| Parameter          | How to Modify  | Result   |
|--------------------|--|--|
| Instance Name      | Click  , enter a new name, and click  .<br><br>Naming rules: 4–64 characters; starts with a letter; can contain only letters, digits, hyphens (-), and underscores (_).      | The modification result is displayed in the upper right corner of the page.                                      |
| Enterprise Project | Click  , select a new enterprise project from the drop-down list, and click  .<br><br>Only for enterprise users. Modifying this parameter does not restart the instance. | The modification result is displayed in the upper right corner of the page.                                      |
| Description        | Click  , enter a new description, and click  .<br><br>0 to 1024 characters.  | The modification result is displayed in the upper right corner of the page.                                      |
| Security Group     | Click  , select a new security group from the drop-down list, and click  .<br><br>Modifying this parameter does not restart the instance.                                | The modification result is displayed in the upper right corner of the page.                                      |
| Public Access      | See <a href="#">Configuring RabbitMQ Public Access</a> .   | You will be redirected to the <b>Background Tasks</b> page, which displays the modification progress and result. |

----End

## 8.2 Viewing RabbitMQ Client Connection Addresses

When a client is connected to a RabbitMQ instance for message production and consumption, you can view the client connection addresses on the RabbitMQ management UI.

### NOTE

A client's connection addresses can be viewed only when the client is connected to a RabbitMQ instance.

### Procedure

- Step 1** Log in to the [RabbitMQ management UI](#).
- Step 2** In the navigation pane, choose **Connections**.
- Step 3** View client connection addresses, as shown in [Figure 8-1](#).

**Figure 8-1** Client connection addresses

| Overview            |                                   |           |         | Details   |            |          | Network     |           | +/- |
|---------------------|-----------------------------------|-----------|---------|-----------|------------|----------|-------------|-----------|-----|
| Name                | Node                              | User name | State   | SSL / TLS | Protocol   | Channels | From client | To client |     |
| 10.234.177.66:50996 | rabbit@dms-vm-4cd31738-rabbitmq-1 | root      | running | •         | AMQP 0-9-1 | 1        | 0iB/s       | 0iB/s     |     |
| 10.234.177.66:53332 | rabbit@dms-vm-4cd31738-rabbitmq-1 | root      | running | •         | AMQP 0-9-1 | 1        | 0iB/s       | 0iB/s     |     |
| 10.234.177.66:56272 | rabbit@dms-vm-4cd31738-rabbitmq-2 | root      | running | •         | AMQP 0-9-1 | 1        | 0iB/s       | 0iB/s     |     |
| 172.31.1.152:5004   | rabbit@dms-vm-4cd31738-rabbitmq-0 | root      | running | •         | AMQP 0-9-1 | 1        | 0iB/s       | 0iB/s     |     |

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

A client can function as a producer to create messages and as a consumer to retrieve messages. The producer and consumer IP addresses are the same, as shown in [Figure 8-1](#), and are difficult to distinguish. To differentiate between producer and consumer IP addresses, you can set the **clientProperties** parameter on the client. The following is an example:

```
// Configure client connection parameters.
HashMap<String, Object> clientProperties = new HashMap<>();
clientProperties.put("connection_name", "producer");
connectionFactory.setClientProperties(clientProperties);

// Create a connection.
Connection connection = connectionFactory.newConnection();
```

After the **clientProperties** parameter is set, the connection addresses are displayed as shown in [Figure 8-2](#).

**Figure 8-2** Client connection addresses (with producer/consumer differentiated)

Connections

▼ All connections (2)

Pagination

Page 1 of 1 - Filter:   Regex ?

| Overview                        |           |         | Details   |            |          | Network     |           |           |                        | +/- |
|---------------------------------|-----------|---------|-----------|------------|----------|-------------|-----------|-----------|------------------------|-----|
| Name                            | User name | State   | SSL / TLS | Protocol   | Channels | From client | To client | Heartbeat | Connected at           |     |
| 10.234.177.66:65260<br>consumer | admin     | running | o         | AMQP 0-9-1 | 1        | 0iB/s       | 0iB/s     | 60s       | 10:53:21<br>2022-07-13 |     |
| 10.234.177.66:58373<br>producer | admin     | running | o         | AMQP 0-9-1 | 1        | 0iB/s       | 0iB/s     | 60s       | 10:44:16<br>2022-07-13 |     |

[HTTP API](#)
[Server Docs](#)
[Tutorials](#)
[Community Support](#)
[Community Slack](#)
[Commercial Support](#)
[Plugins](#)
[GitHub](#)

----End

## 8.3 Managing RabbitMQ Instance Tags

Tags facilitate RabbitMQ instance identification and management.

You can add tags to a RabbitMQ instance when creating the instance or add tags on the details page of the created instance. Up to 20 tags can be added to an instance. Tags can be deleted.


A tag consists of a tag key and a tag value. [Table 8-4](#) lists the tag key and value requirements.

**Table 8-4** Tag key and value requirements

| Name      | Rules   |
|-----------|---|
| Tag key   | <ul style="list-style-type: none"> <li>Cannot be left blank.</li> <li>Must be unique for the same instance.</li> <li>Can contain 1 to 128 characters.</li> <li>Can contain letters, digits, spaces, and special characters <code>_.:=+@</code></li> <li>Cannot start or end with a space.</li> <li>Cannot start with <code>_sys_</code>.</li> </ul> |
| Tag value | <ul style="list-style-type: none"> <li>Can contain 0 to 255 characters.</li> <li>Can contain letters, digits, spaces, and special characters <code>_.:=+@</code></li> <li>Cannot start or end with a space in instance creation.</li> </ul>   |

### Procedure

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**


Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click the desired instance to view its details.

**Step 5** Click the **Tags** tab. Tags of the instance are displayed.

**Step 6** Perform the following operations as required:

- Adding a tag
  - a. Click **Create/Delete Tag**.
  - b. Enter a tag key and a tag value, and click **Add**.  
If you have created predefined tags, select a pair of tag key and value, and click **Add**.
  - c. Click **OK**.
- Deleting a tag  
Delete a tag using either of the following methods:
  - In the row containing the tag to be deleted, click **Delete**. Click **Yes**.
  - Click **Create/Delete Tag**. In the dialog box that is displayed, click  next to the tag to be deleted and click **OK**.

----End

## 8.4 Resetting the RabbitMQ Instance Password


If you forget the password of a RabbitMQ instance, reset the password so that you can normally access the instance.

### Prerequisite

The instance must be in the **Running** state.

### Procedure

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Reset the instance password using either of the following methods:

- In the row containing the desired instance, choose **More > Reset Password**.

- Click the desired RabbitMQ instance to view its details. In the upper right corner, choose **More > Reset Password**.

**Step 5** Enter and confirm a new password, and click **OK**.

- If the password is successfully reset, a success message will be displayed.
- If the password fails to be reset, a failure message will be displayed. If you still fail to reset the password after multiple attempts, contact customer service.

 **NOTE**

A success message is displayed only after the password is successfully reset on all brokers.

----End

## 8.5 Enabling RabbitMQ Plug-ins

After creating a RabbitMQ instance, you can enable add-ons through plug-ins. The plug-ins are disabled by default when the instance is created.

RabbitMQ plug-ins can be used for testing and service migration. Do not use them for production. Reliability issues caused from using plug-ins are not within commitments on SLAs. For details, see [Service Overview > Notes and Constraints](#).

