Distributed Message Service for RabbitMQ

User Guide

Issue 01

Date 2025-07-31

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

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.

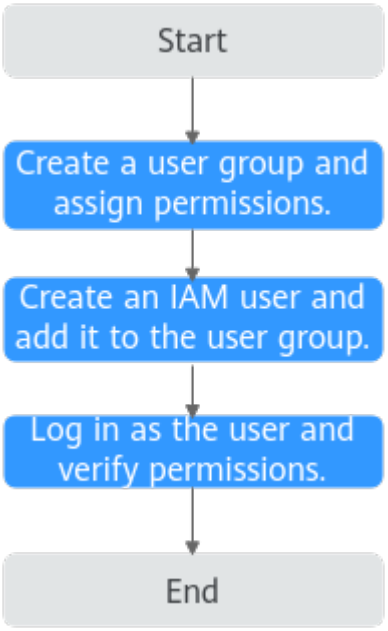
Prerequisites

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**.

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

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 and add it to the created user group**.
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.

- 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.

  For example, if you want to assign all of the permissions of the **DMS FullAccess** policy to a user, except for deleting instances, you can create a custom policy to deny only instance deletion. When you apply both the **DMS FullAccess** policy and the custom policy denying instance deletion, since "Deny" always takes precedence over "Allow", the "Deny" will be applied for that one conflicting permission. The user will then be able to perform all operations on instances except deleting instances. The following is an example of a deny policy:

```
{
    "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	[Format] DMS:*:*: rabbitmq: *instance ID* [Note] For instance resources, IAM automatically generates the prefix (**DMS:*:*:rabbitmq:**) of the resource path. For the path of a specific instance, add the *instance ID* to the end. You can also use an asterisk * to indicate any instance. For example: **DMS:*:*:rabbitmq:*** indicates any RabbitMQ instance.

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, you can use the condition key **dms:ssl** to check whether SSL is enabled for a RabbitMQ instance. The following table lists the predefined condition keys of DMS for RabbitMQ.

Table 2-2 Predefined condition keys of DMS for RabbitMQ

Condition Key	Operator	Description
dms:publicIP	Bool Null	Whether public access is enabled
dms:ssl	Bool Null	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 Required Resources

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 shared ones, or create new ones. VPC owners can share the subnets in a VPC with one or multiple accounts through Resource Access Manager (RAM). Through VPC sharing, you can easily configure, operate, and manage multiple accounts' resources at low costs. For more information about VPC and subnet sharing, see **VPC Sharing**. Note: VPCs must be created in the same region as the RabbitMQ instance.	For details on how to create a VPC and subnet, see **Creating a VPC and Subnet**. If you need to create and use a new subnet in an existing VPC, see **Creating a Subnet for an Existing VPC**.

Resource Name	Requirement	Reference
Security group	Different RabbitMQ instances can use the same or different security groups. The security group must be in the same region as the RabbitMQ instance. Before accessing a RabbitMQ instance, configure security groups based on the access mode. For details, see **Table 5-2** or **Table 5-3**.	For details on how to create a security group, see **Creating a Security Group**. For details on how to add rules to a security group, see **Adding a Security Group Rule**.
EIP	To access a RabbitMQ instance on a client over a public network, create EIPs in advance. Note the following when creating EIPs: • The EIPs must be created in the same region as the RabbitMQ instance. • **The RabbitMQ console cannot identify IPv6 EIPs.**	For details about how to create an EIP, see **Assigning an EIP**.

Buying a RabbitMQ Instance

DMS for RabbitMQ provides multiple options for you to purchase RabbitMQ instances.

Table 3-2 Purchasing a RabbitMQ instance

How to Purchase	Scenario
Quickly Configuring a RabbitMQ Instance	For a quick purchase, DMS for RabbitMQ offers preconfigured instance specifications.
Customizing a RabbitMQ 3.x.x Instance	For a standard purchase, you can customize a single-node or cluster RabbitMQ 3.x.x instance as required.
Customizing a RabbitMQ AMQP-0-9-1 Instance	For a standard purchase, you can customize a single-node or cluster RabbitMQ AMQP-0-9-1 instance as required.

Quickly Configuring a RabbitMQ Instance

Step 1 Go to the **Buy Instance page**.

Step 2 Set basic instance configurations on the **Quick Config** tab page.

Table 3-3 Basic instance configuration parameters

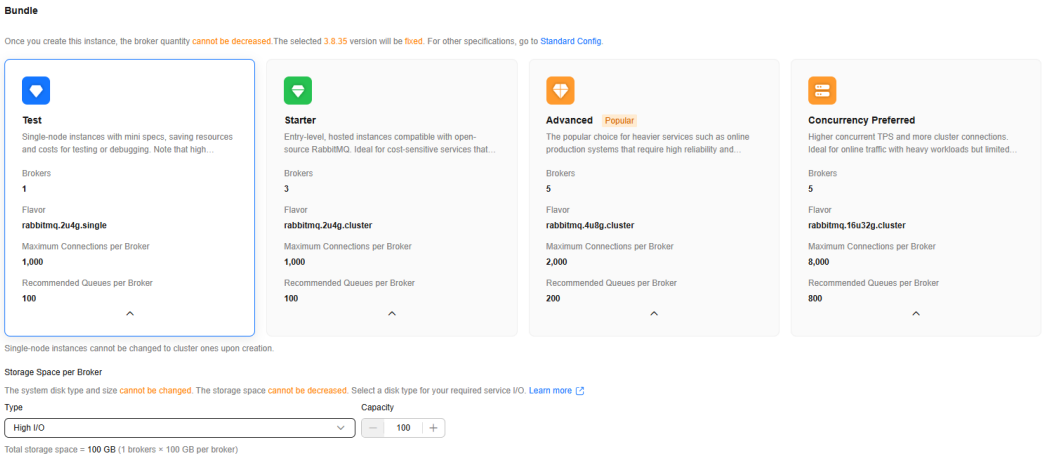
Parameter	Description
Billing Mode	• Yearly/Monthly is a prepaid mode. You need to pay first, and will be billed for your subscription period. • Pay-per-use is a postpaid mode. You can pay after using the service, and will be billed for your usage duration. The fees are calculated in seconds and settled by hour.
Region	DMS for RocketMQ instances in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access.
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 AZ or at least three AZs. The AZ setting is fixed once the instance is created.

Step 3 Set the bundle.

DMS for RabbitMQ provides preconfigured specifications. You can select one as required. Specify the disk type and capacity as required. **The disk type is fixed once the instance is created.**

The disk supports high I/O, ultra-high I/O, Extreme SSD, and General Purpose SSD types. For more information, see **Disk Types and Performance**.

Figure 3-1 Bundle



Step 4 Set the network.

Table 3-4 Instance network parameters

Parameter	Description
VPC	Select a created or shared VPC. A VPC provides an isolated virtual network for your RabbitMQ instances. You can configure and manage the network as required. You can click **Manage VPCs** on the right to go to the VPC console, and view or create VPCs. After the RabbitMQ instance is created, its VPC cannot be changed.
Subnet	Select a created or shared subnet. After the RabbitMQ instance is created, its subnet cannot be changed.
Security Group	Select a created security group. A security group is a set of rules for accessing a RabbitMQ instance. You can click **Manage Security Group** to view or create security groups on the network console. Before accessing a RabbitMQ instance on the client, configure security group rules based on the access mode. For details about security group rules, see **Table 5-2**.

Step 5 Set the instance access mode.

Table 3-5 Instance access mode parameters

Parameter	Description
Public Network Access	Clients can access RabbitMQ instances with public access enabled through elastic IP addresses (EIPs). Enabling public access requires EIPs. If EIPs are insufficient, click **Create Elastic IP** to create EIPs. Then, return to the RabbitMQ console and click next to **Public IP Addresses** to refresh the elastic IP address list. **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.
Encryption Mode	Enabling SSL secures data transmission with encryption. **Once the instance is created, SSL cannot be manually configured.**

Step 6 Set the instance authentication.

Table 3-6 Instance authentication parameters

Parameter	Description
RabbitMQ Authentication Username	Enter the username used for accessing the instance. A username should contain 4 to 64 characters, start with a letter, and contain only letters, digits, hyphens (-), and underscores (_).
Password	Enter the password used for accessing the instance. 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 7 Configure advanced settings.

Table 3-7 Advanced setting parameters

Parameter	Description
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 (_).
Enterprise Project	Available for enterprise users. Enterprise projects facilitate project-level management and grouping of cloud resources and users. The default project is **default**.

Parameter	Description
Tags	Tags are used to identify cloud resources. When you have many cloud resources of the same type, you can use tags to classify cloud resources by dimension (for example, usage, owner, or environment). If your organization has configured tag policies for DMS for RabbitMQ, add tags to RabbitMQ instances based on the tag policies. If a tag does not comply with the tag policies, RabbitMQ instance creation may fail. Contact your organization administrator to learn more about tag policies. • If you have created predefined tags, select a predefined pair of tag key and value. You can click **Create 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**.
Description	Set the description of the instance for up to 1024 characters.

Step 8 Select the required duration.

This parameter is displayed only if the billing mode is yearly/monthly. If **Auto-renew** is selected, the instance will be renewed automatically.

- Monthly subscriptions auto-renew for 1 month every time.
- Yearly subscriptions auto-renew for 1 year every time.

Step 9 Click **Confirm**.

Step 10 Confirm the instance information, and read and agree to the *Huawei Cloud Customer Agreement*. If you have selected **Yearly/Monthly** for **Billing Mode**, click **Pay Now** and make the payment as prompted. If you have selected the pay-per-use mode, click **Submit**.

Step 11 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 **Failed** state, delete it by referring to **Deleting a RabbitMQ Instance** and try purchasing another one. If the purchase fails a second time, contact customer service.

----End

Customizing a RabbitMQ Instance (3.x.x)

Step 1 Go to the **Buy Instance page**.

Step 2 Set basic instance configurations on the **Standard Config** tab page.

Table 3-8 Basic instance configuration parameters

Parameter	Description
Billing Mode	• Yearly/Monthly is a prepaid mode. You need to pay first, and will be billed for your subscription period. • Pay-per-use is a postpaid mode. You can pay after using the service, and will be billed for your usage duration. The fees are calculated in seconds and settled by hour.
Region	DMS for RocketMQ instances in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access.
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 AZ or at least three AZs. The AZ setting is fixed once the instance is created.

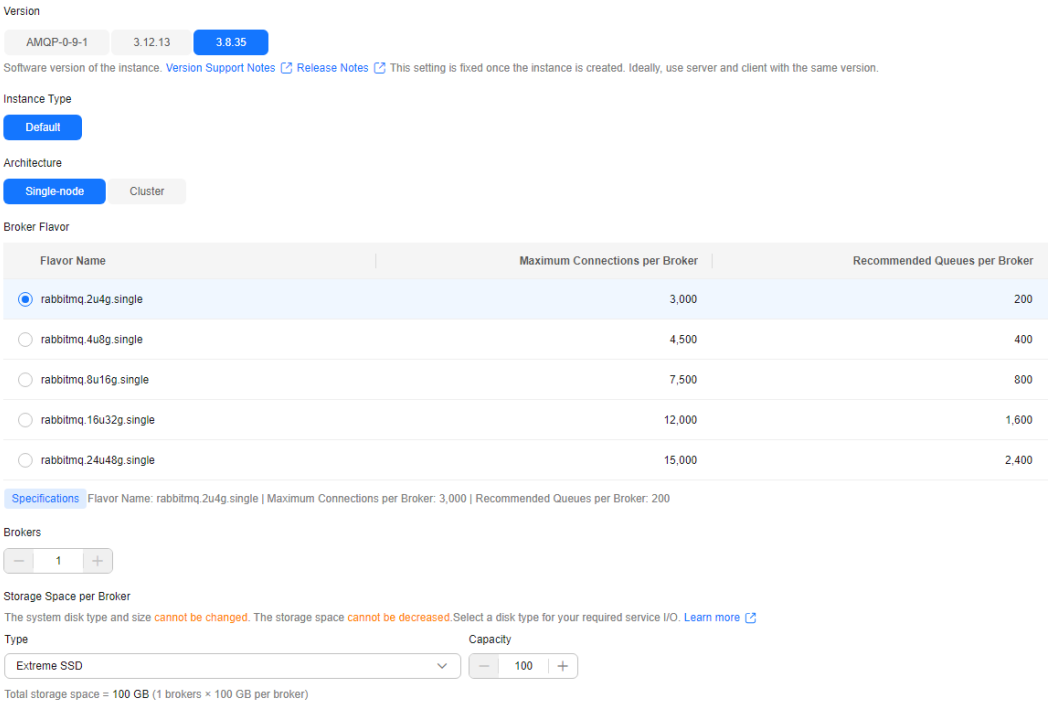
Step 3 Configure the following instance parameters:

Table 3-9 Instance specifications parameters

Parameter	Description
Version	RabbitMQ version. Select **3.8.35** and **3.12.13**. **The version is fixed once the instance is created.** The 3.12.13 version is available only in certain regions. See the console.
Architecture	**Single-node** or **Cluster** are available. • **Single-node**: There is only one RabbitMQ broker. • **Cluster**: There are multiple RabbitMQ brokers, achieving highly reliable message storage.
Broker Flavor	Select a broker flavor as required. Learn more about **Specifications**. 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.
Brokers	Specify the broker quantity.

Parameter	Description
Storage Space per Broker	Select the disk type and specify the disk size. **The disk type is fixed once the instance is created.** The disk supports high I/O, ultra-high I/O, Extreme SSD, and General Purpose SSD types. For more information, see **Disk Types and Performance**.

Figure 3-2 V3.x.x instance specifications



Step 4 Set the network.

Table 3-10 Instance network parameters

Parameter	Description
VPC	Select a created or shared VPC. A VPC provides an isolated virtual network for your RabbitMQ instances. You can configure and manage the network as required. You can click **Manage VPCs** on the right to go to the VPC console, and view or create VPCs. After the RabbitMQ instance is created, its VPC cannot be changed.
Subnet	Select a created or shared subnet. After the RabbitMQ instance is created, its subnet cannot be changed.

Parameter	Description
Security Group	Select a created security group. A security group is a set of rules for accessing a RabbitMQ instance. You can click **Manage Security Group** to view or create security groups on the network console. Before accessing a RabbitMQ instance on the client, configure security group rules based on the access mode. For details about security group rules, see **Table 5-2**.

Step 5 Set the instance access mode.

Table 3-11 Instance access mode parameters

Parameter	Description
Public Network Access	Clients can access RabbitMQ instances with public access enabled through elastic IP addresses (EIPs). Enabling public access requires EIPs. If EIPs are insufficient, click **Create Elastic IP** to create EIPs. Then, return to the RabbitMQ console and click next to **Public IP Addresses** to refresh the elastic IP address list. **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.
Encryption Mode	Enabling SSL secures data transmission with encryption. **Once the instance is created, SSL cannot be manually configured.**