**Table 8-5** lists plug-ins supported by RabbitMQ. **The ports of the plug-ins cannot be changed.**

**Table 8-5** Plug-ins

| Name                              | Function  | Port  |
|-----------------------------------|---|-------|
| rabbitmq_amqp1_0                  | Support for AMQP 1.0  | -     |
| rabbitmq_delayed_message_exchange | Delayed messages<br>There may be an error of about 1%. The actual delivery time may be earlier or later than the scheduled delivery time. | -     |
| rabbitmq_federation               | Federation  | -     |
| rabbitmq_sharding                 | Sharding  | -     |
| rabbitmq_shovel                   | Message moving  | -     |
| rabbitmq_tracing                  | Message tracing   | -     |
| rabbitmq_mqtt                     | Support for MQTT over TCP   | 1883  |
| rabbitmq_web_mqtt                 | Support for MQTT over WebSocket   | 15675 |
| rabbitmq_stomp                    | Support for STOMP over TCP  | 61613 |
| rabbitmq_web_stomp                | Support for STOMP over WebSocket  | 15674 |

| Name                              | Function  | Port |
|-----------------------------------|---|------|
| rabbitmq_consistent_hash_exchange | Support for x-consistent-hash. x-consistent-hash exchanges can be created after this plugin is enabled. | -    |

## Notes and Constraints


- When plug-ins are enabled, the instance will not be restarted. However, enabling plug-ins rabbitmq\_mqtt, rabbitmq\_web\_mqtt, rabbitmq\_stomp, and rabbitmq\_web\_stomp will restart Keepalived and disconnect the instance. After the instance is disconnected, it may be automatically reconnected depending on the service logic.
- The rabbitmq\_shovel, rabbitmq\_federation, and rabbitmq\_tracing plug-ins can be enabled only for specific instances. For details, see [Table 8-6](#).

**Table 8-6** Instances for which plug-ins can be enabled

| Instance                                | rabbitmq_shovel | rabbitmq_federation | rabbitmq_tracing |
|---|-----------------|---------------------|------------------|
| Single-node instances with SSL disabled | Supported       | Supported           | Supported        |
| Single-node instances with SSL enabled  | Not supported   | Not supported       | Not supported    |
| Cluster instances with SSL disabled     | Not supported   | Supported           | Supported        |
| Cluster instances with SSL enabled      | Not supported   | Not supported       | Not supported    |

## Enabling RabbitMQ Plug-ins

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Click the desired instance to view its details.

**Step 5** On the **Plug-ins** tab page, click **Enable** next to the desired plug-in.

Confirm that you want to enable the plug-in and wait for it to be enabled successfully.

----End

## 8.6 Using the rabbitmq\_tracing Plug-in

The rabbitmq\_tracing plug-in provides the message tracing function. It traces incoming and outgoing messages of RabbitMQ, captures the messages, and saves message logs to the corresponding trace file.

The rabbitmq\_shovel plug-in can be enabled only for single-node RabbitMQ instances with SSL disabled, and not for cluster instances or single-node instances with SSL enabled.

### Operation Impact

- The tracing log files may use up the disk space. You are advised not to enable the rabbitmq\_tracing plug-in for heavy-load instances.
- The disk space occupied by tracing log files is freed only after the plug-in is disabled. Do not enable the plug-in for long term. After the fault is located, close the tracing task and plug-in.

### Prerequisites

You have purchased an instance.

### Using the rabbitmq\_tracing Plug-in

**Step 1** Enable the rabbitmq\_tracing plug-in by referring to [Enabling RabbitMQ Plug-ins](#).

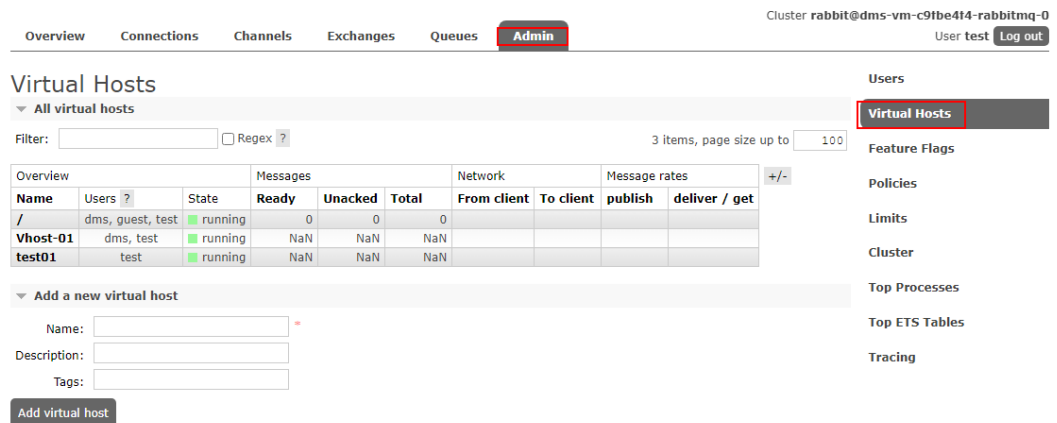
**Step 2** [Log in to the RabbitMQ management UI](#).

**Step 3** On the top navigation bar, choose **Admin**.

**Step 4** In the navigation tree on the right, choose **Virtual Hosts**.

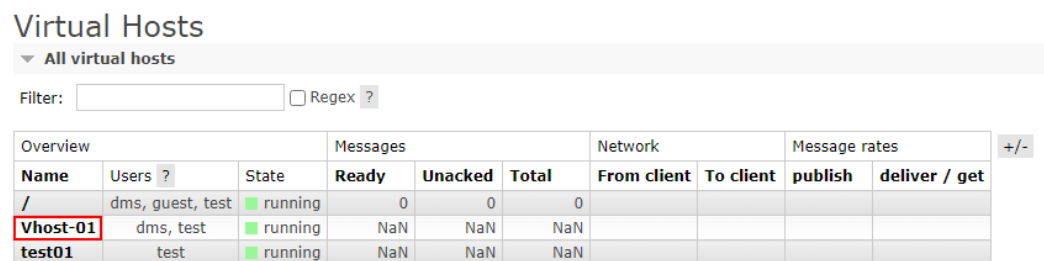


Figure 8-3 Virtual Hosts page



Step 5 Click the name of the virtual host to create a trace for.

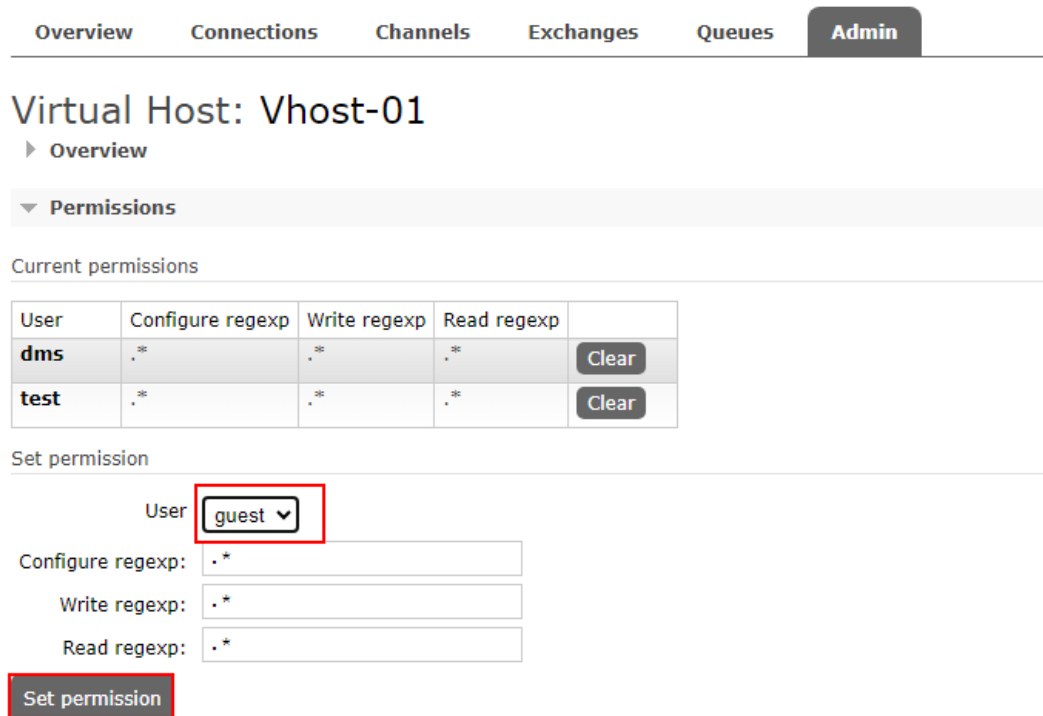
Figure 8-4 Virtual host to create a trace for



Step 6 In **Permissions** area, set the **guest** permission for the user.

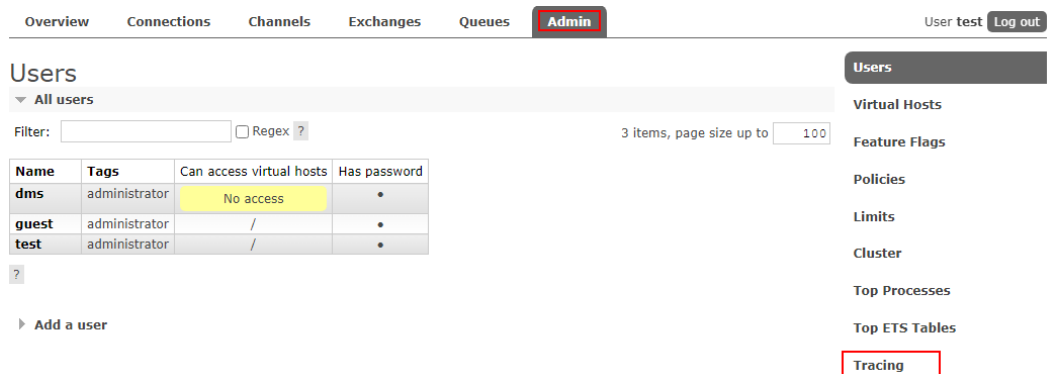
Virtual hosts must be configured the **guest** permission to enable tracing. Otherwise, an error is reported when a trace is created.

**Figure 8-5** Granting the guest permission for a virtual host



**Step 7** In the navigation tree on the right, choose **Tracing**.

**Figure 8-6** Admin page



**Step 8** In the **Add a new trace** area, set the following parameters and click **Add trace** to add a trace.

**Table 8-7** Trace parameters

| Parameter    | Description   |
|--------------|---|
| Virtual host | Name of the virtual host.   |
| Name         | Name of the trace.  |
| Format       | Format of the output message log. <b>Text</b> (easy to read) and <b>JSON</b> (easy to parse) are supported. |

| Parameter                  | Description  |
|----------------------------|--|
| Tracer connection username | Name of the user that creates tracing.   |
| Tracer connection password | Password of the user that creates a trace.   |
| Max payload bytes          | Maximum size of a message, in bytes.<br>Assume that <b>Max payload bytes</b> is set to <b>10</b> . A message larger than 10 bytes will be truncated when it is transferred through RabbitMQ. For example, <b>trace test payload</b> will become <b>trace test</b> after truncation.  |
| Pattern                    | Matching pattern. Options: <ul style="list-style-type: none"> <li>• <b>#</b>: Trace all messages entering and leaving RabbitMQ.</li> <li>• <b>publish#</b>: Trace all messages entering RabbitMQ.</li> <li>• <b>deliver#</b>: Trace all messages leaving RabbitMQ.</li> <li>• <b>publish.delay_exchange</b>: Trace messages entering a specified exchange. <i>delay_exchange</i> indicates an exchange name. Change it to the actual value.</li> <li>• <b>deliver.delay_queue</b>: Trace messages entering a specified queue. <i>delay_queue</i> indicates a queue name. Change it to the actual value.</li> </ul> |

Figure 8-7 Adding a trace

▼ Add a new trace

Virtual host:

Name:

Format:

Tracer connection username:

Tracer connection password:

Max payload bytes:  Examples: #, publish.#, deliver.# #.amq.direct, #.myqueue

Pattern:

After the trace is created, it is displayed in the **All traces** area.

Figure 8-8 Trace list

Traces: rabbit@dms-vm-c9fbe4f4-rabbitmq-0

Node:

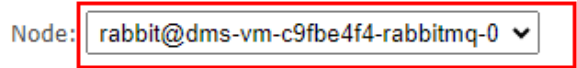
▼ All traces

| Currently running traces |                      |                        |        |               |      |           |                            | Trace log files          |      |        |
|--------------------------|----------------------|------------------------|--------|---------------|------|-----------|----------------------------|--------------------------|------|--------|
| Virtual host             | Name                 | Pattern                | Format | Payload limit | Rate | Queued    | Tracer connection username | Name                     | Size |        |
| Vhost-01                 | delay_queue_trace    | deliver.delay_queue    | text   | Unlimited     |      | 0 (queue) |                            | delay_queue_trace.log    | 0 B  | Delete |
| Vhost-01                 | delay_exchange_trace | publish.delay_exchange | text   | Unlimited     |      | 0 (queue) |                            | delay_exchange_trace.log | 0 B  | Delete |

**Step 9** (Optional) For a cluster RabbitMQ instance, switch nodes by specifying **Node** and repeat **Step 8** to create traces for them.

**Figure 8-9** Switching nodes

Traces: rabbit@dms-vm-c9fbe4f4-rabbitmq-0



**Step 10** After message logs are stored in the trace log file, click the trace log file name to view the log content.

**Figure 8-10** Trace log files

Traces: rabbit@dms-vm-c9fbe4f4-rabbitmq-0

Node: rabbit@dms-vm-c9fbe4f4-rabbitmq-0

▼ All traces

| Currently running traces |                      |                        |        |               |      |           |                            | Trace log files          |      |        |
|--------------------------|----------------------|------------------------|--------|---------------|------|-----------|----------------------------|--------------------------|------|--------|
| Virtual host             | Name                 | Pattern                | Format | Payload limit | Rate | Queued    | Tracer connection username | Name                     | Size |        |
| vhost-01                 | delay_queue_trace    | deliver.delay_queue    | text   | Unlimited     |      | 0 (queue) |                            | delay_queue_trace.log    | 0 B  | Delete |
| vhost-01                 | delay_exchange_trace | publish.delay_exchange | text   | Unlimited     |      | 0 (queue) |                            | delay_exchange_trace.log | 0 B  | Delete |

**Figure 8-11** shows the content of **delay\_exchange\_trace.log**.

**Figure 8-11** delay\_exchange\_trace.log

```

=====
2022-07-20 3:22:32:837: Message published

Node:          rabbit@dms-vm-3492b4ba-rabbitmq-0
Connection:    <rabbit@dms-vm-3492b4ba-rabbitmq-0.1657790484.10274.7>
Virtual host:  /
User:          admin
Channel:       1
Exchange:      delay_exchange
Routing keys:  [<<>>]
Routed queues: []
Properties:    [{<<"delivery_mode">>,signedint,2},{<<"headers">>,table,[]}]
Payload:       hello world
    
```

**Figure 8-12** shows the content of **delay\_queue\_trace.log**.

**Figure 8-12** delay\_queue\_trace.log

```

=====
2022-07-20 3:23:22:468: Message received

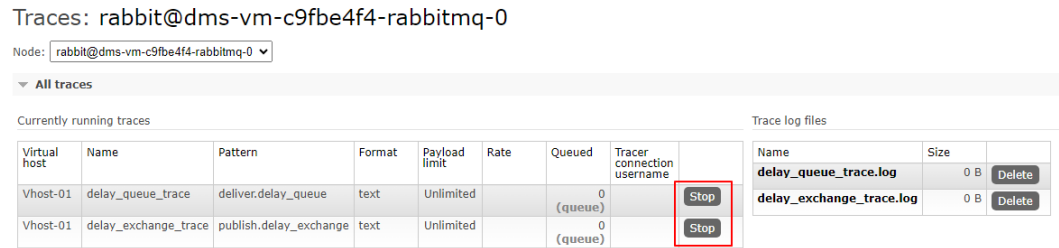
Node:          rabbit@dms-vm-3492b4ba-rabbitmq-0
Connection:    <rabbit@dms-vm-3492b4ba-rabbitmq-0.1657790484.10565.7>
Virtual host:  /
User:          admin
Channel:       1
Exchange:
Routing keys:  [<<"delay_queue">>]
Queue:         delay_queue
Properties:    [{<<"delivery_mode">>,signedint,1},{<<"headers">>,table,[]}]
Payload:
hello world

----End
    
```

## Stopping the Tracing Task and Disabling the Plug-in

**Step 1** In the **All traces** area on the **Tracing** page, click **Stop** to stop the tracing task.

**Figure 8-13** Stopping a tracing task



**Step 2** Go to the **Plug-ins** page on the RabbitMQ console, and click **Disable** next to rabbitmq\_tracing. The **Disable Plug-in** dialog box is displayed.