Step 6 Set the instance authentication.

Table 3-12 Instance authentication parameters

Parameter	Description
RabbitMQ Authentication Username	Enter the username used for accessing the instance. A username should contain 4 to 64 characters, start with a letter, and contain only letters, digits, hyphens (-), and underscores (_).

Parameter	Description
Password	Enter the password used for accessing the instance. 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 7 Configure advanced settings.

Table 3-13 Advanced setting parameters

Parameter	Description
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 (_).
Enterprise Project	Available for enterprise users. Enterprise projects facilitate project-level management and grouping of cloud resources and users. The default project is **default**.
Tags	Tags are used to identify cloud resources. When you have many cloud resources of the same type, you can use tags to classify cloud resources by dimension (for example, usage, owner, or environment). If your organization has configured tag policies for DMS for RabbitMQ, add tags to RabbitMQ instances based on the tag policies. If a tag does not comply with the tag policies, RabbitMQ instance creation may fail. Contact your organization administrator to learn more about tag policies. • If you have created predefined tags, select a predefined pair of tag key and value. You can click **Create 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**.
Description	Set the description of the instance for up to 1024 characters.

Step 8 Select the required duration.

This parameter is displayed only if the billing mode is yearly/monthly. If **Auto-renew** is selected, the instance will be renewed automatically.

- Monthly subscriptions auto-renew for 1 month every time.
- Yearly subscriptions auto-renew for 1 year every time.

Step 9 In **Summary** on the right, view the selected instance configuration.

Step 10 Click **Confirm**.

Step 11 Confirm the instance information, and read and agree to the *Huawei Cloud Customer Agreement*. If you have selected **Yearly/Monthly** for **Billing Mode**, click **Pay Now** and make the payment as prompted. If you have selected the pay-per-use mode, click **Submit**.

Step 12 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 **Failed** state, delete it by referring to **Deleting a RabbitMQ Instance** and try purchasing another one. If the purchase fails a second time, contact customer service.

----End

Customizing a RabbitMQ Instance (AMQP-0-9-1)

Step 1 Go to the **Buy Instance page**.

Step 2 Set basic instance configurations on the **Standard Config** tab page.

Table 3-14 Basic instance configuration parameters

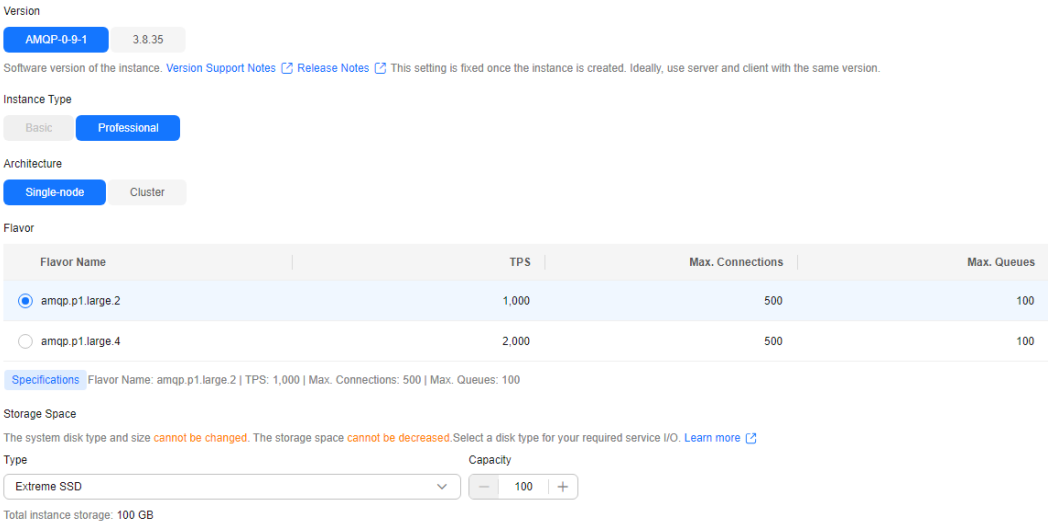
Parameter	Description
Billing Mode	• Yearly/Monthly is a prepaid mode. You need to pay first, and will be billed for your subscription period. • Pay-per-use is a postpaid mode. You can pay after using the service, and will be billed for your usage duration. The fees are calculated in seconds and settled by hour.
Region	DMS for RocketMQ instances in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access.
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 AZ or at least three AZs. The AZ setting is fixed once the instance is created.

Step 3 Configure the following instance parameters:

Table 3-15 Instance specifications parameters

Parameter	Description
Version	RabbitMQ version. Select **AMQP-0-9-1**. **The version is fixed once the instance is created.**
Instance Type	Select **Professional**.
Architecture	**Single-node** or **Cluster** are available. • **Single-node**: There is only one RabbitMQ broker. • **Cluster**: There are multiple RabbitMQ brokers, achieving highly reliable message storage.
Flavor	Select a flavor as required. Learn more about **Specifications**.
Storage Space	Specify the disk type and total storage space of all the brokers. **The disk type is fixed once the instance is created.** The disk supports high I/O, ultra-high I/O, Extreme SSD, and General Purpose SSD types. For more information, see **Disk Types and Performance**.

Figure 3-3 AMQP-0-9-1 instance specifications



Step 4 Set the network.

Table 3-16 Instance network parameters

Parameter	Description
VPC	Select a created or shared VPC. A VPC provides an isolated virtual network for your RabbitMQ instances. You can configure and manage the network as required. You can click **Manage VPCs** on the right to go to the VPC console, and view or create VPCs. After the RabbitMQ instance is created, its VPC cannot be changed.
Subnet	Select a created or shared subnet. After the RabbitMQ instance is created, its subnet cannot be changed.
Security Group	Select a created security group. A security group is a set of rules for accessing a RabbitMQ instance. You can click **Manage Security Group** to view or create security groups on the network console. Before accessing a RabbitMQ instance on the client, configure security group rules based on the access mode. For details about security group rules, see **Table 5-3**.

Step 5 Set the instance access mode.

RabbitMQ AMQP-0-9-1 instances do not support public access.

Step 6 Configure advanced settings.

Table 3-17 Advanced setting parameters

Parameter	Description
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 (_).
Enterprise Project	Available for enterprise users. Enterprise projects facilitate project-level management and grouping of cloud resources and users. The default project is **default**.
ACL	RabbitMQ instances support ACL-based permission isolation among producers and consumers. You can create multiple users, and grant virtual host (resource) permissions to them. When ACL is enabled, message production and consumption require authentication.

Parameter	Description
Tags	Tags are used to identify cloud resources. When you have many cloud resources of the same type, you can use tags to classify cloud resources by dimension (for example, usage, owner, or environment). If your organization has configured tag policies for DMS for RabbitMQ, add tags to RabbitMQ instances based on the tag policies. If a tag does not comply with the tag policies, RabbitMQ instance creation may fail. Contact your organization administrator to learn more about tag policies. • If you have created predefined tags, select a predefined pair of tag key and value. You can click **Create 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**.
Description	Set the description of the instance for up to 1024 characters.

Step 7 Select the required duration.

This parameter is displayed only if the billing mode is yearly/monthly. If **Auto-renew** is selected, the instance will be renewed automatically.

- Monthly subscriptions auto-renew for 1 month every time.
- Yearly subscriptions auto-renew for 1 year every time.

Step 8 In **Summary** on the right, view the selected instance configuration.

Step 9 Click **Confirm**.

Step 10 Confirm the instance information, and read and agree to the *Huawei Cloud Customer Agreement*. If you have selected **Yearly/Monthly** for **Billing Mode**, click **Pay Now** and make the payment as prompted. If you have selected the pay-per-use mode, click **Submit**.

Step 11 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 **Failed** state, delete it by referring to **Deleting a RabbitMQ Instance** and try purchasing another one. If the purchase fails a second time, contact customer service.

----End

Purchasing a RabbitMQ Instance with Same Configurations

To purchase another RabbitMQ instance with the same configuration as the current one, reuse the current configuration through the **Buy Another** function.

Step 1 Go to the **RabbitMQ console**.

Step 2 Select a target RabbitMQ instance and choose **More** > **Buy Another** in the **Operation** column.

Step 3 Adjust the automatically replicated parameter settings as required. For details, see **Buying a RabbitMQ Instance**.

For security purposes, parameter settings involved in the following scenarios will not be replicated and a re-configuration is required:

- Username and password for a client to access a RabbitMQ instance.
- Public IP addresses of a RabbitMQ instance with public access enabled.
- Name of a target RabbitMQ instance.

Step 4 In **Summary** on the right, view the selected instance configuration.

Step 5 Click **Confirm**.

Step 6 Confirm the instance information, and read and agree to the *Huawei Cloud Customer Agreement*. If you have selected the yearly/monthly billing mode, click **Pay Now** and make the payment as prompted. If you have selected the pay-per-use mode, click **Submit**.

Step 7 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 **Failed** state, delete it by referring to **Deleting a RabbitMQ Instance** and try purchasing another one. If the purchase fails a second time, contact customer service.

----End

Related Document

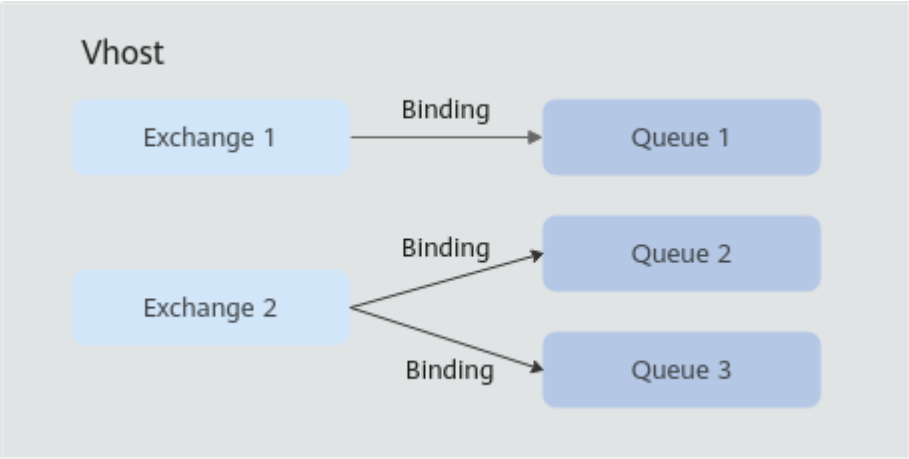
To create a RabbitMQ instance by calling an API, see **Creating an Instance**.

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



Notes and Constraints

- After an instance is created, a virtual host named **/** is automatically created.
- After an instance is created, a virtual host named **/** is automatically created in RabbitMQ 3.x.x and a virtual host named **default** is automatically created in RabbitMQ AMQP-0-9-1.
- When a virtual host name of a RabbitMQ 3.x.x instance starts with a special character, such as a period (.), the monitoring data may not be displayed. When a virtual host name contains special characters such as percent (%), vertical bar (|), or slash (/), this name is not consistently displayed on the

monitoring page. The special characters are displayed as underscores (_) instead. For example, virtual host **Vhost.1%1|2_3/** is displayed as **Vhost.1_1_2_3_** in monitoring.

- When a virtual host name of a RabbitMQ AMQP-0-9-1 instance contains special characters such as percent (%), vertical bar (|), slash (/), and period (.), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_). For example, virtual host **Vhost.1%1|2_3/** is displayed as **Vhost_1_1_2_3_** in monitoring.
- Available on the console or management UI for RabbitMQ 3.x.x, and only on the console for RabbitMQ AMQP-0-9-1.

Creating a RabbitMQ Virtual Host

A virtual host can be created on the console or management UI.

Creating a RabbitMQ Host (Console)

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click **Create Virtual Host**.

Step 6 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. This parameter is available only in RabbitMQ 3.x.x. If it is enabled, you can trace the message forwarding path.

Figure 4-2 Virtual host list (RabbitMQ 3.x.x)

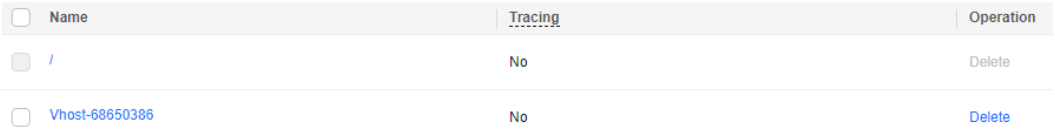
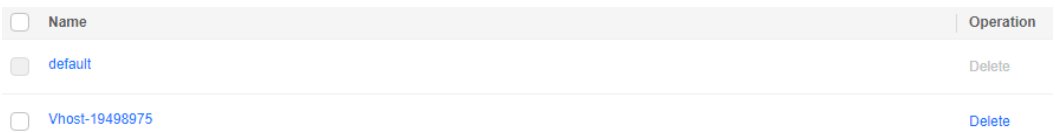


Figure 4-3 Virtual hosts (RabbitMQ AMQP-0-9-1)



----End

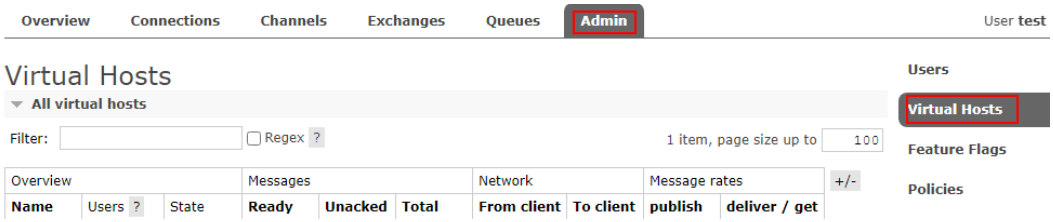
Creating a RabbitMQ Virtual Host (Management UI)

Step 1 Log in to the **RabbitMQ management UI**.

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

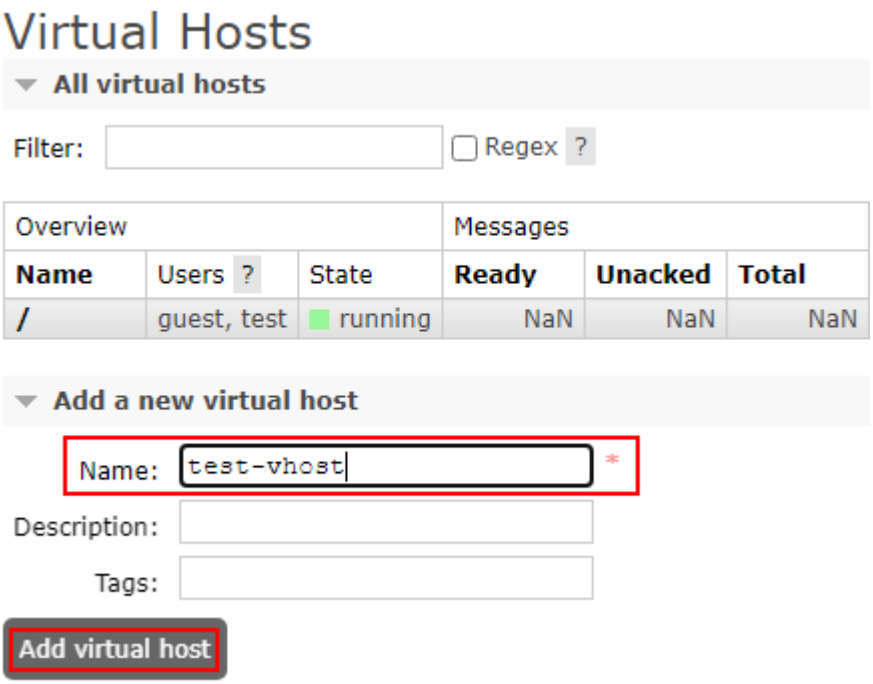
Step 3 In the navigation tree on the right, choose **Virtual Hosts**.

Figure 4-4 Virtual hosts



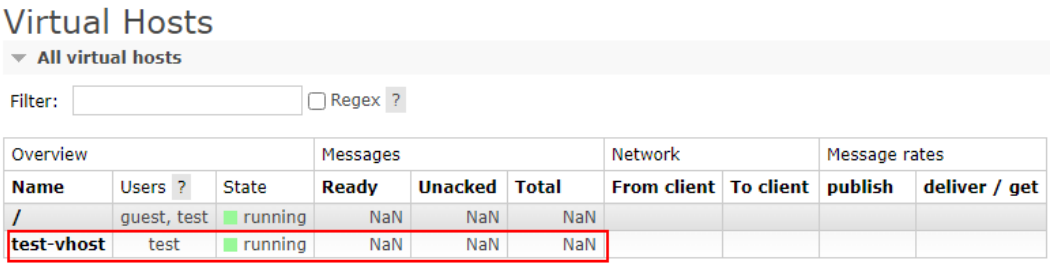
Step 4 In the **Add a new virtual host** area, enter the virtual host name and click **Add virtual host**.

Figure 4-5 Creating a virtual host (management UI)



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

Figure 4-6 Virtual host list (management UI)



----End

Related Document

To create a virtual host by calling an API, see **Creating a Virtual Host**.

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.

Notes and Constraints

- 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.
- When an exchange name of a RabbitMQ AMQP-0-9-1 instance contains special characters such as percent (%), vertical bar (|), slash (/), andperiod (.), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_). For example, exchange **Exchange.1%1|2_3/** is displayed as **Exchange_1_1_2_3_** in monitoring.

Prerequisite

A virtual host has been created.

Creating a RabbitMQ Exchange

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Exchange** tab page, click **Create Exchange**. The **Create Exchange** dialog box is displayed.

Step 7 Configure the exchange name and other parameters by referring to **Table 4-1**.

Table 4-1 Exchange parameters

Parameter	Description
Name	When creating an exchange, you can modify the automatically generated exchange name. Naming rules: 3–128 characters and only letters, digits, periods (.), percent signs (%), vertical bars (|), hyphens (-), underscores (_), or slashes (/). The name cannot be changed once the exchange is created.
Type	Select a routing type. For details, see **Exchanges**. • **direct**: Exchanges route messages to matching queues based on the routing keys. • **fanout**: Exchanges route messages to all bound queues. • **topic**: Exchanges route messages to queues based on routing key wildcard matching. • **headers**: 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). • **x-delayed-message**: Exchanges delay message delivery. Delayed messages will be routed by the exchange type-specified rules. • **x-consistent-hash**: Exchanges calculate a hash value based on a routing key, and route a message to the hashed queue.
x-delayed-type	This parameter is displayed only when **Type** is set to **x-delayed-message**. Messages are routed by x-delayed-type-specified rules. • **direct**: Exchanges route messages to matching queues based on the routing keys. • **fanout**: Exchanges route messages to all bound queues. • **topic**: Exchanges route messages to queues based on routing key wildcard matching. • **headers**: 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).

Parameter	Description
Auto-Delete	Indicates whether to enable automatic exchange deletion. • Enabled: The exchange will be automatically deleted when the last bound queue unbound from the exchange. • Disabled: The exchange will not be deleted when the last bound queue unbound from the exchange.
Persistence	This parameter is mandatory for RabbitMQ 3.x.x instances, and enabled by default for RabbitMQ AMQP-0-9-1 ones. Indicates whether to enable exchange persistence. • Enabled: The exchange survives server restart. • Disabled: The exchange will be deleted after server restarts and needs to be recreated.
Internal	Only RabbitMQ 3.x.x instances have this parameter. Indicates whether exchanges are for internal use. • Yes: An exchange can only bind another exchange instead of a queue. • No: Exchanges can bind exchanges and queues.

Step 8 Click **OK**.

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

----End

Related Document

To create an exchange by calling an API, see **Creating an Exchange**.

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.
- In RabbitMQ AMQP-0-9-1, exchanges can only be bound with queues and not exchanges.
- **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 **RabbitMQ console**.

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange. The **Bind Exchange** page is displayed.

Step 7 Click **Add Binding**. The **Add Binding** dialog box is displayed.

Step 8 Set the parameters by referring to **Table 4-2**.

Table 4-2 Binding parameters

Parameter	Description
Type	This parameter is only available in RabbitMQ 3.x.x. RabbitMQ AMQP-0-9-1 supports only queue bindings. Select the binding type: Select **Exchange**.
Target	Select a target exchange to be bound.
Routing Key	Enter a key string to inform the exchange of which target exchanges to deliver messages to. • This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to target exchanges with the routing keys matched. Messages cannot be routed to target exchanges without a routing key. • Enabling x-consistent-hash exchanges requires a routing key which indicates the weight of a queue. The larger the key, the heavier the weight, which means that the queue receives more messages. • For fanout exchanges and header exchanges, skip this parameter.

Step 9 Click **OK**.

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

----End

Binding Source Exchanges to a Queue

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Queue** tab page, click **View Detail** in the row containing the desired queue.

Step 7 On the **Bindings** tab, click **Add Binding**. The **Add Binding** dialog box is displayed.

Step 8 Set the parameters by referring to **Table 4-3**.

Table 4-3 Binding parameters

Parameter	Description
Source	Select an exchange to be bound.
Routing Key	Enter a key string to inform the exchange of which queues to deliver messages to. • This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to queues with the routing keys matched. Messages cannot be routed to target queues without a routing key. • Enabling x-consistent-hash exchanges requires a routing key which indicates the weight of a queue. The larger the key, the heavier the weight, which means that the queue receives more messages. • For fanout exchanges and header exchanges, skip this parameter.

Step 9 Click **OK**.

On the **Bindings** tab, view the new exchange.

----End

Related Document

To bind an exchange by calling an API, see **Adding a Binding**.

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.

Notes and Constraints

- When a queue name of a RabbitMQ 3.x.x instance starts with a special character, such as a period (.), the monitoring data may not be displayed. When a queue name contains special characters such as percent (%), vertical bar (|), and slash (/), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_) instead. For example, queue **Queue.1%1|2_3/** is displayed as **Queue.1_1_2_3_** in monitoring.
- When a queue name of a RabbitMQ AMQP-0-9-1 instance contains special characters such as percent (%), vertical bar (|), slash (/), and period (.), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_). For example, queue **Queue.1%1|2_3/** is displayed as **Queue_1_1_2_3_** in monitoring.

Prerequisite

A virtual host has been created.

Creating a RabbitMQ Queue

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Queue** tab page, click **Create Queue**. The **Create Queue** dialog box is displayed.

Step 7 Configure the queue name and other parameters by referring to **Table 4-4**.

Table 4-4 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, periods (.), percent signs (%), vertical bars (|), hyphens (-), underscores (_), or slashes (/). The name cannot be changed once the queue is created.

Parameter	Description
Persistence	This parameter is mandatory for RabbitMQ 3.x.x instances, and enabled by default for RabbitMQ AMQP-0-9-1 instances. Indicates whether to enable queue persistence. • Enabled: The queue survives after server restart. • Disabled: The queue will be deleted after server restart and needs to be recreated.
Auto-Delete	Indicates whether to enable automatic queue deletion. • Yes: The queue will be automatically deleted when the last consumer unsubscribes from the queue. • No: The queue will not be deleted when the last consumer unsubscribes from the queue.
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.
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.
Highest Priority	Only RabbitMQ AMQP-0-9-1 has this parameter. Priority of a queue. Range: 1–9. The larger the value, the higher the priority.
Lazy Queue	Only available for RabbitMQ 3.x.x instances. Enter **lazy** to make the queue lazy. 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 8 Click **OK**.

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

----End

Related Document

To create a queue by calling an API, see **Creating a Queue**.

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 **RabbitMQ console**.

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange. The **Bind Exchange** page is displayed.

Step 7 Click **Add Binding**. The **Add Binding** dialog box is displayed.

Step 8 Set the parameters by referring to **Table 4-5**.

Table 4-5 Binding parameters

Parameter	Description
Type	Only RabbitMQ 3.x.x instances have this parameter. RabbitMQ AMQP-0-9-1 instances only support binding queues. Select the binding type: To bind a queue, select **Queue**.
Target	Select a target queue to be bound.

Parameter	Description
Routing Key	Enter a key string to inform the exchange of which queues to deliver messages to. • **This parameter is required by direct exchanges and topic exchanges.** Such exchanges route messages to queues with the routing keys matched. Messages cannot be routed to target queues without a routing key. • **Enabling x-consistent-hash exchanges requires a routing key** which indicates the weight of a queue. The larger the key, the heavier the weight, which means that the queue receives more messages. • **For fanout exchanges and header exchanges, skip this parameter.**

Step 9 Click **OK**.

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

----End

Related Document

To bind a queue by calling an API, see **Adding a Binding**.

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 **RabbitMQ console**.

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 View the sum of exchanges or queues in the top area and their details on the **Exchange** and **Queue** tab pages. **Table 4-6** describes exchange details. **Table 4-7** describes queue details.

Figure 4-7 Virtual host details page (for RabbitMQ AMQP-0-9-1)

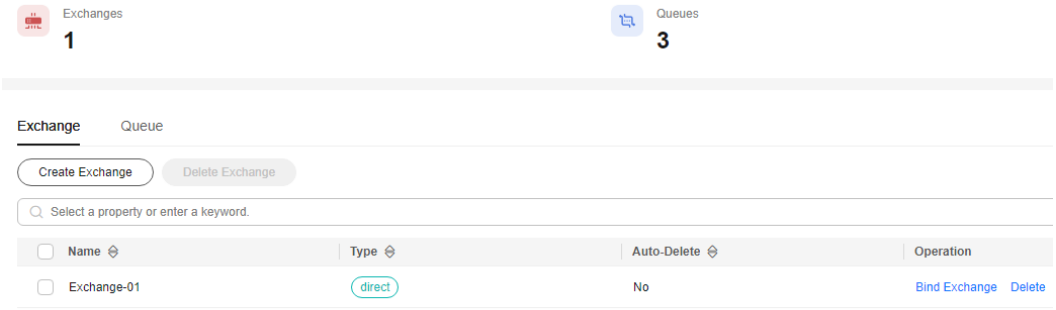


Table 4-6 Exchange details

Parameter	Description
Name	Name of an exchange.
Default	Available only for RabbitMQ 3.x.x. Whether this exchange is created by the system. • Yes: This type of exchange cannot be deleted. • No: This type of exchange is manually created, and can be deleted.
Target Type	Type of an exchange. • **direct**: Exchanges route messages to matching queues based on the routing keys. • **fanout**: Exchanges route messages to all bound queues. • **topic**: Exchanges route messages to queues based on routing key wildcard matching. • **headers**: 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). • **x-delayed-message**: Exchanges delay message delivery. Delayed messages will be routed by the exchange type-specified rules. • **x-consistent-hash**: Exchanges calculate a hash value based on a routing key, and route a message to the hashed queue.
Persistence	Available only for RabbitMQ 3.x.x. Whether this exchange supports persistence. • Yes: The exchange survives server restart. • No: The exchange will be deleted after server restarts and needs to be recreated.

Parameter	Description
Internal	Available only for RabbitMQ 3.x.x. Whether this exchange is for internal use. • Yes: This exchange can only bind another exchange instead of a queue. • No: This exchange can bind exchanges and queues.
Automatically delete	Whether this exchange can be automatically deleted. • Yes: This exchange will be automatically deleted when the last bound queue unbound from the exchange. • No: This exchange will not be deleted when the last bound queue unbound from the exchange.

Table 4-7 Queue details

Parameter	Description
Name	Name of a queue.
Accumulated Messages	Number of accumulated messages in this queue.
Persistence	This parameter is displayed only for RabbitMQ 3.x.x. Whether this queue supports persistence. • Yes: This queue survives after server restart. • No: This queue will be deleted after server restart and needs to be recreated.
Auto-Delete	Whether this queue can be automatically deleted. • Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue. • No: This queue will not be deleted when the last consumer unsubscribes from the queue.
Policy	This parameter is displayed only for RabbitMQ 3.x.x. Name of the policy set for this queue.

----End

4.6.2 Deleting RabbitMQ Virtual Hosts

This section describes how to delete virtual hosts. Methods of deleting virtual hosts:

- **Deleting Virtual Hosts (Console)**
- **Deleting Virtual Hosts (RabbitMQ Management UI)**

Notes and Constraints

- The default virtual host created in instance creation cannot be deleted.
- Deleting a virtual host removes all its resources including exchanges and queues permanently.
- Available on the console and management UI for RabbitMQ 3.x.x, and only on the console for RabbitMQ AMQP-0-9-1.

Procedure

The following describes how to delete a virtual host.

Deleting Virtual Hosts (Console)

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 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.

WARNING

Deleting a virtual host removes all its resources including exchanges and queues permanently.

Step 6 In the displayed dialog box, click **OK**.

This virtual host is deleted when it is removed from the virtual host list.

----End

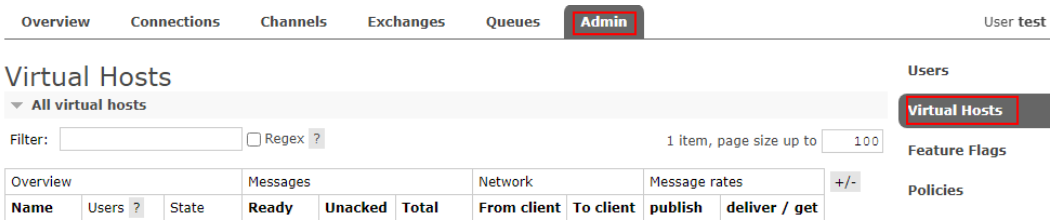
Deleting Virtual Hosts (RabbitMQ Management UI)

Step 1 Log in to the **RabbitMQ management UI**.

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

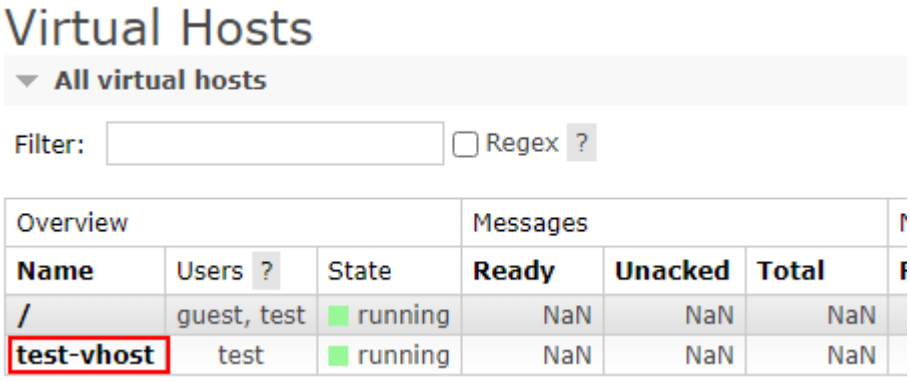
Step 3 In the navigation tree on the right, choose **Virtual Hosts**.

Figure 4-8 Virtual Hosts page



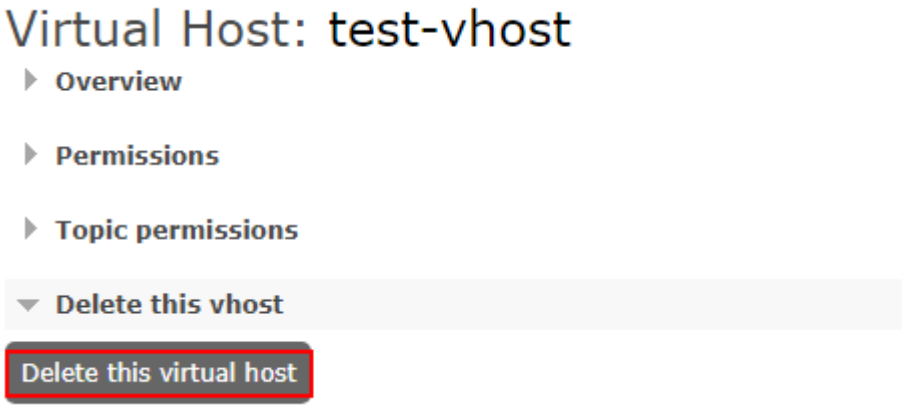
Step 4 Click the name of the virtual host to be deleted.