**Step 3** Click **Yes**. The **Background Tasks** page is displayed. If the task is in the **Successful** state, the rabbitmq\_tracing plug-in is disabled.


----End

## 8.7 Exporting the RabbitMQ Instance List

You can export a list of instances on the RabbitMQ console.


### Procedure

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is located.

- Step 3** Click  and choose **Application > Distributed Message Service (for RabbitMQ)** to open the console of DMS for RabbitMQ.
- Step 4** Export the instance list in either of the following ways:
- Select the desired instances and choose **Export > Export selected data to an XLSX file** to export specified instances.
  - Choose **Export > Export all data to an XLSX file** to export all instances.
- End

## 8.8 Restarting a RabbitMQ Instance

Restart one or more RabbitMQ instances at a time on the RabbitMQ console.




### NOTICE

When a RabbitMQ instance is being restarted, message retrieval and creation requests of the client will be rejected.

### Prerequisites

The status of the RabbitMQ instance you want to restart is in the **Running** or **Faulty** state.

### Restarting a RabbitMQ Instance

- Step 1** Log in to the console.
- Step 2** In the upper left corner, click  and select a region.
-  **NOTE**
- Select the region where your RabbitMQ instance is.
- Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.
- Step 4** Restart RabbitMQ instances using one of the following methods:
- Select one or more RabbitMQ instances and click **Restart** in the upper left corner.
  - In the row containing the desired RabbitMQ instance, click **Restart**.
  - Click the desired RabbitMQ instance to view its details. In the upper right corner, click **Restart**.
- Step 5** Click **Yes**.

It takes 3 to 15 minutes to restart a RabbitMQ instance. After it is successfully restarted, the instance should be in the **Running** state.

 **NOTE**

Restarting a RabbitMQ instance only restarts the instance process and does not restart the VM where the instance is located.

----End

## 8.9 Deleting a RabbitMQ Instance

Delete one or more RabbitMQ instances at a time on the DMS for RabbitMQ console.

---

**NOTICE**

Deleting a RabbitMQ instance will delete the data in the instance without any backup. Exercise caution when performing this operation.

---

### Prerequisite

The instance must be in the **Running**, **Faulty**, **Frozen**, or **Creation failed** state.

### Procedure

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Delete pay-per-use RabbitMQ instances in one of the following ways:

- Select one or more RabbitMQ instances and choose **More > Delete** in the upper left corner.
- In the row containing the RabbitMQ instance to be deleted, choose **More > Delete**.
- Click the desired RabbitMQ instance to view its details. In the upper right corner, choose **More > Delete**.

 **NOTE**

RabbitMQ instances in the **Creating**, **Starting**, **Changing**, or **Change failed** state cannot be deleted.

**Step 5** In the **Delete Instance** dialog box, enter **DELETE** and click **OK** to delete the RabbitMQ instance.

It takes 1 to 60 seconds to delete a RabbitMQ instance.


----End

## 8.10 Logging In to RabbitMQ Management UI

RabbitMQ instances support an open-source cluster management tool. The management UI can be accessed at the RabbitMQ management address for instance configurations.


### Procedure

**Step 1** Obtain the management address of an instance.

1. Log in to the management console.
2. In the upper left corner, click  and select a region.

#### NOTE

Select the same region as your application service.




3. Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.
4. Click the name of the instance whose management address you want to obtain. On the **Basic Information** tab page, view the **Mgmt. UI Address**, and **Username**.

#### NOTE

The username and password are customized when the RabbitMQ instance was created.

**Figure 8-14** Viewing the management UI address (without SSL)

#### Connection

|                                    |  |
|------------------------------------|--|
| SSL                                | Disabled <span style="color: orange;">Fixed for this instance</span>   |
| Username                           | root  <a href="#">Reset Password</a>  |
| Instance Address (Private Network) | IPv4 192.168.0.101:5672   |
| Mgmt. UI Address                   | <span style="border: 2px solid red; padding: 2px;">http://192.168.0.101:15672</span>  |

**Step 2** Check whether the rules of the security group of the instance are correctly configured.

1. In the **Network** section on the **Basic Information** tab page, click the name of the security group.
2. Click the **Inbound Rules** tab to view the inbound rules of the security group.
  - If SSL is disabled, allow port 15672.
  - If SSL is enabled, allow port 15671.



**Step 3** In the address box of the browser, enter the URL of the management UI.

 **NOTE**

- If public access is enabled for the RabbitMQ instance, you can use a browser to access the web page through the public network.
- If public access is not enabled for the RabbitMQ instance, you must purchase a Windows ECS that can connect to the RabbitMQ instance. Then, log in to the ECS and access the web page.

For details on how to purchase an ECS, see [Purchasing an ECS](#).

**Figure 8-15** Logging in to the management UI



**Step 4** Enter the username and password and click **Login**.

----End

# 9 Modifying RabbitMQ Instance Specifications

After creating a RabbitMQ instance, you can increase or decrease its specifications. For details, see [Table 9-1](#).

**Table 9-1** Supported specification changes (for RabbitMQ 3.x.x)

| Instance Type | Modified Object | Increase | Decrease |
|---------------|-----------------|----------|----------|
| Cluster       | Broker quantity | √        | ×        |
|               | Storage space   | √        | ×        |
|               | Broker flavor   | √        | √        |
| Single-node   | Broker quantity | ×        | ×        |
|               | Storage space   | √        | ×        |
|               | Broker flavor   | √        | √        |

## Notes and Constraints

- To ensure that the instance runs properly, do not perform other operations on the instance during the modification.
- The price may change after the modification.

## Prerequisites

A RabbitMQ instance has been created and is in the **Running** state.


## Modifying RabbitMQ Instance Specifications

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 NOTE

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** Modify the instance specifications using either of the following methods:

- In the row containing the desired instance, choose **More > Modify Specifications**.
- Click the desired RabbitMQ instance to view its details. In the upper right corner, choose **More > Modify Specifications**.

**Step 5** Specify the required storage space, number of brokers, or bandwidth.

- Expand the storage space.

For **Modify By**, select **Storage**. For **Storage Space per Broker**, specify a new storage space.

View the new storage space (Total storage space = Storage space per broker x Number of brokers) in the **Used/Available Storage Space (GB)** column in the instance list.

 NOTE

- Available storage space = Actual storage space – Storage space for storing logs – Disk formatting loss For example, if the storage space is expanded to 700 GB, the storage space for storing logs is 100 GB, and the disk formatting loss is 7 GB, then the available storage space after capacity expansion will be 593 GB.
- Storage space expansion does not affect services.
- Add brokers.  
For **Modify By**, select **Brokers**. Then, enter the number of brokers.  
View the number of brokers in the **Specifications** column in the instance list.

 NOTE

Services may temporarily stutter during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.

- Increase or decrease the broker flavor.  
For **Modify By**, select **Broker Flavor**. Then, select a new flavor.  
View the broker flavor in the **Flavor** column of the instance list.

 NOTE

- RabbitMQ 3.x.x: For cluster instances without mirrored/quorum queues configured and single-node instances, services may stutter for several minutes during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.
- RabbitMQ 3.x.x: For cluster instances configured with mirrored/quorum queues, services may stutter for several seconds during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.