Figure 4-9 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 4-10 Deleting a virtual host



WARNING

Deleting a virtual host removes all its resources including exchanges and queues permanently.

Step 6 Click **OK**.

This virtual host is deleted when it is removed from the **All virtual hosts** area on the **Virtual Hosts** page.

----End

Related Document

To delete a virtual host by calling an API, see **Deleting Specified Virtual Hosts in Batches**.

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 exchange.

Prerequisite

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

Notes and Constraints

Unbinding an exchange makes it unavailable permanently. Exercise caution.

Unbinding an Exchange from a Target Exchange

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange.

Step 7 In the row containing the desired exchange, click **Remove Binding**.

WARNING

Removing a binding makes it unavailable permanently. Exercise caution.

Step 8 Click **Yes**.

----End

Unbinding a Queue from a Source Exchange

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

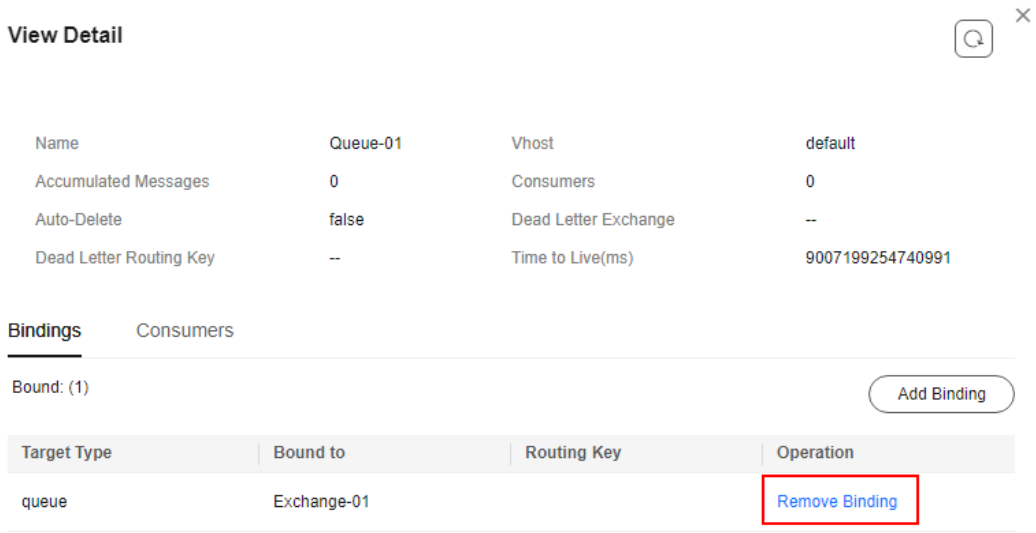
Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Queue** tab page, click **View Detail** in the row containing the desired queue.

Step 7 In the row containing the desired exchange, click **Remove Binding**.

Figure 4-11 Queue details



WARNING

Removing a binding makes it unavailable permanently. Exercise caution.

Step 8 Click **Yes**.

----End

Related Document

To unbind an exchange by calling an API, see **Deleting Bindings**.

4.7.2 Deleting RabbitMQ Exchanges

This section describes how to delete exchanges on the console.

Notes and Constraints

- Deleting an exchange removes all its configurations including exchange-exchange and exchange-queue bindings permanently.
- The default exchange cannot be deleted for RabbitMQ 3.x.x.

Prerequisite

An exchange has been created.

Deleting RabbitMQ Exchanges

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 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**.

WARNING

Deleting an exchange removes all its configurations including exchange-exchange and exchange-queue bindings permanently.

Step 7 Click **OK**.

This Exchange is deleted when it is removed from the exchange list.

----End

Related Document

To delete an exchange by calling an API, see **Deleting Specified Exchanges in Batches**.

4.8 Managing RabbitMQ Queues

4.8.1 Viewing a RabbitMQ Queue

After a queue is created, you can view the basic information, bindings, and consumers of it on the console.

Prerequisite

A queue has been created.

Viewing a RabbitMQ Queue

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Queue** tab page, click **View Detail** in the row containing the desired queue. The **basic information**, **bindings**, and **consumers** of the queue are displayed.

Figure 4-12 Queue details

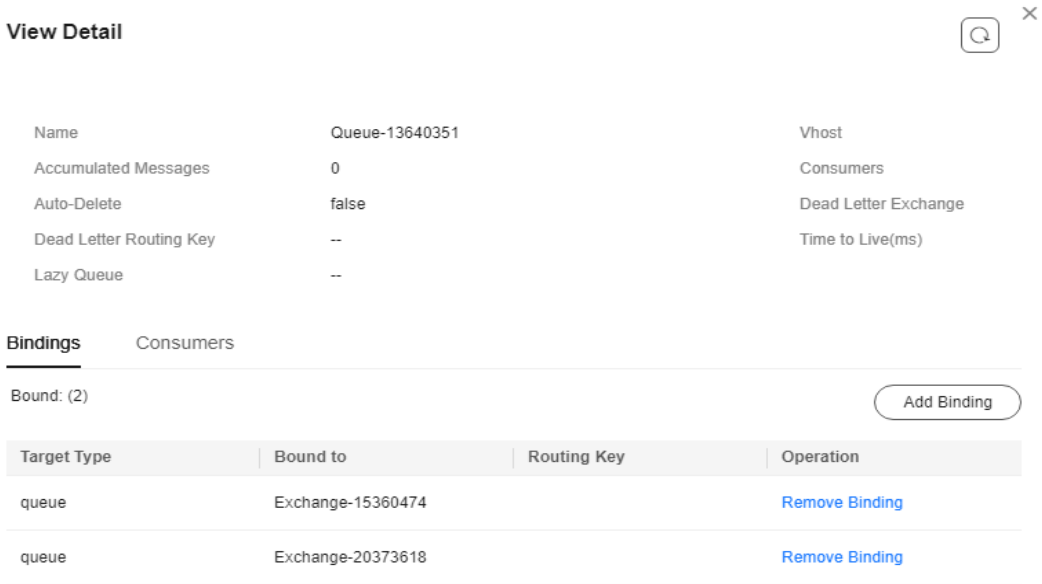


Table 4-8 Queue basic information parameters

Parameter	Description
Name	Name of a queue.
Vhost	Name of the virtual host to which the queue belongs.
Accumulated Messages	Number of accumulated messages in this queue.

Parameter	Description
Consumers	Number of consumers that consume messages in this queue.
Automatically delete	Whether this queue can be automatically deleted. • Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue. • No: This queue will not be deleted when the last consumer unsubscribes from the queue.
Dead Letter Exchange	When this queue is bound to a dead letter exchange, the name of that exchange is displayed. Otherwise, -- is displayed.
Dead Letter Routing Key	When this queue is bound to a dead letter exchange and a dead letter routing key is set, the key is displayed. Otherwise, -- is displayed.
Time to Live(ms)	When this queue is configured with message retention, this time is displayed. Otherwise, -- is displayed. 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	Available only for RabbitMQ 3.x.x. If this queue is a lazy queue, **lazy** is displayed. Otherwise, -- is displayed. For more information, see **Configuring Lazy Queues**.
Highest Priority	Available only for RabbitMQ AMQP. Priority of a queue. The larger the value, the higher the priority.

Table 4-9 Bindings parameters

Parameter	Description
Target Type	Binding type of an exchange. Only **queue** is displayed, indicating that the exchange is bound to a queue.
Bound to	Name of the exchange bound to this queue.
Routing Key	Key for routing messages to this queue.

Table 4-10 Consumers parameters

Parameter	Description
Consumer Tag	Unique identifier of a consumer client.

Parameter	Description
Channel	Channel through which a consumer client connects to the RabbitMQ instance.
Ack Required	Whether messages are automatically acknowledged. • **true**: Messages are marked as acknowledged and deleted from the queue immediately after being sent to a consumer. • **false**: Messages are marked as unacknowledged and retained in the queue until a consumer sends an acknowledgment (ack) to RabbitMQ. RabbitMQ then deletes the messages.
Prefetch Count	Prefetch value of messages. For more information, see **Configuring RabbitMQ Message Prefetch**.
User	Username used by a consumer client to connect to the RabbitMQ instance.

----End

Related Document

To view queue details by calling an API, see **Querying Specified Queue Details**.

4.8.2 Clearing Messages in a RabbitMQ Queue

This section describes how to clear all the messages in a queue.

Notes and Constraints

- **All the messages in the queue will be deleted permanently and cannot be restored.** Exercise caution.
- The two methods are supported for RabbitMQ 3.x.x. For RabbitMQ AMQP-0-9-1 instances, messages in a queue can only be cleared on the console.

Prerequisite

A queue has been created.

Procedure

Methods of deleting messages:

- **Clearing Messages in a Queue (Console)**
- **Clearing Messages in a Queue (RabbitMQ Management UI)**

Clearing Messages in a Queue (Console)

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Queue** tab page, click **Clear Message** in the row containing the desired queue. The **Clear Message** dialog box is displayed.

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

Step 7 Click **OK**.

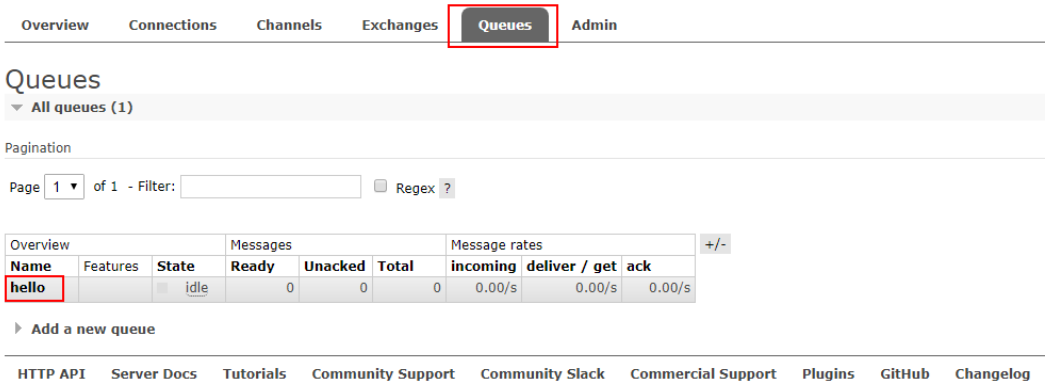
----End

Clearing Messages in a Queue (RabbitMQ Management UI)

Step 1 Log in to the **RabbitMQ management UI**.

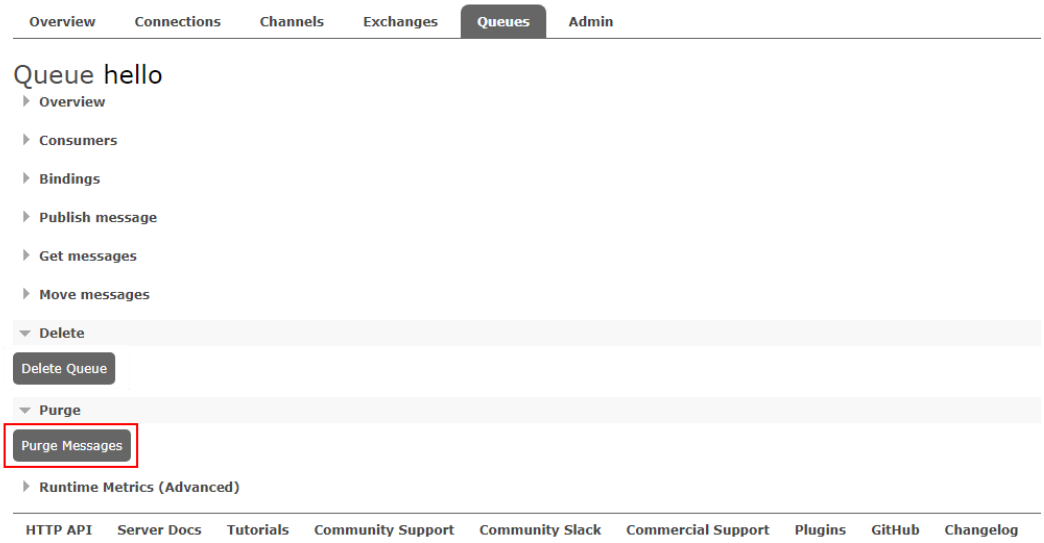
Step 2 On the **Queues** tab page, click the name of a queue.

Figure 4-13 Queues



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

Figure 4-14 Clearing messages in a queue



CAUTION

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

----End

Related Document

To clear messages in a queue by calling an API, see **Clearing Messages in a Queue**.

4.8.3 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.

Notes and Constraints

Unbinding a queue makes it unavailable permanently. Exercise caution.

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 **RabbitMQ console**.

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange.

Step 7 In the row containing the queue, click **Remove Binding**.

WARNING

Removing a binding makes it unavailable permanently. Exercise caution.

Step 8 Click **Yes**.

----End

Related Document

To unbind a queue by calling an API, see **Deleting Bindings**.

4.8.4 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.

Notes and Constraints

Queue mirroring is not supported for RabbitMQ AMQP-0-9-1 instances.

Prerequisite

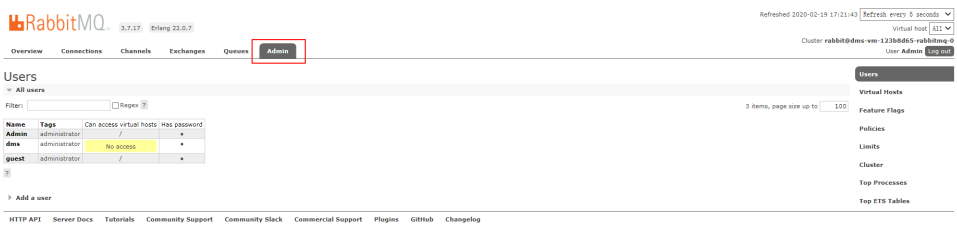
A cluster RabbitMQ instance has been created.

Configuring RabbitMQ Queue Mirroring

Step 1 Log in to the **RabbitMQ management UI**.

Step 2 Click the **Admin** tab.

Figure 4-15 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 4-16 Creating a virtual host

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

Figure 4-17 Setting virtual host policies

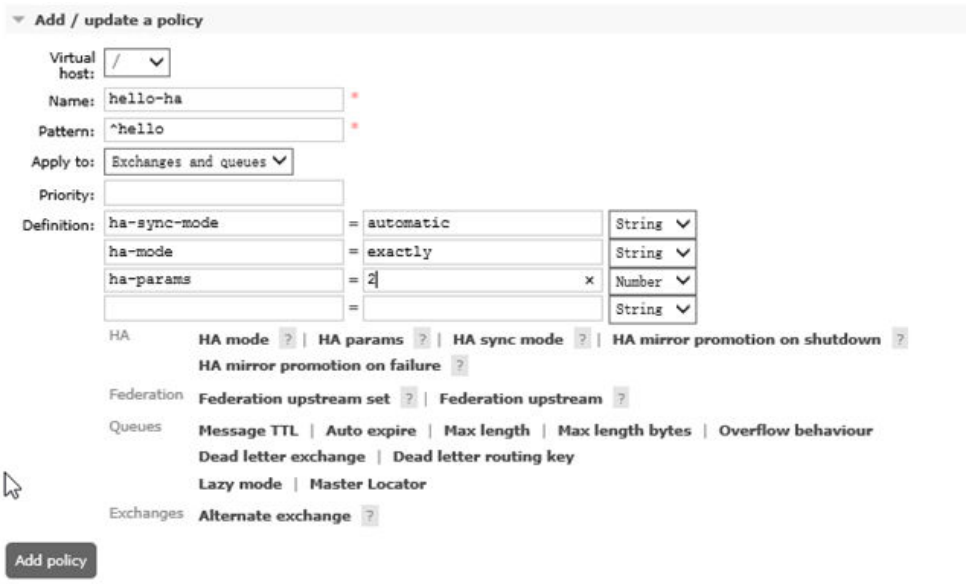


Table 4-11 Policy elements

Parameter	Description
Virtual Host	Select the virtual host to which the policy applies. The slash (/) indicates the default virtual host.
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	Definition of the mirror, which consists of **ha-sync-mode**, **ha-mode**, and **ha-params**. • **ha-sync-mode**: queue synchronization mode. Options: **automatic** and **manual**. – **automatic**: Data is automatically synchronized from the master. – **manual**: Data is manually synchronized from the master. • **ha-mode**: queue mirroring mode. Options: **all**, **exactly**, and **nodes**. – **all**: Mirror the queue across all nodes in the cluster. – **exactly**: Mirror the queue to a specific number (determined through **ha-params**) of nodes in the cluster. – **nodes**: Mirror the queue to specific nodes (determined through **ha-params**). • **ha-params**: This parameter is used in the **ha-mode** mode. **NOTE** Mirroring queues to all nodes in a cluster may waste network and disk I/O resources. You are advised to set parameters as follows: • **ha-sync-mode**: automatic • **ha-mode**: exactly • **ha-params**: *n*/2 + 1. *n* is the total number of nodes in a cluster. The value of *n*/2 is rounded down. For example, if the total number of nodes in a cluster is 3, set **ha-params** to **2** (3/2 = 1.5, 1.5 is rounded down to 1, 1 + 1 = 2). Queues will be mirrored to a master and a standby node. This configuration ensures high data availability, and avoids unnecessary resource overhead.

Step 5 Click **Add policy**.

The following figure shows a successfully added policy.

Figure 4-18 Virtual host policy

----End

4.8.5 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**.

Notes and Constraints

Available for RabbitMQ 3.8.35 and later.

Configuring a Lazy Queue

A queue has two modes: **default** and **lazy**. The default mode is **default**.

If these methods are used, the configuration set by the policy takes precedence.

- **Java client**: Set the queue in a parameter when calling method **channel.queueDeclare**.
- **RabbitMQ management UI**: Set the queue using a policy.
- **RabbitMQ console**: Set the queue when creating a queue.

Configuring a Lazy Queue on a Java Client

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

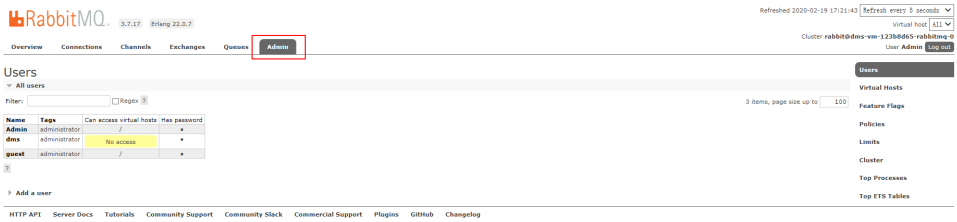
```
Map<String, Object> args = new HashMap<String, Object>();
args.put("x-queue-mode", "lazy");
channel.queueDeclare("myqueue", false, false, false, args);
```


Configuring a Lazy Queue on the RabbitMQ Management UI

Step 1 Log in to the **RabbitMQ management UI**.

Step 2 Click the **Admin** tab.

Figure 4-19 Admin tab page



Step 3 Choose **Policies** from the navigation pane.

Step 4 Add a policy.

Figure 4-20 Adding a policy

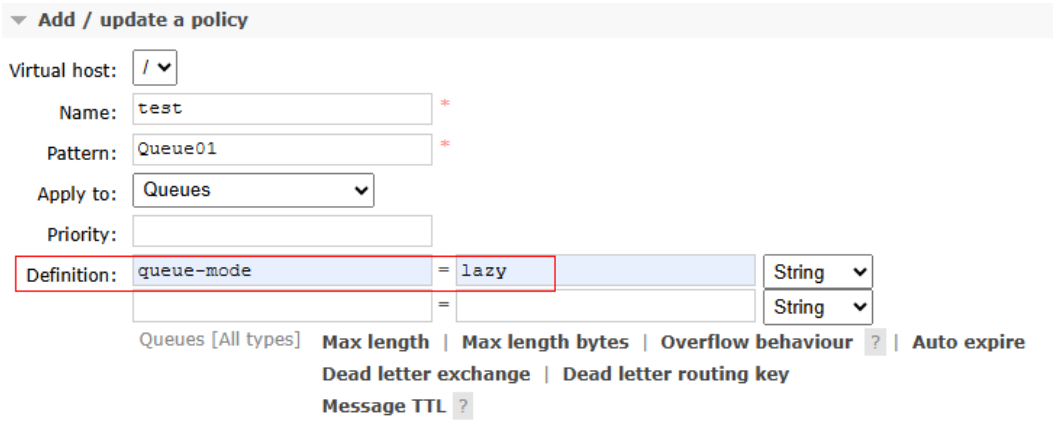


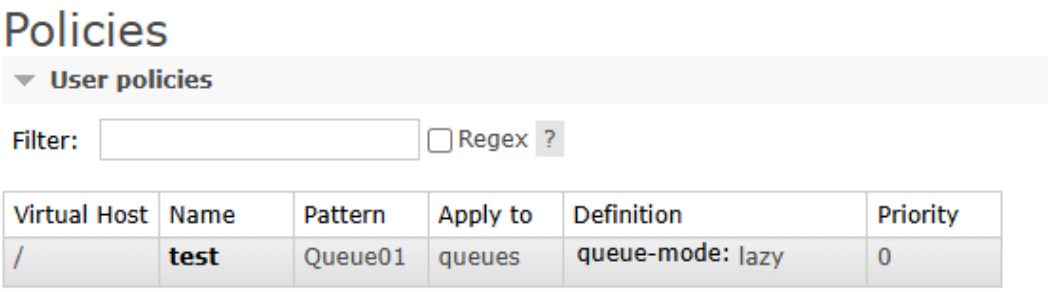
Table 4-12 Policy elements

Parameter	Description
Virtual Host	Specify the virtual host. The slash (/) indicates the default virtual host.
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. Select **Queues**.
Priority	A larger value indicates a higher priority.
Definition	Configure lazy queues by setting **queue-mode=lazy**.

Step 5 Click **Add/update policy**.

The following figure shows a successfully added policy.

Figure 4-21 Viewing a lazy queue policy

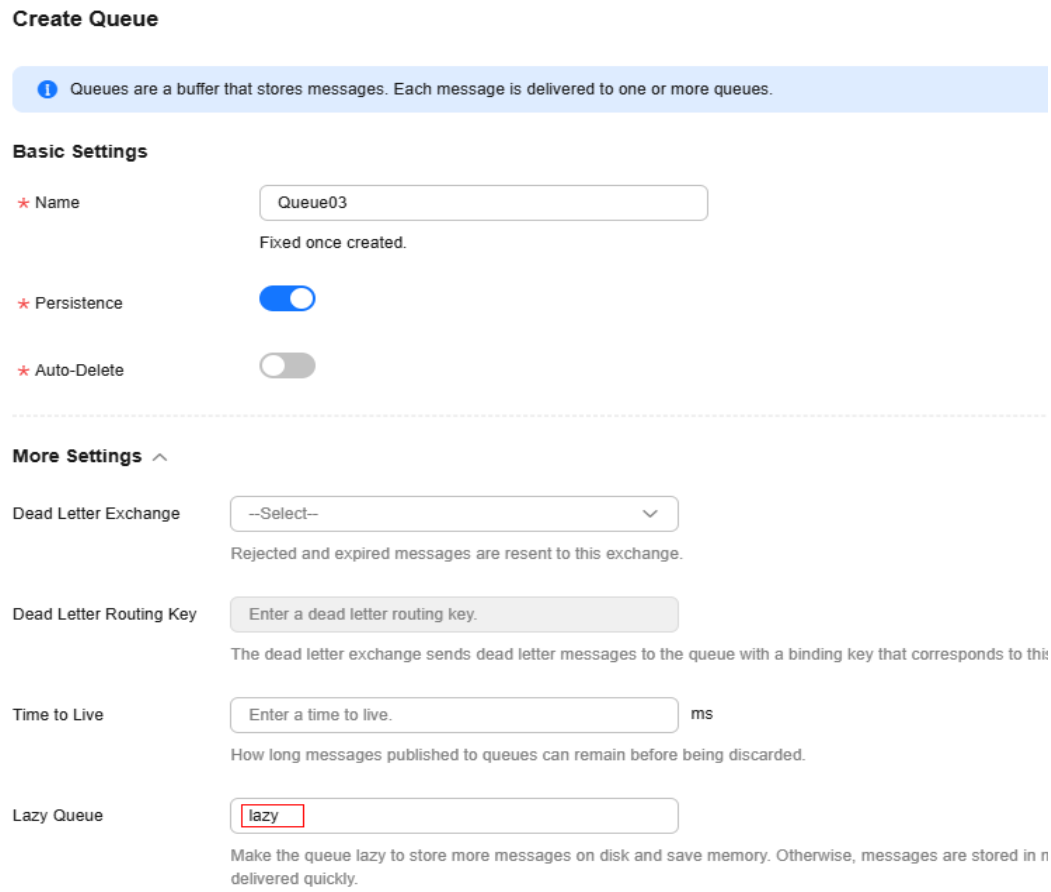


----End

Configuring a Lazy Queue on the RabbitMQ Console

Step 1 Create a queue by referring to **Creating a RabbitMQ Queue** and set **Lazy Queue** to **lazy**.

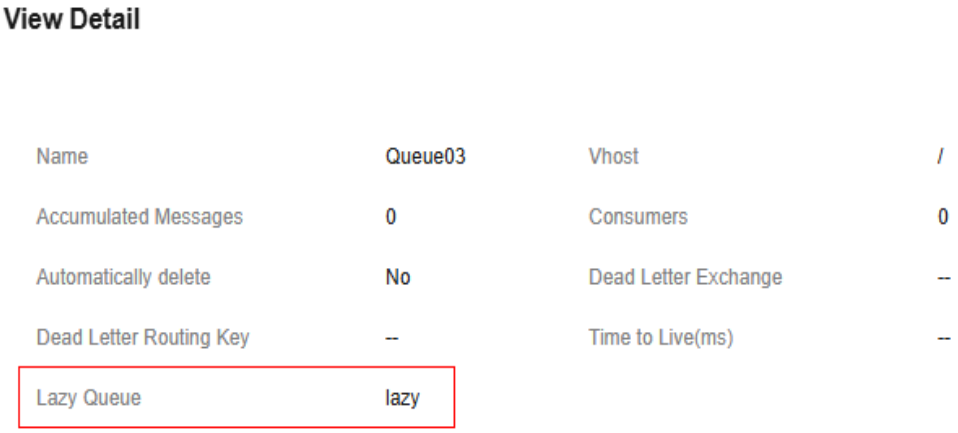
Figure 4-22 Creating a lazy queue



Step 2 In the queue list, click **View Detail** next to the created queue.

The lazy queue is created when **Lazy Queue** is **lazy**.

Figure 4-23 Viewing queue details



----End

Related Document

To create a lazy queue by calling an API, see **Creating a Queue**.

4.8.6 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**.

Notes and Constraints

Available for RabbitMQ 3.8.35 and later.

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 4-13**. Quorum queues consume more memory and disk space.

Table 4-13 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 **x-overflow: reject-publish-dlx**)

Feature	Mirrored Queues	Quorum Queues
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

A quorum queue can be configured when a queue is declared on a client or on the RabbitMQ management UI.

Configuring a Quorum Queue on a Java Client

When declaring a queue, set the **x-queue-type** queue argument to **quorum**. 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);
```