**Step 6** Click **Next**, confirm the details, and click **Submit**.

----End

# 10 Migrating RabbitMQ Services

There are two scenarios for migrating RabbitMQ services:

- Single-node or cluster RabbitMQ instances can be migrated from on-premises to on-cloud RabbitMQ instances.
- An earlier RabbitMQ instance can be migrated to a later one, for example, from 3.7.17 to 3.8.35.

## Migration Principle

A RabbitMQ instance has multiple producers and consumers. To migrate services, add and remove them one by one without altering data. This process has no impact on services.

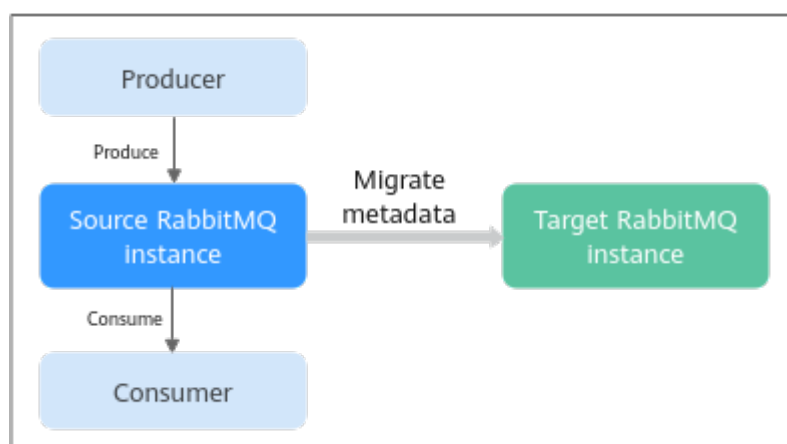
## Prerequisite

A target RabbitMQ instance has been created. For details, see [Buying a RabbitMQ Instance](#).

## Implementation (Dual-Read)

**Step 1** Migrate source RabbitMQ instance metadata to a target RabbitMQ instance.

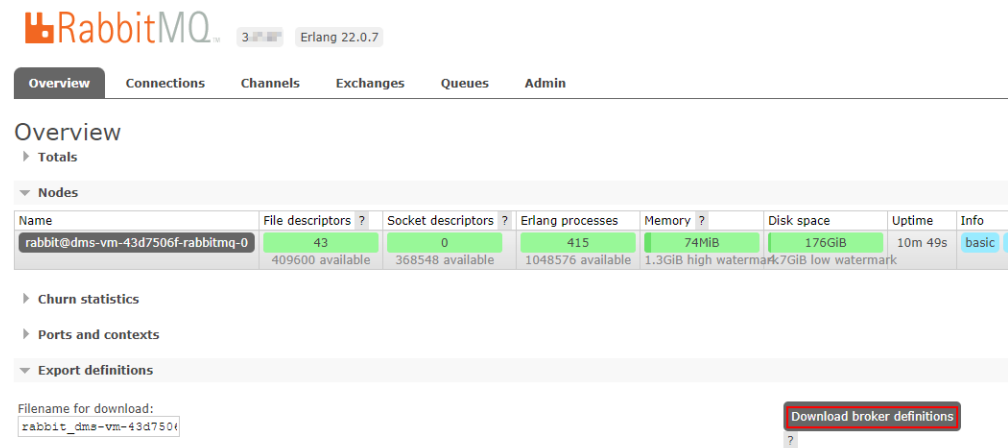
**Figure 10-1** Migrating metadata



**Do as follows:**

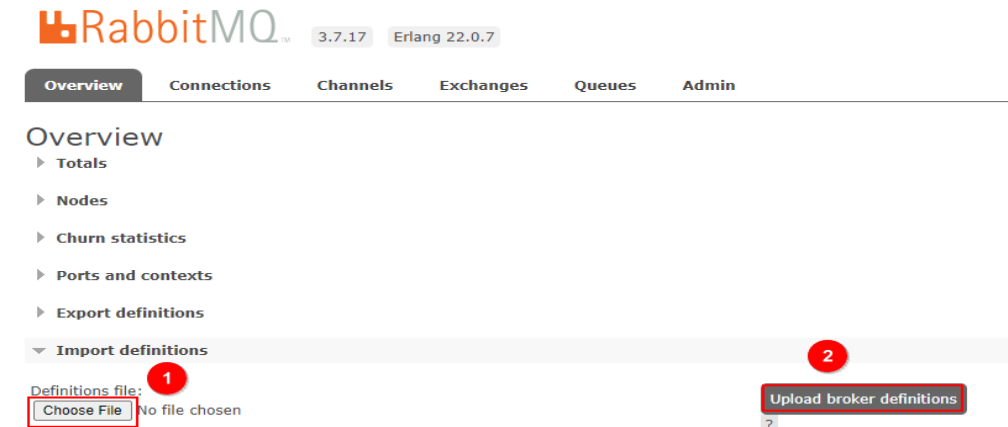
1. Log in to the management UI of the source RabbitMQ. On the **Overview** tab page, click **Download broker definitions** to export the metadata.

**Figure 10-2** Exporting metadata



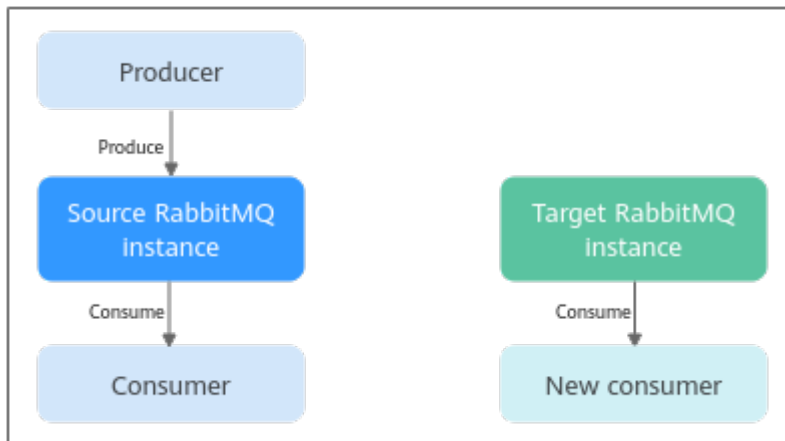
2. Log in to the management UI of the target RabbitMQ. On the **Overview** tab page, click **Choose File** and select the metadata exported in **Step 1.1**, and click **Upload broker definitions** to upload the metadata.

**Figure 10-3** Importing metadata



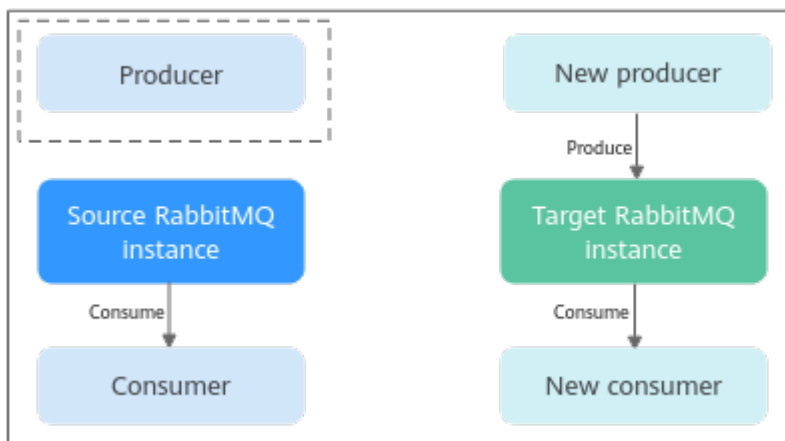
**Step 2** Add new consumers for the target RabbitMQ instance.

**Figure 10-4** Adding new consumers



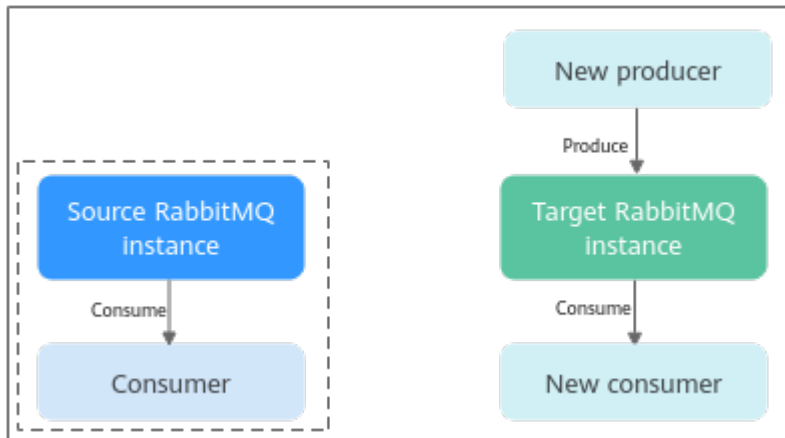
**Step 3** Add new producers for the target RabbitMQ instance and remove the producers of the source RabbitMQ instance. The old consumers continue consuming messages from the source RabbitMQ instance.

**Figure 10-5** Migrating producers



**Step 4** After the old consumers have consumed all messages from the source RabbitMQ instance, remove them along with the source RabbitMQ instance.

**Figure 10-6** Removing an old consumer and a source RabbitMQ instance



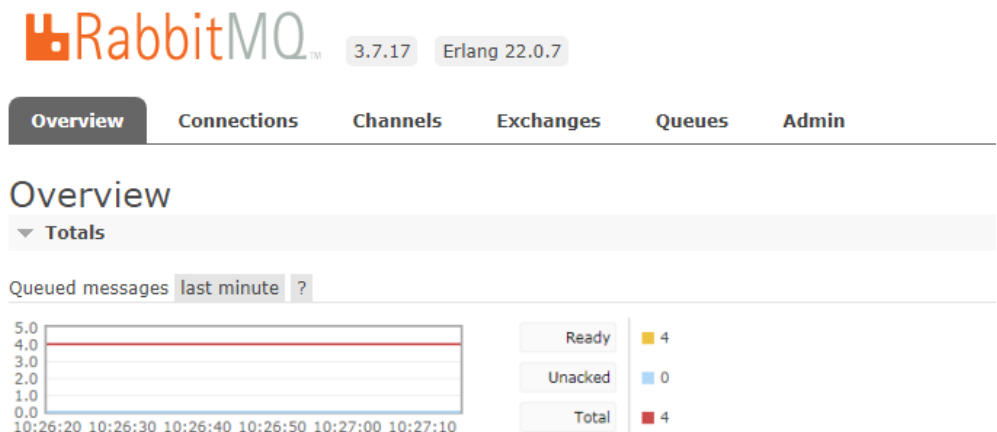
----End

## Check After Migration

Check whether the consumption from the source instance is complete in either of the following ways:

- Using the RabbitMQ management UI, as shown in [Figure 10-7](#).  
On the **Overview** tab page, if the number of messages that can be consumed (**Ready**) and the number of messages that are not acknowledged (**Unacked**) are both 0, the consumption is complete.

**Figure 10-7** RabbitMQ management UI



- Calling an API  

```
curl -s -u username:password -XGET http://ip:port/api/overview
```

Parameter description:

- *username*: account of the source instance to log in to the RabbitMQ management UI
- *password*: password of the source instance to log in to the RabbitMQ management UI
- *ip*: IP address of the source instance to log in to the RabbitMQ management UI

- *port*: port of the source instance to log in to the RabbitMQ management UI

The consumption is complete when **messages\_ready** and **messages\_unacknowledged** values in the command output are both **0**.

**Figure 10-8** Command output

```
"queue_totals":␣{  
  "messages":0,  
  "messages_details":␣{  
    "rate":0  
  },  
  "messages_ready":0,  
  "messages_ready_details":␣{  
    "rate":0  
  },  
  "messages_unacknowledged":0,  
  "messages_unacknowledged_details":␣{  
    "rate":0  
  }  
},
```



# 11 Applying for Increasing RabbitMQ Quotas

---

## What Is Quota?

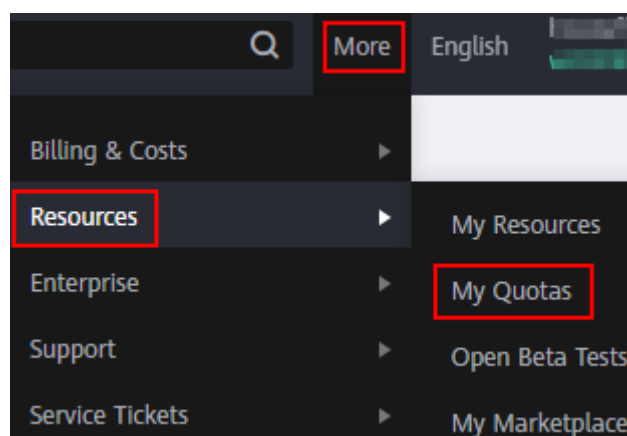
A quota is a limit on the quantity or capacity of a certain type of service resources that you can use, for example, the maximum number of RabbitMQ instances that you can create.

If the current resource quota cannot meet your service requirements, you can apply for a higher quota.

## How Do I View My Quota?

1. Log in to the console.
2. In the upper right corner of the page, choose **Resources > My Quotas**.  
The **Quotas** page is displayed.

Figure 11-1 My Quotas



3. On the **Quotas** page, view the used and total quotas of resources.  
If a quota cannot meet your needs, apply for a higher quota by performing the following operations.

## How Do I Increase My Quota?

1. Log in to the console.
2. In the upper right corner of the page, choose **Resources > My Quotas**.  
The **Service Quota** page is displayed.
3. Click **Increase Quota**.
4. On the **Create Service Ticket** page, set the parameters.  
In the **Problem Description** area, enter the required quota and the reason for the quota adjustment.
5. Read the agreements and confirm that you agree to them, and then click **Submit**.

# 12 Viewing Metrics and Configuring Alarms

---

## 12.1 Viewing RabbitMQ Metrics


Cloud Eye monitors DMS for RabbitMQ metrics in real time. You can view these metrics on the console.

### Prerequisites

At least one RabbitMQ instance has been created. The instance has at least one available message.


### Viewing RabbitMQ Metrics

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.

 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** View the instance metrics using either of the following methods:

- In the row containing the desired instance, click **View Metric**. On the Cloud Eye console, view the metrics of the instance, nodes, and queues. Metric data is reported to Cloud Eye every minute.
- Click the desired RabbitMQ instance to go to the instance details page. In the navigation pane, choose **Monitoring**. On the displayed page, view the metrics of the instance, nodes, and queues. Metric data is updated every minute.

The queue name of a RabbitMQ 3.x.x instance is displayed in two ways on the monitoring page. The name of a queue is displayed if the queue is on the default virtual host. If a queue is not on the default virtual host, the queue name is

displayed in the format "*Name of the virtual host where the queue is\_Queue name*". For example, if the **test01** queue is on **Vhost-13142708**, the queue name displayed on the monitoring page is **Vhost-13142708\_test01**.

----End

## 12.2 RabbitMQ Metrics

### Introduction

This section describes metrics reported by DMS for RabbitMQ to Cloud Eye as well as their namespaces and dimensions. You can use the Cloud Eye console to query the metrics and alarms of RabbitMQ instances. You can also view the metrics on the **Monitoring** page of the RabbitMQ console.