Configuring a Quorum Queue on the RabbitMQ Management UI

A quorum queue can only be set in queue creation but not by a policy.

Step 1 Log in to the **RabbitMQ management UI**.

Step 2 Click the **Queues** tab.

Step 3 Create a quorum queue.

Figure 4-24 Creating a quorum queue

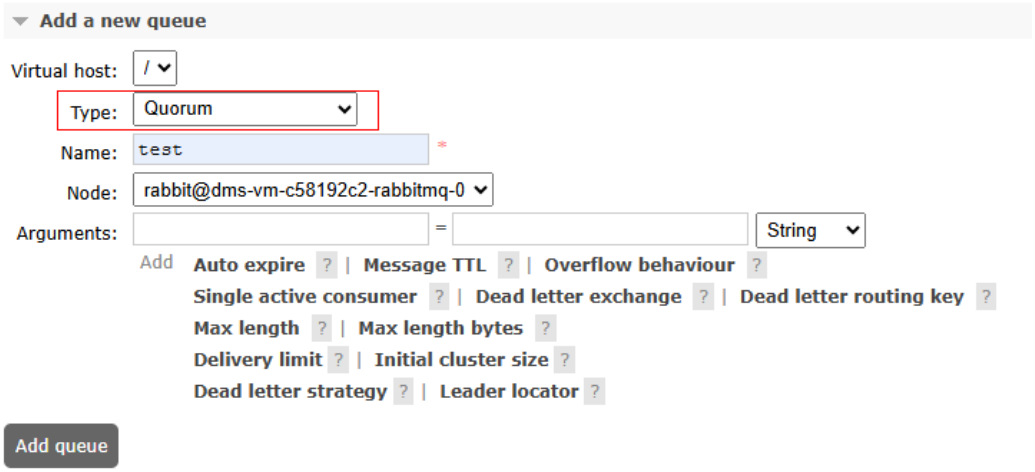


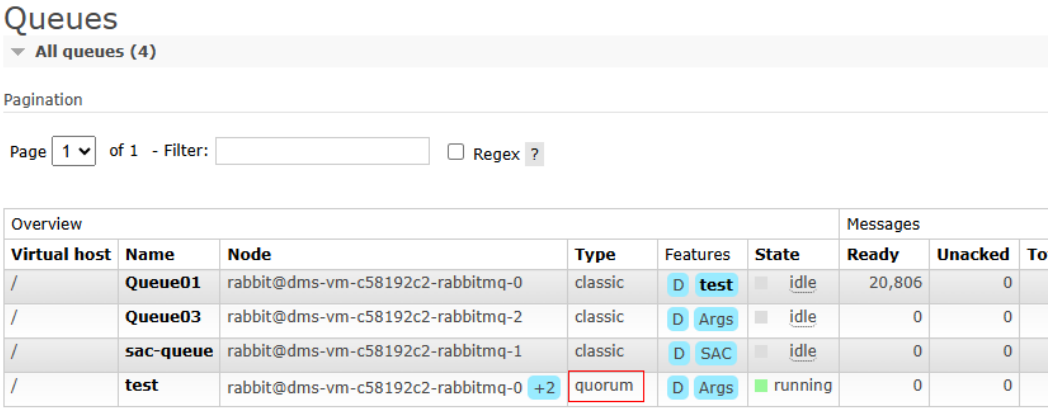
Table 4-14 Queue parameters

Parameter	Description
Virtual Host	Virtual host to which a quorum queue belongs. The slash (/) indicates the default virtual host.
Type	Queue type. Select **Quorum**.
Name	Quorum queue name, which is user-defined.
Node	Node where a quorum queue is deployed.
(Optional) Arguments	Extended attribute. You do not need to set it.

Step 4 Click **Add queue**.

Step 5 On the **Queues** page, check that **Type** is **quorum**, as shown in **Figure 4-25**. The quorum queue is created.

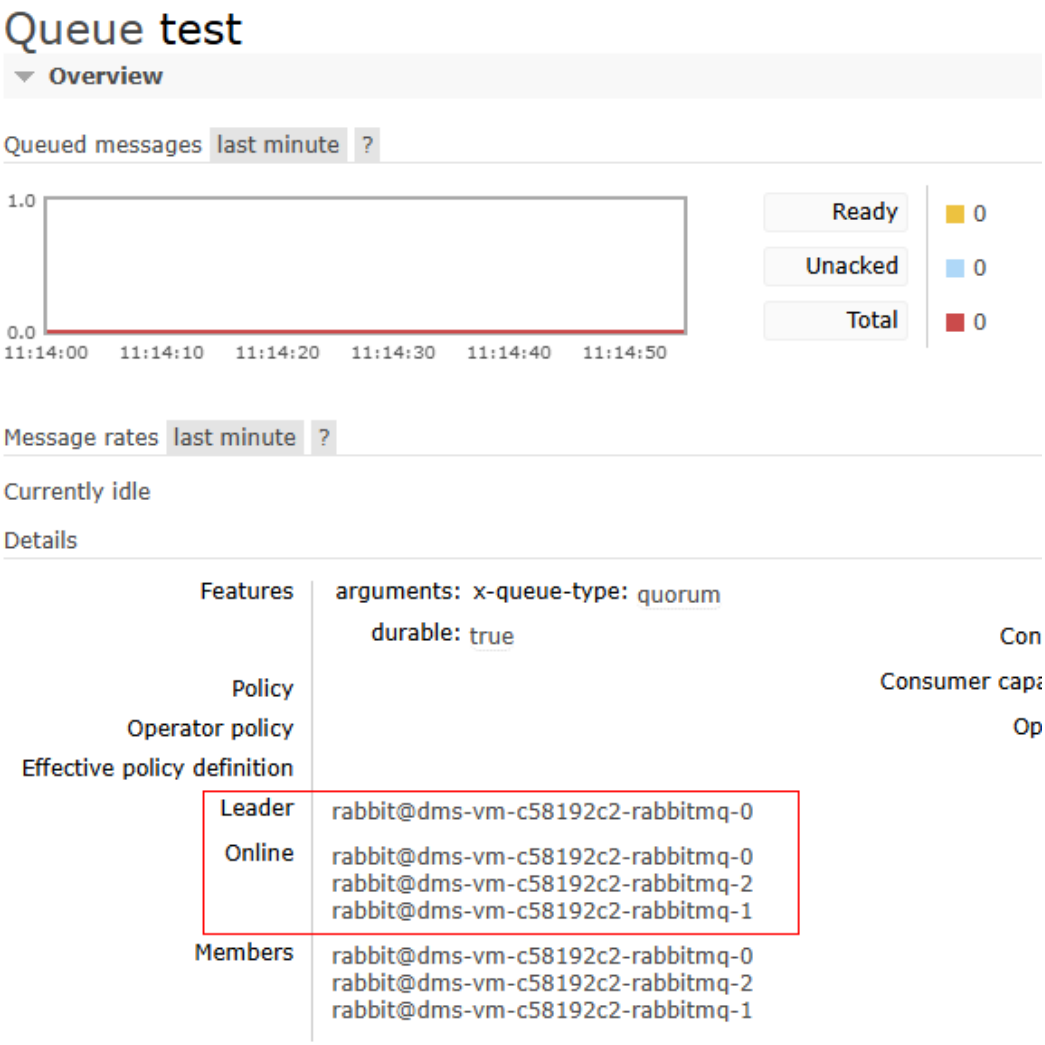
Figure 4-25 Checking the queue type



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.

Step 6 (Optional) 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 4-26 Queue details page



----End

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.

Table 4-15 Configuring the Quorum Queue Length

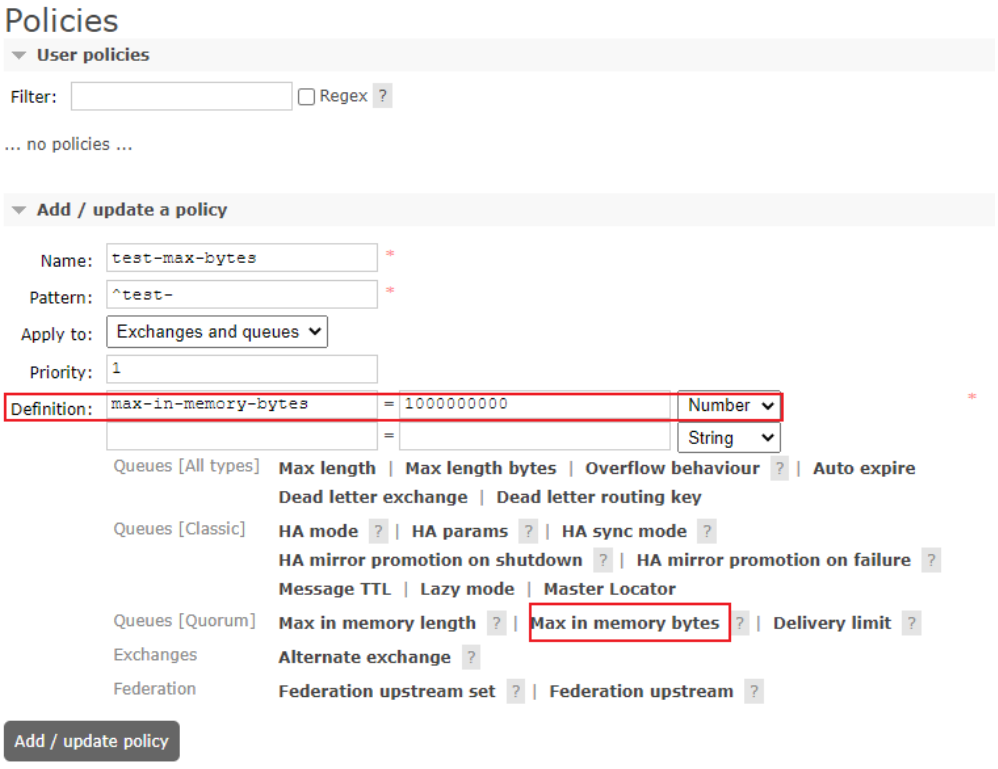
Parameter	Description
x-max-length	Maximum number of messages in the quorum queue. If this limit is exceeded, excess messages will be discarded or sent to the dead letter exchange.
x-max-length-bytes	Maximum size (in bytes) of messages in the quorum queue. If this limit is exceeded, excess messages will 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.

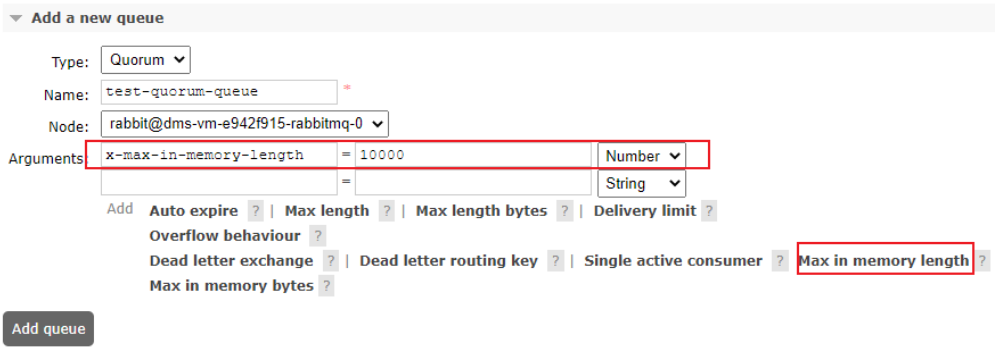
  Figure 4-27 Using a policy to set x-max-in-memory-bytes



- By setting the **queue parameter**

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

Figure 4-28 Setting x-max-in-memory-length in the queue argument



4.8.7 Configuring a RabbitMQ Exclusive Queue

An exclusive queue is visible and accessible only to the connection that declares it for the first time. The connection can be used to produce and consume messages, and clear and delete the queue. Other connections can neither access the queue, nor declare another exclusive queue with its name.

An exclusive queue cannot be persistent. Once the connection is closed, the queue and its data are automatically deleted.

Generally, exclusive queues temporarily store messages and the connection is shared by a producer and a consumer.

Notes and Constraints

- Even if an exclusive queue is declared persistent, it is deleted upon the disconnection.
- Exercise caution and note that queue data may be lost.

Example Configuration

On a **Java client**:

```
boolean durable = false;  //Whether to enable persistence.
boolean exclusive = true;  //Whether to enable exclusion.
boolean autoDelete = true;  //Whether to enable automatic queue deletion.
Map<String, Object> arguments = new HashMap<>();  //Other parameters
channel.queueDeclare(EXCLUSIVE_QUEUE_NAME, durable, exclusive, autoDelete, arguments);
```

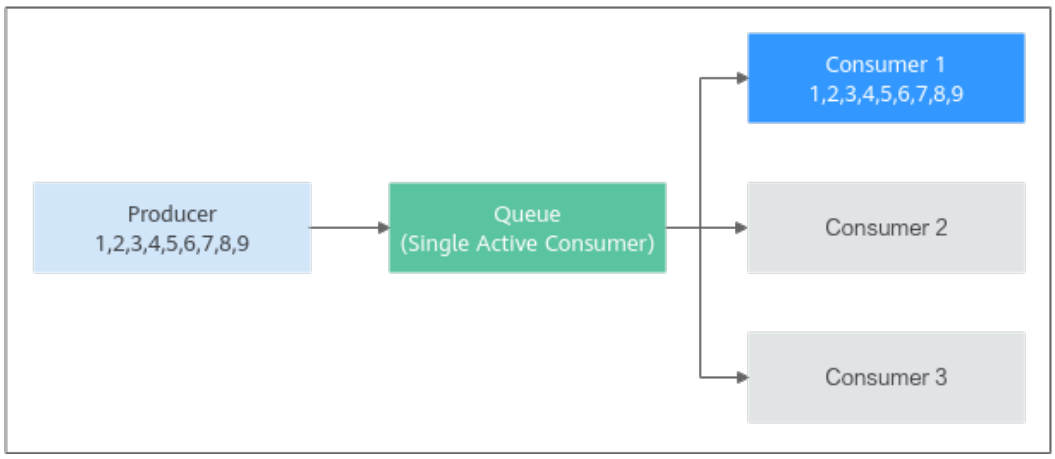
4.8.8 Configuring a Single Active Consumer

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.

In Figure 4-29, 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 4-29 Single active consumer



Notes and Constraints

Available for RabbitMQ 3.8.35 and later.

Procedure

A single active consumer can be configured when a queue is declared on a client or on the RabbitMQ management UI.

Configuring a Single Active Consumer on a Java Client

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);
```

Configuring a Single Active Consumer on the RabbitMQ Management UI

Step 1 Log in to the **RabbitMQ management UI**.

Step 2 Click the **Queues** tab.

Step 3 Create a single active consumer queue.

Figure 4-30 Creating a single active consumer queue

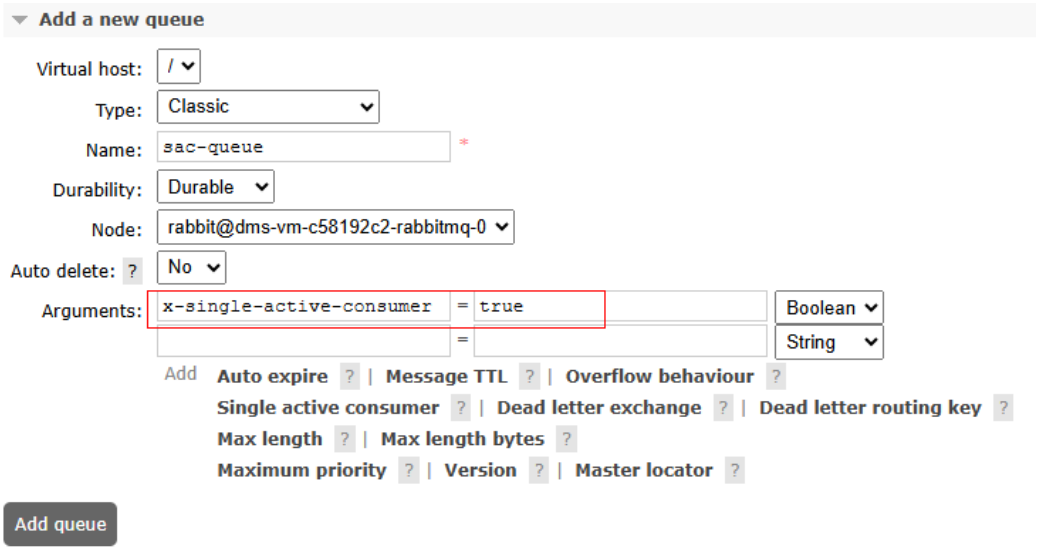


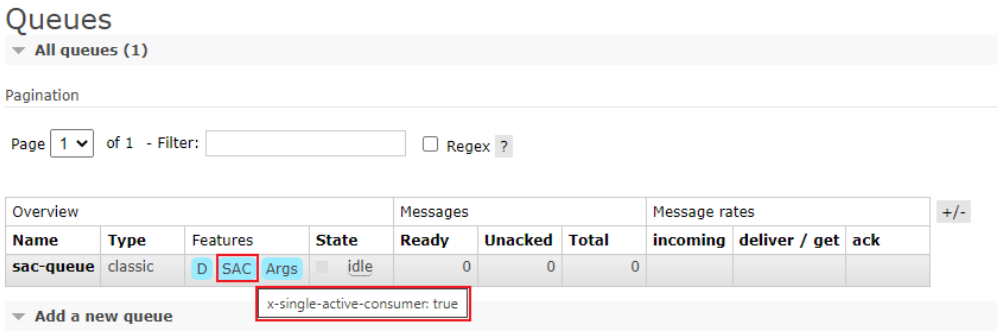
Table 4-16 Queue parameters

Parameter	Description
Virtual Host	Virtual host to which a single active consumer queue belongs. The slash (/) indicates the default virtual host.
Type	Queue type.
Name	Name of the single active consumer queue, which is user-defined.
Durability	Whether to enable persistence. • Durable: This queue survives after server restart. • Transient: This queue will be deleted after server restart and needs to be recreated.
Node	Node where a single active consumer queue is deployed.
Auto delete	Whether to enable automatic deletion. • Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue. • No: This queue will not be deleted when the last consumer unsubscribes from the queue.
Arguments	Set the single active consumer attribute through parameter **x-single-active-consumer=true**.

Step 4 Click **Add queue**.

Step 5 Check whether the queue features contain single active consumer on the **Queues** page. As shown in **Figure 4-31**, **SAC** indicates that a single active consumer has been set in the queue.

Figure 4-31 Viewing queue features



----End

4.8.9 Deleting RabbitMQ Queues

This section describes how to delete queues. Methods of deleting a queue:

- **Deleting a Queue (Console)**
- **Deleting a Queue (RabbitMQ Management UI)**
- **Deleting Queues in Batches (RabbitMQ Management UI)**

Notes and Constraints

- Deleting a queue removes all its configurations including exchange-queue bindings permanently.
- The three methods are supported for RabbitMQ 3.x.x. For RabbitMQ AMQP-0-9-1 instances, queues can only be deleted on the console.

Prerequisite

A queue has been created.

Procedure

The following describes how to delete a queue.

Deleting a Queue (Console)

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Instance** > **Virtual Hosts**.

Step 5 Click a virtual host name.

Step 6 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**.

> **WARNING**
>
> Deleting a queue removes all its configurations including exchange-queue bindings permanently.

Step 7 Click **OK**.

This queue is deleted when it is removed from the queue list.

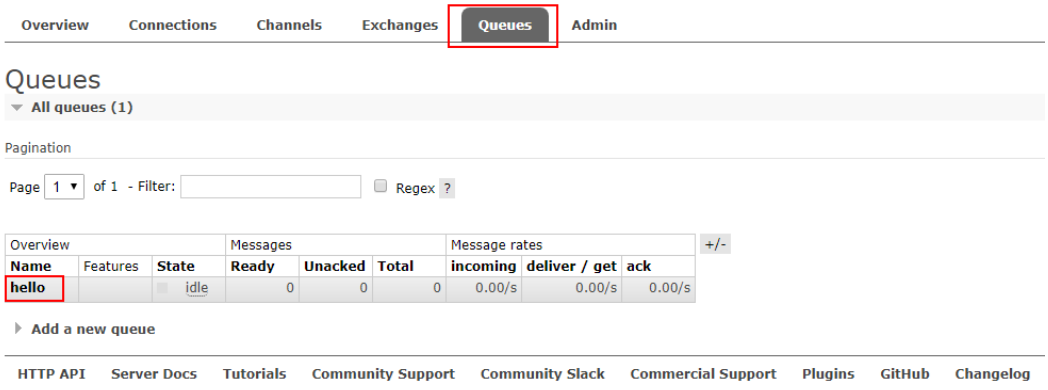
----End

Deleting a Queue (RabbitMQ Management UI)

Step 1 **Log in to the RabbitMQ management UI**.

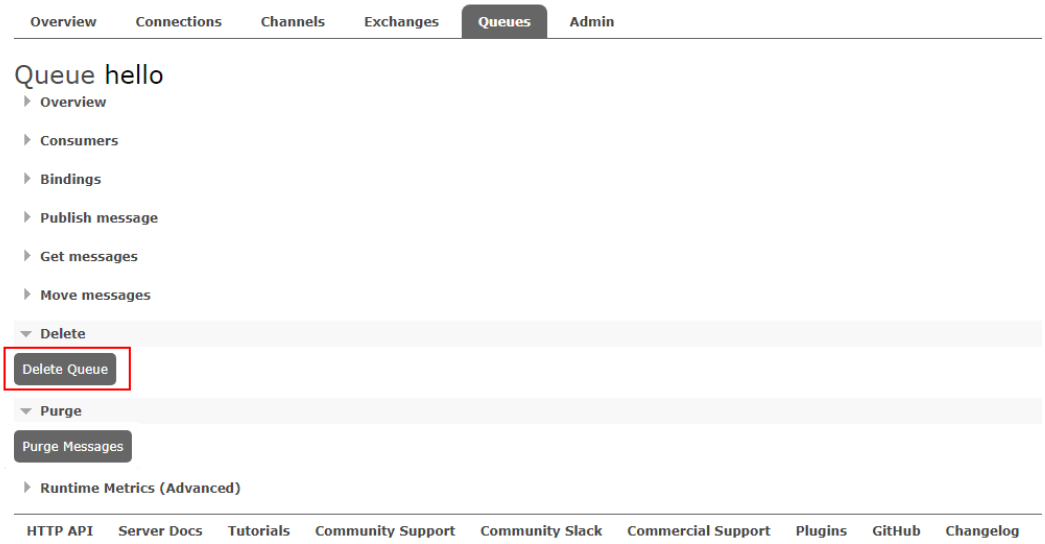
Step 2 On the **Queues** tab page, click the name of the desired queue.

Figure 4-32 Queues



Step 3 Click **Delete Queue**.

Figure 4-33 Deleting a queue



WARNING

Deleting a queue removes all its configurations including exchange-queue bindings permanently.

In the queue list in the **Overview** area, check that the deleted queue is removed.

----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 4-34 Adding a policy to delete queues in batches

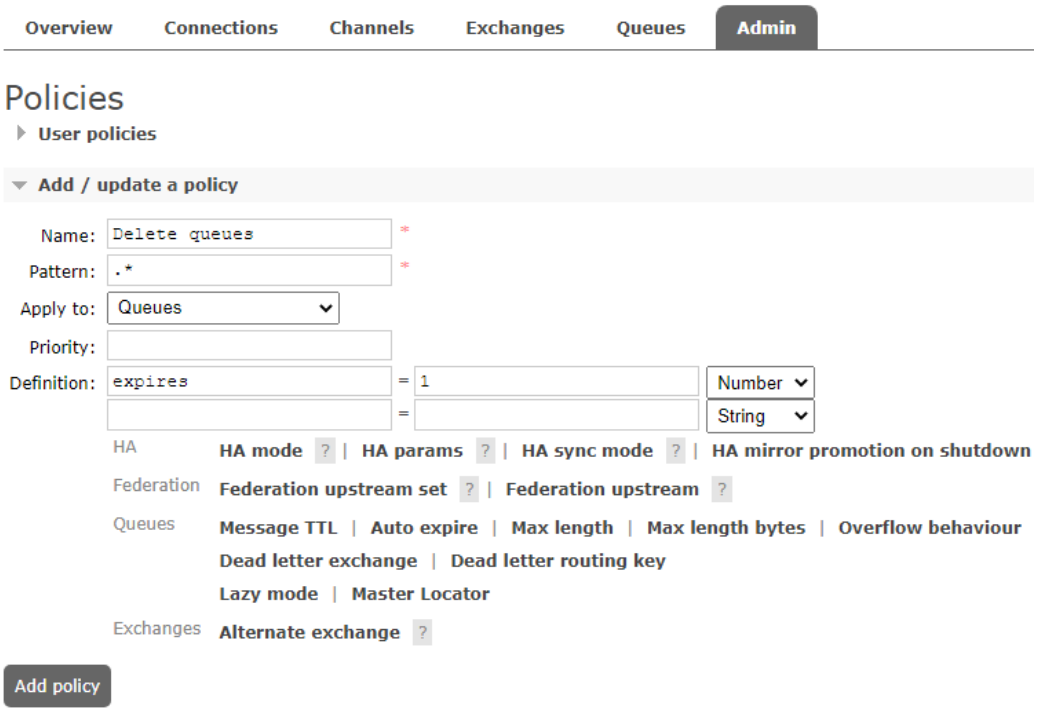


Table 4-17 Policy elements

Parameter	Description
Name	The policy name, which can be customized.

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

WARNING

Deleting a queue removes all its configurations including exchange-queue bindings permanently.

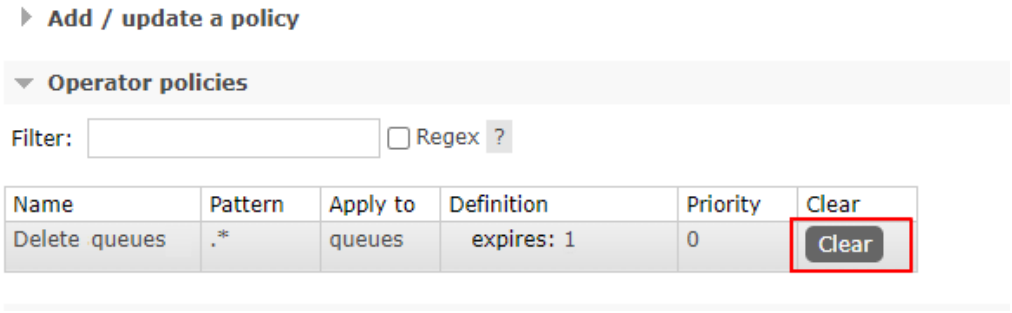
Step 3 Click **Add policy**.

In the queue list in the **Overview** area on the **Queues** page, check that the deleted queue is removed.

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 4-35 Deleting the policy



----End

Related Document

To delete a queue by calling an API, see **Deleting Specified Queues in Batches**.

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.	**Configuring RabbitMQ Public Access**
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.	**VPC Peering Connection**

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** or **Table 5-3**.

Table 5-2 Security group rules (RabbitMQ 3.x.x)

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)

Table 5-3 Security group rules (RabbitMQ AMQP-0-9-1)

Direction	Type	Protocol	Port	Source	Description
Inbound	IPv4	TCP	5672	IP address or IP address group of the RabbitMQ client	Accessing a RabbitMQ instance

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.

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.

Notes and Constraints

The RabbitMQ console only supports IPv4 EIPs. IPv6 EIPs are not supported.

Prerequisite

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

Procedure

The following describes how to enable or disable public access.

Enabling Public IPv4 Access (RabbitMQ 3.x.x)

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click the desired instance to view its details.

Step 4 Click  next to **Public Access**.

Step 5 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 10–30s 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.

Step 6 Set the security group rules listed in **Table 5-4** for the RabbitMQ instance so that you can access RabbitMQ using the IPv4 EIP.

Table 5-4 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)

----End

Disabling Public IPv4 Access (RabbitMQ 3.x.x)

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click the desired instance to go to the instance details page.

Step 4 Click next to **Public Access**.

Step 5 Click .

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

----End

5.2 Configuring RabbitMQ Access Control

5.2.1 Enabling RabbitMQ ACL

RabbitMQ instances support ACL-based permission isolation among producers and consumers. You can create multiple users, and grant virtual host (resource) permissions to them. When ACL is enabled, message production and consumption require authentication.

Notes and Constraints

- Enabling ACL will disconnect clients without authentication configuration.
- Available on the console only for RabbitMQ AMQP-0-9-1. By default, you can create users on the management UI, and assign permissions for RabbitMQ 3.x.x instances.

Prerequisite

A RabbitMQ AMQP-0-9-1 instance has been purchased.

Enabling ACL

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the **Connection** area, click next to **ACL** to enable ACL.

WARNING

Enabling ACL will disconnect clients without authentication configuration.

Step 5 Click **OK**.

ACL is enabled when **ACL** is **Enabled** in the **Connection** area.

----End

5.2.2 Configuring RabbitMQ ACL Users

When ACL is enabled for a RabbitMQ instance, message production and consumption require authentication.

This section describes how to create, edit, and delete a user.

Notes and Constraints

- Deleting a user will remove its authorization relationship and disconnect it from the instance.
- Available on the console only for RabbitMQ AMQP-0-9-1. By default, you can create users on the management UI, and assign permissions for RabbitMQ 3.x.x instances.

Prerequisites

- A RabbitMQ AMQP-0-9-1 instance has been purchased.
- **ACL** has been enabled.

Creating a User

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Users**.

Step 5 Click **Create User**.

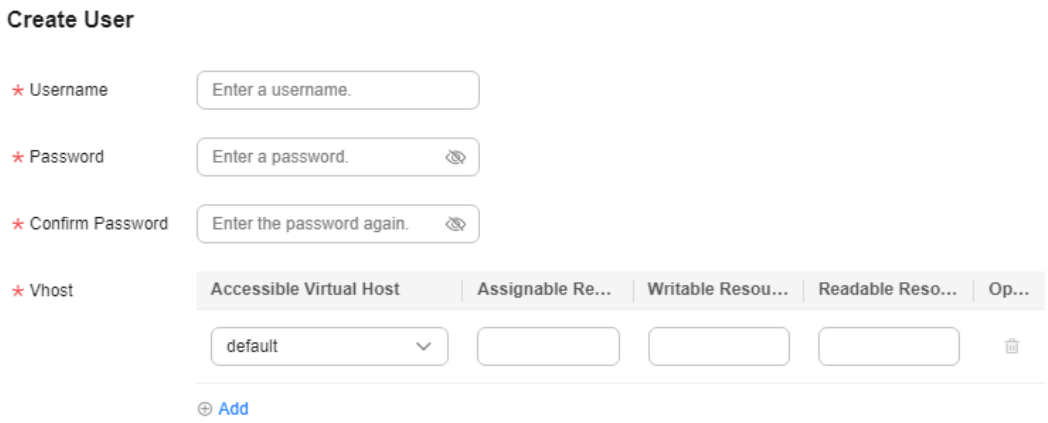
Step 6 Configure the user's name and other parameters by referring to **Table 5-5**.

Table 5-5 User parameters

Parameter	Description
Username	You can customize a username that complies with the rules: 7–64 characters; starts with a letter; can contain only letters, digits, hyphens (-), and underscores (_). The name cannot be changed after the user is created.

Parameter	Description
Password	Password of the user. 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.
Confirm Password	Enter the password again.
Vhost	• **Accessible Virtual Host**: Select a virtual host from the drop-down list box. • **Assignable Resource**: Use regular expressions to grant user permissions for virtual host resources. For example, **^test-.*** grants the user permissions for all the resources whose names start with **test-**. • **Writable Resource**: Use regular expressions to grant user write permissions for virtual host resources. For example, **.***, grants user write permissions for all virtual host resources. • **Readable Resource**: Use regular expressions to grant user read permissions for virtual host resources. For example, **.*** grants user read permissions for all virtual host resources. Click **Add** to add virtual hosts as required.

Figure 5-1 Creating a User



Step 7 Click **OK**.

View the new user on the user list page.

Step 8 After ACL is enabled, user authentication information (username and password) must be added to both the producer and consumer configurations. For details, see the following instructions:

- **Example Java Code**
- **Example Java Code**

----End

Modifying User Information

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Users**.

Step 5 In the row containing the desired user, click **Edit**.

Step 6 To edit the password, click **Edit** next to **Password** and enter a new password. To edit the virtual host:

Table 5-6 Parameters for editing a virtual host

Parameter	How to Edit
Accessible Virtual Host	Select a virtual host from the drop-down list box.
Assignable Resource	Use regular expressions to grant user permissions for virtual host resources. For example, **^test-.*** grants the user permissions for all the resources whose names start with **test-**.
Writable Resource	Use regular expressions to grant user write permissions for virtual host resources. For example, **.***, grants user write permissions for all virtual host resources.
Readable Resource	Use regular expressions to grant user read permissions for virtual host resources. For example, **.*** grants user read permissions for all virtual host resources.

Step 7 Click **OK**.

----End

Deleting a User

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

Step 2 Click  in the upper left corner to select the region where your instance is located.

Step 3 Click an instance name to go to the instance details page.

Step 4 In the navigation pane, choose **Users**.

Step 5 In the row containing the user to be deleted, click **Delete**.

> ⚠ WARNING
>
> Deleting a user will remove its authorization relationship and disconnect it from the instance.

Step 6 Click **OK**.

The user is deleted when it is removed from the user list.

----End

Related Documents

- To create an ACL user by calling an API, see **Creating a User**.
- To delete an ACL user by calling an API, see **Deleting Users**.

5.3 Configuring Heartbeats on the RabbitMQ Client

If messages may be consumed more than 90s after they are produced, enable heartbeats on the client and set the heartbeats to shorter than 90s, to prevent the client from being disconnected from a cluster RabbitMQ instance. Connections may be reestablished shortly after network jitters when heartbeats are low. Performing a master/standby switchover may take longer when heartbeats are high. **The recommended heartbeat setting is 10s.**

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**.

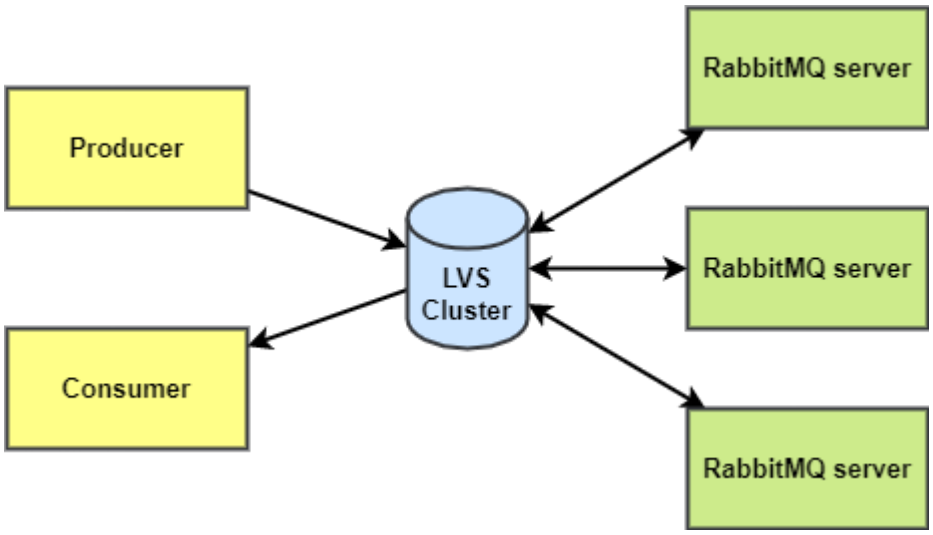
CAUTION

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 5-2**. Single-node instances do not have LVSs.

Figure 5-2 Load balancing of a cluster instance



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

If messages are consumed more than 90s after they are produced, enable heartbeats on the client and set the heartbeat timeout to shorter than 90s. The recommended heartbeat setting is 10s.

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 10s.
cf.setRequestedHeartbeat(10);
```