### Namespace

SYS.DMS

### Instance Metrics

**Table 12-1** Instance metrics

| Metric ID   | Metric Name | Description   | Value Range | Monitored Object  | Monitoring Period (Raw Data) |
|-------------|-------------|---|-------------|-------------------|------------------------------|
| connections | Connections | Number of connections in the RabbitMQ instance<br>Unit: Count | $\geq 0$    | RabbitMQ instance | 1 minute                     |
| channels    | Channels    | Number of channels in the RabbitMQ instance<br>Unit: Count    | 0-2047      | RabbitMQ instance | 1 minute                     |
| queues      | Queues      | Number of queues in the RabbitMQ instance<br>Unit: Count      | 0-7,000     | RabbitMQ instance | 1 minute                     |
| consumers   | Consumers   | Number of consumers in the RabbitMQ instance<br>Unit: Count   | 0-280,000   | RabbitMQ instance | 1 minute                     |

| Metric ID               | Metric Name                 | Description   | Value Range  | Monitored Object  | Monitoring Period (Raw Data) |
|-------------------------|-----------------------------|---|--------------|-------------------|------------------------------|
| messages_ready          | Available Messages          | Number of messages that can be consumed in the RabbitMQ instance<br>Unit: Count   | 0-10,000,000 | RabbitMQ instance | 1 minute                     |
| messages_unacknowledged | Unacknowledged Messages     | Total number of messages that have been consumed but not acknowledged in a RabbitMQ instance<br>Unit: Count                 | 0-10,000,000 | RabbitMQ instance | 1 minute                     |
| publish                 | Message Creation Rate       | Rate at which messages are produced in the RabbitMQ instance<br>Unit: Count/second  | 0-25,000     | RabbitMQ instance | 1 minute                     |
| deliver                 | Retrieval Rate (Manual Ack) | Rate at which messages are consumed (in the manual acknowledgment scenario) in a RabbitMQ instance<br>Unit: Count/second    | 0-25,000     | RabbitMQ instance | 1 minute                     |
| deliver_no_ack          | Retrieval Rate (Auto Ack)   | Rate at which messages are consumed (in the automatic acknowledgment scenario) in a RabbitMQ instance<br>Unit: Count/second | 0-50,000     | RabbitMQ instance | 1 minute                     |

## Broker Metrics

**Table 12-2** Broker metrics

| Metric ID              | Metric Name               | Description  | Value Range              | Monitored Object       | Monitoring Period (Raw Data) |
|------------------------|---------------------------|--|--------------------------|------------------------|------------------------------|
| fd_used                | File Handles              | Number of file handles used by RabbitMQ in the node<br>Unit: Count           | 0–65,535                 | RabbitMQ instance node | 1 minute                     |
| socket_used            | Socket Connections        | Number of socket connections used by RabbitMQ in the node<br>Unit: Count     | 0–50,000                 | RabbitMQ instance node | 1 minute                     |
| proc_used              | Erlang Processes          | Number of Erlang processes used by RabbitMQ in the node<br>Unit: Count       | 0–1,048,576              | RabbitMQ instance node | 1 minute                     |
| mem_used               | Memory Usage              | Memory usage of RabbitMQ in the node<br>Unit: byte, KB, MB, GB, TB or PB     | 0–32,000,000,000         | RabbitMQ instance node | 1 minute                     |
| disk_free              | Available Memory          | Available memory of RabbitMQ in the node<br>Unit: byte, KB, MB, GB, TB or PB | 0–500,000,000,000        | RabbitMQ instance node | 1 minute                     |
| rabbitmq_alive         | Node Alive                | Whether the RabbitMQ node is alive   | 1: alive<br>0: not alive | RabbitMQ instance node | 1 minute                     |
| rabbitmq_disk_usage    | Disk Capacity Usage       | Disk usage of the RabbitMQ VM<br>Unit: %                                     | 0–100%                   | RabbitMQ instance node | 1 minute                     |
| rabbitmq_cpu_usage     | CPU Usage                 | CPU usage of the RabbitMQ VM<br>Unit: %                                      | 0–100%                   | RabbitMQ instance node | 1 minute                     |
| rabbitmq_cpu_core_load | Average Load per CPU Core | Average load of each CPU core of the RabbitMQ VM                             | > 0                      | RabbitMQ instance node | 1 minute                     |

| Metric ID                      | Metric Name             | Description   | Value Range     | Monitored Object       | Monitoring Period (Raw Data) |
|--------------------------------|-------------------------|---|-----------------|------------------------|------------------------------|
| rabbitmq_memory_usage          | Memory Usage            | Memory usage of the RabbitMQ VM<br>Unit: %  | 0–100%          | RabbitMQ instance node | 1 minute                     |
| rabbitmq_disk_read_await       | Average Disk Read Time  | Average time for each disk I/O read in the monitoring period<br>Unit: ms                      | > 0             | RabbitMQ instance node | 1 minute                     |
| rabbitmq_disk_write_await      | Average Disk Write Time | Average time for each disk I/O write in the monitoring period<br>Unit: ms                     | > 0             | RabbitMQ instance node | 1 minute                     |
| rabbitmq_node_bytes_in_rate    | Inbound Traffic         | Inbound traffic per second<br>Unit: byte/s, KB/s, MB/s, or GB/s                               | > 0             | RabbitMQ instance node | 1 minute                     |
| rabbitmq_node_bytes_out_rate   | Outbound Traffic        | Outbound traffic per second<br>Unit: byte/s, KB/s, MB/s, or GB/s                              | > 0             | RabbitMQ instance node | 1 minute                     |
| rabbitmq_node_queues           | Queues                  | Number of queues in the node<br>Unit: Count   | > 0             | RabbitMQ instance node | 1 minute                     |
| rabbitmq_memory_high_watermark | Memory High Watermark   | Whether the node has reached the memory high watermark, blocking all producers in the cluster | 1: yes<br>0: no | RabbitMQ instance node | 1 minute                     |
| rabbitmq_disk_insufficient     | Disk High Watermark     | Whether the node has reached the disk high watermark, blocking all producers in the cluster   | 1: yes<br>0: no | RabbitMQ instance node | 1 minute                     |

## Queue Metrics

Table 12-3 Queue metrics

| Metric ID                     | Metric Name             | Description  | Value Range  | Monitored Object        | Monitoring Period (Raw Data) |
|-------------------------------|-------------------------|--|--------------|-------------------------|------------------------------|
| queue_messages_unacknowledged | Unacknowledged Messages | Number of messages that have been consumed but not acknowledged in the RabbitMQ queue<br>Unit: Count | 0–10,000,000 | RabbitMQ instance queue | 1 minute                     |
| queue_messages_ready          | Available Messages      | Number of messages that can be retrieved in a RabbitMQ queue<br>Unit: Count                          | 0–10,000,000 | RabbitMQ instance queue | 1 minute                     |

## Dimensions

| Key                  | Value                   |
|----------------------|-------------------------|
| rabbitmq_instance_id | RabbitMQ instance       |
| rabbitmq_node        | RabbitMQ instance node  |
| rabbitmq_queue       | RabbitMQ instance queue |

## 12.3 Configuring RabbitMQ Alarms

This section describes the alarm rules of some metrics and how to configure the rules. In actual scenarios, you are advised to configure alarm rules for metrics by referring to the following alarm policies.