- **.NET client**

```
var cf = new ConnectionFactory();
// The heartbeat timeout is 10s.
cf.RequestedHeartbeat = TimeSpan.FromSeconds(10);
```

- **Python Pika**

```
// The heartbeat is 10s.
params = pika.ConnectionParameters(host='host', heartbeat=10,
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 10s.
$connection = new AMQPStreamConnection(RMQ_HOST, RMQ_PORT, RMQ_USER, RMQ_PASS,
RMQ_vhost, ['heartbeat'=> 10]);
```

- **Spring-amqp**

```
// The heartbeat is 10s.
@Bean
CachingConnectionFactory connectionFactory(ConnectionFactory rabbitConnectionFactory) {
    CachingConnectionFactory ccf = new CachingConnectionFactory(rabbitConnectionFactory);
    ccf.setHost("...");
    ccf.setRequestedHeartBeat(10);
    // ...
    return ccf;
}
```

5.4 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

The following uses Linux as an example.

Step 1 Log in to the client server.

Step 2 Download **RabbitMQ-Tutorial.zip** sample project code.

```
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}
```

Table 5-7 Message production parameters

Parameter	Description
host	Connection address of the RabbitMQ instance, which is obtained from **Prerequisites**.
port	Port of the RabbitMQ instance. Enter **5672**.
user	Username for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.
password	Password for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.

Sample message production:

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

Step 6 Retrieve messages using the sample project.

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

Table 5-8 Message consumption parameters

Parameter	Description
host	Connection address of the RabbitMQ instance, which is obtained from **Prerequisites**.
port	Port of the RabbitMQ instance. Enter **5672**.

Parameter	Description
user	Username for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.
password	Password for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.

Sample message consumption:

```
[root@ecs-test RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Recv 192.168.xx.40 5672 test Zxxxxxs
 [*] 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 are as follows. **Table 5-9** describes the parameters to be modified.

```
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(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();
```

Table 5-9 Parameters

Parameter	Description
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

- Accessing an instance and consuming messages are as follows. **Table 5-10** describes the parameters to be modified.

```
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(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);
```

Table 5-10 Parameters

Parameter	Description
VHOST_NAME	Name of the virtual host that contains the queue to consume messages
QUEUE_NAME	Name of the queue to consume messages

Related Documents

- **Why Does a Client Fail to Connect to a RabbitMQ Instance?**
- **Why Does a RabbitMQ Cluster Have Only One Connection Address?**

5.5 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

The following uses Linux as an example.

Step 1 Log in to the client server.

Step 2 Download **RabbitMQ-Tutorial.zip** sample project code.

```
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
```

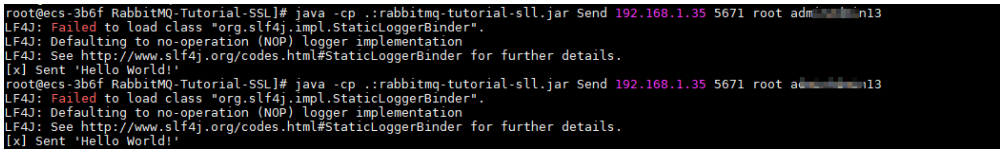
Step 5 Produce messages using the sample project.

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

Table 5-11 Message production parameters

Parameter	Description
host	Connection address of the RabbitMQ instance, which is obtained from **Prerequisites**.
port	Port of the RabbitMQ instance. Enter **5671**.
user	Username for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.
password	Password for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.

Figure 5-3 Sample project for message creation



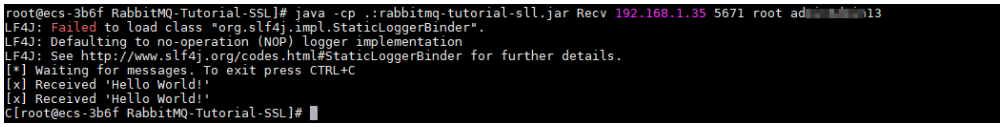
Step 6 Consume messages using the sample project.

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


Table 5-12 Message consumption parameters

Parameter	Description
host	Connection address of the RabbitMQ instance, which is obtained from **Prerequisites**.
port	Port of the RabbitMQ instance. Enter **5671**.
user	Username for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.
password	Password for connecting to the RabbitMQ instance, which is obtained from **Prerequisites**.

Figure 5-4 Sample project for message retrieval



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

----End

Java Sample Code

- Accessing an instance and producing messages are as follows. **Table 5-13** describes the parameters to be modified.

```
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();
```

Table 5-13 Parameters

Parameter	Description
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

Parameter	Description
Hello World!	The message to be sent in this sample

- Accessing an instance and consuming messages are as follows. **Table 5-14** describes the parameters to be modified.