**Table 12-4** RabbitMQ instance metrics and alarm policies

| Metric                | Alarm Policy  | Description  | Solution   |
|-----------------------|---|--|--|
| Memory High Watermark | Alarm threshold:<br>Raw data $\geq 1$<br>Number of consecutive periods: 1<br>Alarm severity: Critical                               | A threshold of 1 indicates that the memory high watermark is reached, blocking message publishing.           | <ul style="list-style-type: none"> <li>Accelerate message retrieval.</li> <li>Use publisher confirms and monitor the publishing rate and duration on the publishing end. When the duration increases significantly, apply flow control.</li> </ul> |
| Disk High Watermark   | Alarm threshold:<br>Raw data $\geq 1$<br>Number of consecutive periods: 1<br>Alarm severity: Critical                               | A threshold of 1 indicates that the disk high watermark is reached, blocking message publishing.             | <ul style="list-style-type: none"> <li>Reduce the number of messages accumulated in lazy queues.</li> <li>Reduce the number of messages accumulated in durable queues.</li> <li>Delete queues.</li> </ul>  |
| Memory Usage          | Alarm threshold:<br>Raw data $>$ Expected usage (30% is recommended)<br>Number of consecutive periods: 3-5<br>Alarm severity: Major | To prevent high memory watermarks from blocking publishing, configure an alarm for this metric on each node. | <ul style="list-style-type: none"> <li>Accelerate message retrieval.</li> <li>Use publisher confirms and monitor the publishing rate and duration on the publishing end. When the duration increases significantly, apply flow control.</li> </ul> |
| CPU Usage             | Alarm threshold:<br>Raw data $>$ Expected usage (70% is recommended)<br>Number of consecutive periods: 3-5<br>Alarm severity: Major | A high CPU usage may slow down publishing rate. Configure an alarm for this metric on each node.             | <ul style="list-style-type: none"> <li>Reduce the number of mirrored queues.</li> <li>For a cluster instance, add nodes and rebalance queues between all nodes.</li> </ul>   |

| Metric             | Alarm Policy   | Description   | Solution   |
|--------------------|--|---|--|
| Available Messages | Alarm threshold:<br>Raw data > Expected number of available messages<br>Number of consecutive periods: 1<br>Alarm severity: Major      | If the number of available messages is too large, messages are accumulated.           | See <a href="#">the solution to preventing message accumulation</a> .  |
| Unacked Messages   | Alarm threshold:<br>Raw data > Expected number of unacknowledged messages<br>Number of consecutive periods: 1<br>Alarm severity: Major | If the number of unacknowledged messages is too large, messages may be accumulated.   | <ul style="list-style-type: none"> <li>• Check whether the consumer is abnormal.</li> <li>• Check whether the consumer logic is time-consuming.</li> </ul> |
| Connections        | Alarm threshold:<br>Raw data > Expected number of connections<br>Number of consecutive periods: 1<br>Alarm severity: Major             | A sharp increase in the number of connections may be a warning of a traffic increase. | The services may be abnormal. Check whether other alarms exist.  |
| Channels           | Alarm threshold:<br>Raw data > Expected number of channels<br>Number of consecutive periods: 1<br>Alarm severity: Major                | A sharp increase in the number of channels may be a warning of a traffic increase.    | The services may be abnormal. Check whether other alarms exist.  |


| Metric           | Alarm Policy  | Description   | Solution  |
|------------------|---|---|---|
| Erlang Processes | Alarm threshold:<br>Raw data ><br>Expected number<br>of processes<br><br>Number of<br>consecutive<br>periods: 1<br><br>Alarm severity:<br>Major | A sharp increase in the number of processes may be a warning of a traffic increase. | The services may be abnormal. Check whether other alarms exist. |

 **NOTE**

- Set the alarm threshold based on the service expectations. For example, if the expected usage is 35%, set the alarm threshold to 35%.
- The number of consecutive periods and alarm severity can be adjusted based on the service logic.

## Configuring RabbitMQ Alarms

**Step 1** Log in to the console.

**Step 2** In the upper left corner, click  and select a region.


 **NOTE**

Select the region where your RabbitMQ instance is.

**Step 3** Click  and choose **Application > Distributed Message Service for RabbitMQ** to open the console of DMS for RabbitMQ.

**Step 4** View the instance metrics using either of the following methods:

- In the row containing the desired instance, click **View Metric**. On the Cloud Eye console, view the metrics of the instance, nodes, and queues. Metric data is reported to Cloud Eye every minute.
- Click the desired RabbitMQ instance to view its details. In the navigation pane, choose **Monitoring**. On the displayed page, view the metrics of the instance, nodes, and queues. Metric data is updated every minute.

**Step 5** Hover the mouse pointer over a metric and click  to create an alarm rule for the metric.

**Step 6** Specify the alarm rule details.

For more information about creating alarm rules, see [Creating an Alarm Rule](#).

1. Enter the alarm name and description.
2. Specify the alarm policy and alarm severity.

For example, an alarm can be triggered and notifications can be sent once every day if the raw value of connections exceeds the preset value for three consecutive periods and no actions are taken to handle the exception.

3. Set **Alarm Notification** configurations. If you enable **Alarm Notification**, set the validity period, notification object, and trigger condition.
4. Click **Create**.

----End

# 13 Viewing RabbitMQ Audit Logs

With Cloud Trace Service (CTS), you can record operations associated with DMS for RabbitMQ for later query, audit, and backtrack operations.

## Prerequisite

CTS has been enabled.

## DMS for RabbitMQ Operations Supported by CTS

**Table 13-1** DMS for RabbitMQ operations supported by CTS

| Operation   | Resource Type | Trace Name                             |
|---|---------------|--|
| Successfully deleting a background task                     | rabbitmq      | deleteDMSBackendJobSuccess             |
| Failing to delete a background task                         | rabbitmq      | deleteDMSBackendJobFailure             |
| Successfully scaling up an instance                         | rabbitmq      | extendDMSInstanceSuccess               |
| Failing to scale up an instance                             | rabbitmq      | extendDMSInstanceFailure               |
| Successfully resetting instance password                    | rabbitmq      | resetDMSInstancePasswordSuccess        |
| Failing to reset instance password                          | rabbitmq      | resetDMSInstancePasswordFailure        |
| Successfully deleting an instance that failed to be created | rabbitmq      | deleteDMSCreateFailureInstancesSuccess |
| Failing to delete an instance that failed to be created     | rabbitmq      | deleteDMSCreateFailureInstancesFailure |

| Operation  | Resource Type | Trace Name                     |
|--|---------------|--------------------------------|
| Successfully restarting an instance                          | rabbitmq      | restartDMSInstanceSuccess      |
| Failing to restart an instance                               | rabbitmq      | restartDMSInstanceFailure      |
| Successfully deleting multiple instances at a time           | rabbitmq      | batchDeleteDMSInstanceSuccess  |
| Failing to delete multiple instances at a time               | rabbitmq      | batchDeleteDMSInstanceFailure  |
| Successfully restarting multiple instances at a time         | rabbitmq      | batchRestartDMSInstanceSuccess |
| Failing to restart multiple instances at a time              | rabbitmq      | batchRestartDMSInstanceFailure |
| Successfully modifying instance information                  | rabbitmq      | modifyDMSInstanceInfoSuccess   |
| Failing to modify instance information                       | rabbitmq      | modifyDMSInstanceInfoFailure   |
| Successfully deleting multiple instance tasks at a time      | rabbitmq      | batchDeleteDMSInstanceTask     |
| Successfully deleting an instance task                       | rabbitmq      | deleteDMSInstanceTaskSuccess   |
| Failing to delete an instance task                           | rabbitmq      | deleteDMSInstanceTaskFailure   |
| Successfully creating an instance task                       | rabbitmq      | createDMSInstanceTaskSuccess   |
| Failing to create an instance task                           | rabbitmq      | createDMSInstanceTaskFailure   |
| Successfully submitting a request for scaling up an instance | rabbitmq      | extendDMSInstanceTaskSuccess   |
| Failing to submit a request for scaling up an instance       | rabbitmq      | extendDMSInstanceTaskFailure   |

| Operation   | Resource Type | Trace Name                          |
|---|---------------|-------------------------------------|
| Successfully submitting a request for restarting an instance                  | rabbitmq      | restartDMSInstanceTaskSuccess       |
| Failing to submit a request for restarting an instance                        | rabbitmq      | restartDMSInstanceTaskFailure       |
| Successfully submitting a request for restarting multiple instances at a time | rabbitmq      | batchRestartDMSInstanceTask-Success |
| Failing to submit a request for restarting multiple instances at a time       | rabbitmq      | batchRestartDMSInstanceTask-Failure |
| Successfully submitting a request for modifying instance information          | rabbitmq      | modifyDMSInstanceInfoTaskSuccess    |
| Failing to submit a request for modifying instance information                | rabbitmq      | modifyDMSInstanceInfoTaskFailure    |

## Viewing Audit Logs

See [Querying Real-Time Traces](#).