```
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);
```

Table 5-14 Parameters

Parameter	Description
VHOST_NAME	Name of the virtual host that contains the queue to consume messages
QUEUE_NAME	Name of the queue to consume messages

Related Documents

- **Why Does a Client Fail to Connect to a RabbitMQ Instance?**
- **Why Does a RabbitMQ Cluster Have Only One Connection Address?**

6 Managing Messages

6.1 Viewing RabbitMQ Messages

Producers send messages to queues. You can view the message content and traces in a specific virtual host and queue on the console.

Notes and Constraints

Available only for RabbitMQ AMQP-0-9-1.

Viewing RabbitMQ Messages

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

Step 2 Click in the upper left corner to select the region where your instance is located.

Step 3 Click a RabbitMQ instance name to go to the instance details page.

Step 4 In the left navigation pane, choose **Message Query**.

Step 5 Set the query parameters by referring to **Table 6-1**.

Table 6-1 Message query parameters

Parameter	Description
Vhost	Select the virtual host that has the messages to be queried.
Queue	Select the queue that has the messages to be queried.
Stored	Specify the time when messages were stored in the queue.

Step 6 Click **Search**.

The query result is as follows.

Figure 6-1 Searching for messages

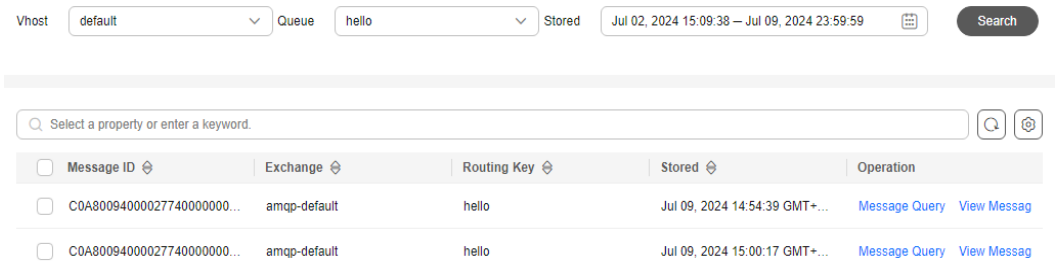


Table 6-2 lists the message parameters.

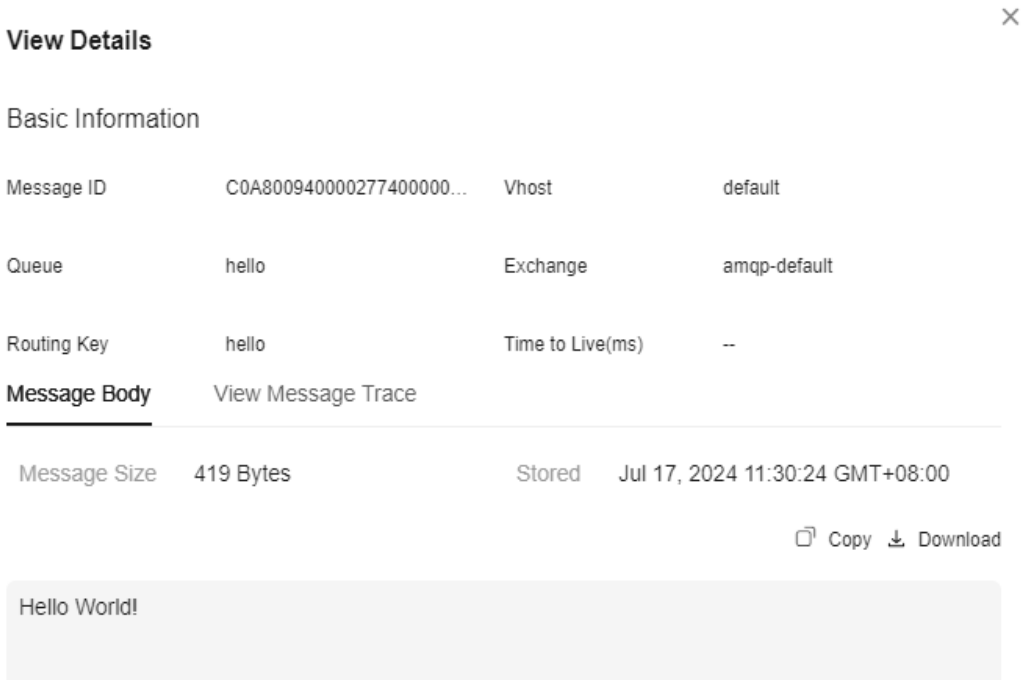
Table 6-2 Message parameters

Parameter	Description
Message ID	Message identifier.
Exchange	Exchange to which a message belongs.
Routing Key	Keyword for routing messages from an exchange to a queue.
Stored	Time when messages were stored in the queue.

Step 7 Click **Message Query** in the row containing the desired message. The **View Details** page is displayed. View the message size, storage time, and content on the **Message Body** tab page.

The console displays messages smaller than 4 KB. To view messages larger than 4 KB, click **Download Message**.

Figure 6-2 Message body



Step 8 Click **View Message Trace** in the row containing the desired message. The **View Details** page is displayed. Check whether messages are consumed on the **View Message Trace** tab page.

Figure 6-3 Viewing message trace

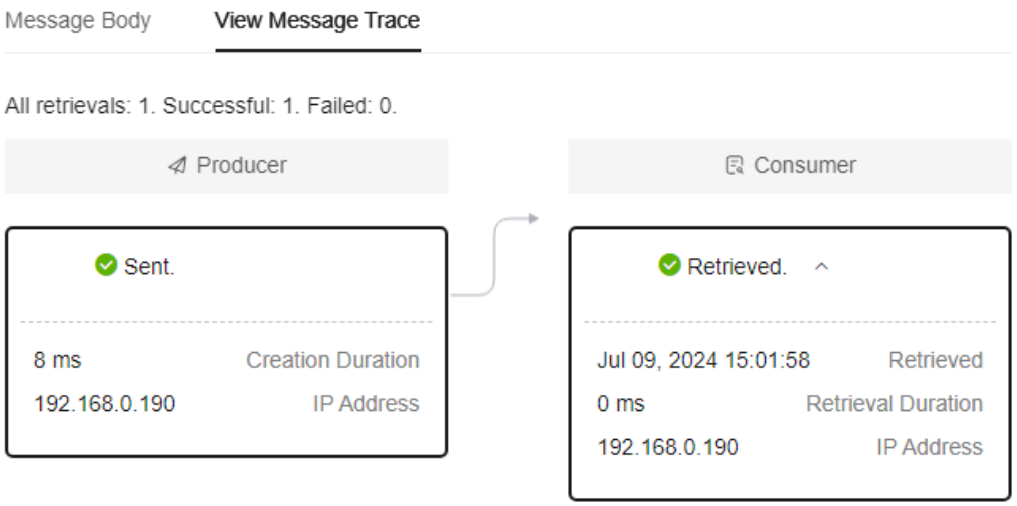


Table 6-3 describes message trace parameters.

Table 6-3 Message trace parameters

Parameter	Description
Producer status	A producer can be in the following state: Sent: The message is sent successfully, and the server has successfully stored the message.
Creation duration	Time taken to send the message by the producer, in milliseconds.
Address	IP address of the producer.
Consumer status	A consumer can be in one of the following states: • Retrieved • Retrieval timed out • Abnormal retrieval • NULL returned • Retrieval failed
Retrieved	Time when the message is retrieved.
Retrieval duration	Time taken to consume the message by the consumer, in milliseconds.
Address	IP address of the consumer.

----End

6.2 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**.

RabbitMQ dead letter messages may compromise performance. Exercise caution.

Configuring a Dead Letter Exchange and Routing Key

A dead letter exchange and routing key can be configured on a client or on the RabbitMQ console.

Configuring an Exchange Key and a Routing Key on a Java Client

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**.

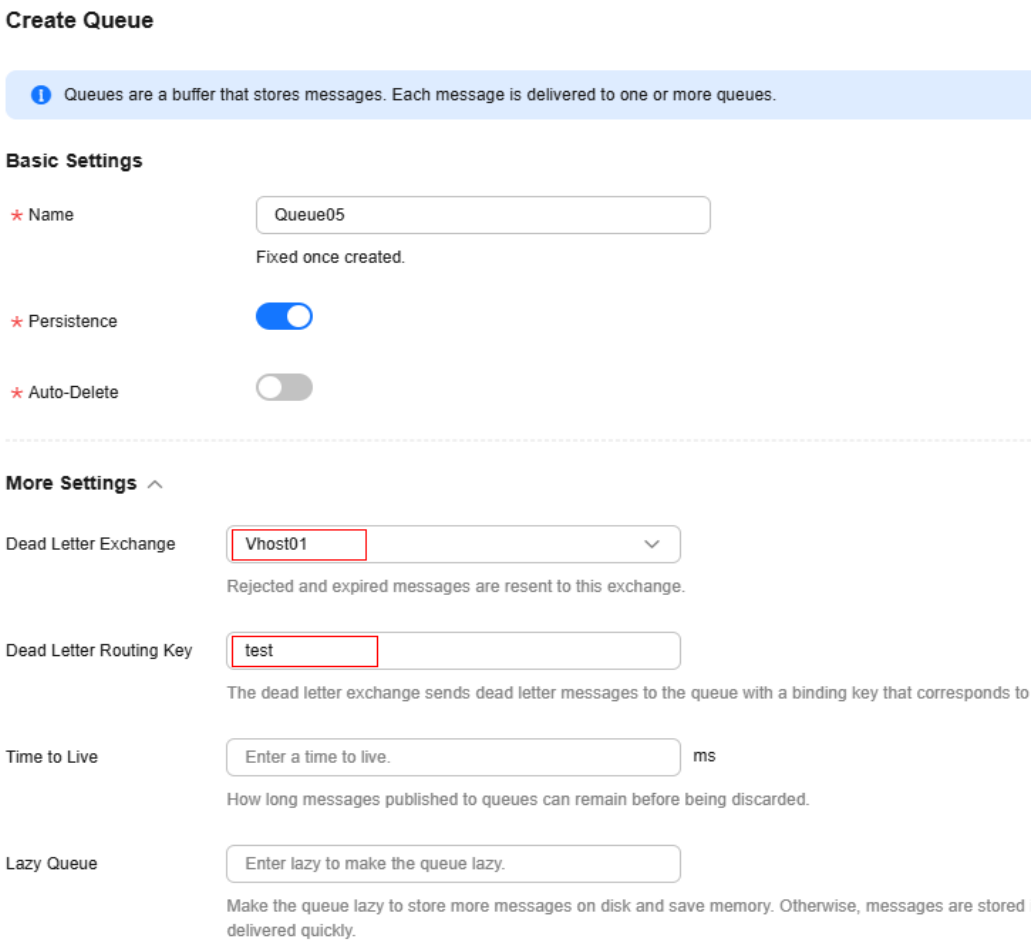
```
channel.exchangeDeclare("some.exchange.name", "direct");

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);
```

Configuring a Dead Letter Exchange and Routing Key on the RabbitMQ Console

Step 1 Create a queue by referring to **Creating a RabbitMQ Queue**, select the dead letter exchange in **Dead Letter Exchange**, and enter the dead letter routing key in **Dead Letter Routing Key**.

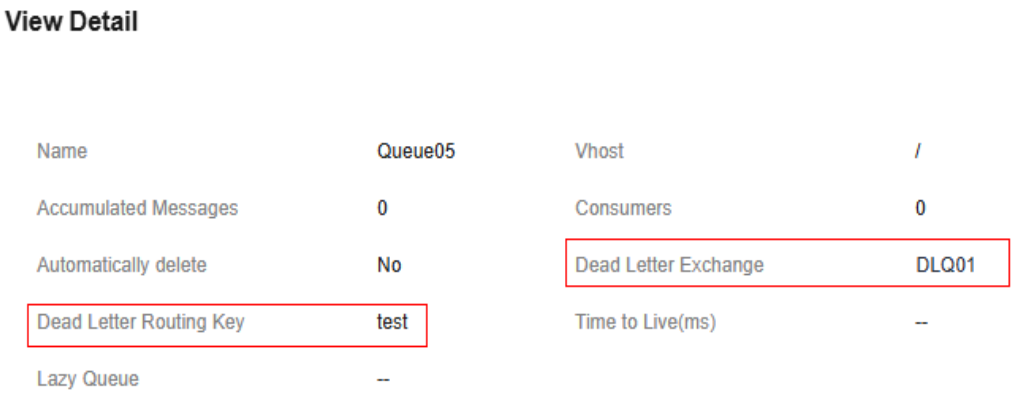
Figure 6-4 Configuring a dead letter exchange and routing key



Step 2 In the queue list, click **View Detail** next to the created queue.

If the dead letter exchange and routing key are the same as those set in **Step 1**, the dead letter exchange and routing key are configured successfully.

Figure 6-5 Viewing queue details



----End

6.3 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.

For details about the message acknowledgment mechanism, see **Consumer Acknowledgment and Publisher Confirms**.

Notes and Constraints

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

Procedure

The following describes how to configure producer and consumer acknowledgment on a Java client.

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**.

6.4 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.

Notes and Constraints

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.

Notes and Constraints

- 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.
- RabbitMQ AMQP-0-9-1 exchanges, queues, and messages support persistence.

Configuring Exchange Persistence

Exchange persistence can be configured on the management UI or RabbitMQ console.

Setting Exchange Persistence on the RabbitMQ Management UI

Step 1 Log in to the **RabbitMQ management UI**.

Step 2 Click the **Exchanges** tab.

Step 3 Create an exchange and set **durable** to **true**, as shown in **Figure 7-1**.

Figure 7-1 Configuring exchange persistence (management UI)

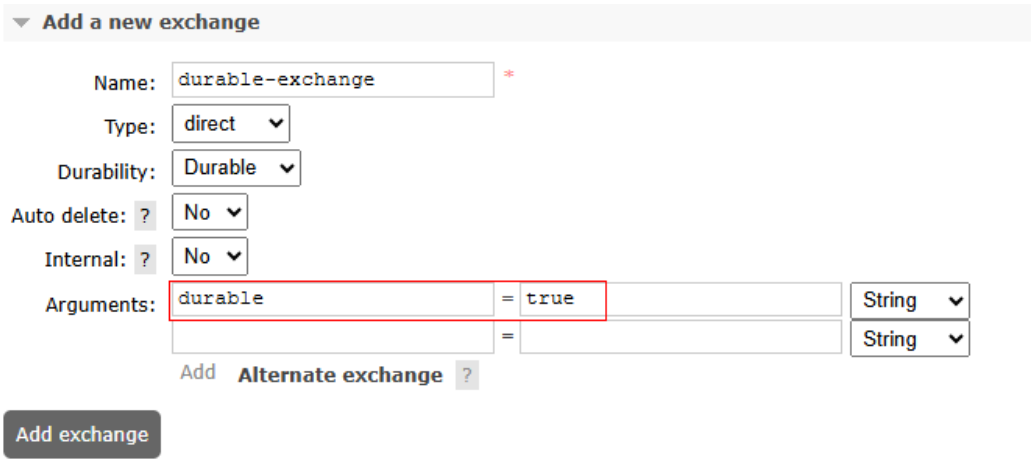


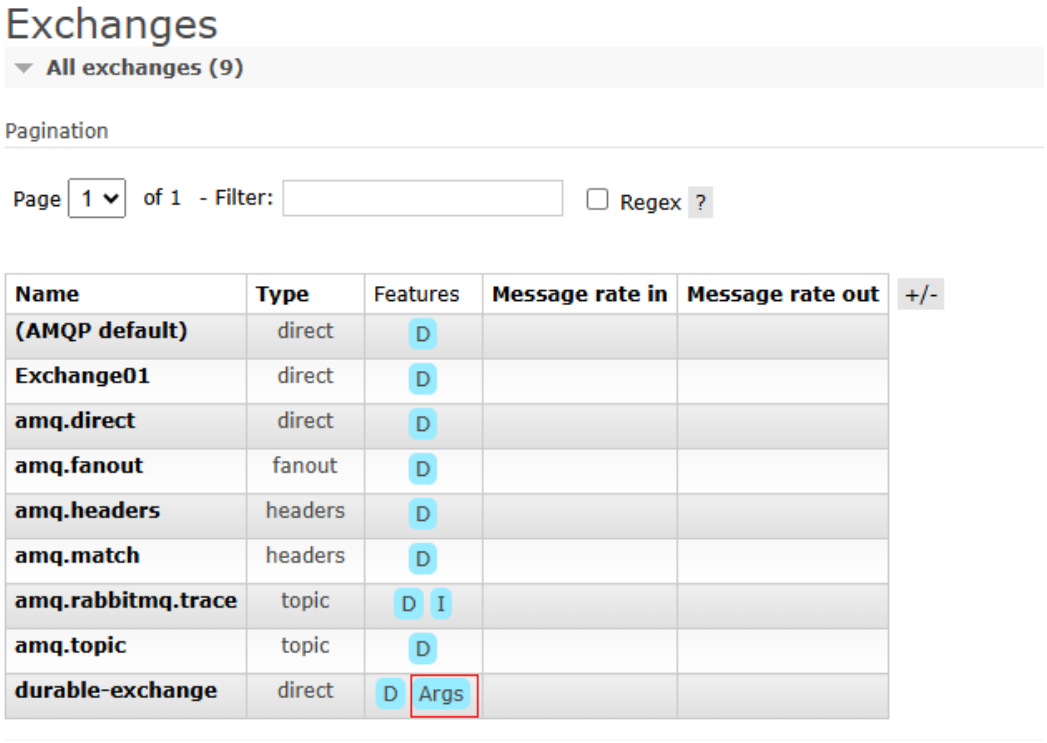
Table 7-1 Exchange creation parameters

Parameter	Description
Name	Exchange name, which is user-defined.
Type	Exchange type. Select a value from the drop-down list. • **direct**: Exchanges route messages to matching queues based on the routing keys. • **fanout**: Exchanges route messages to all bound queues. • **topic**: Exchanges route messages to queues based on routing key wildcard matching. • **headers**: 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).
Durability	Whether to enable persistence. • Durable: The exchange survives server restart. • Transient: The exchange will be deleted after server restarts and needs to be recreated.
Auto delete	Whether to enable automatic deletion. • Yes: This exchange will be automatically deleted when the last bound queue unbound from the exchange. • No: This exchange will not be deleted when the last bound queue unbound from the exchange.

Parameter	Description
Internal	Indicates whether exchanges are for internal use. • Yes: This exchange can only bind another exchange instead of a queue. • No: This exchange can bind other exchanges and queues.
Arguments	Set **durable=true** to configure exchange persistence.

Step 4 In the exchange list, move the cursor to **Args** of the new exchange, as shown in Figure 7-2. If **durable: true** is displayed in the floating window, the exchange persistence is configured successfully.

Figure 7-2 Exchange persistence configured (management UI)

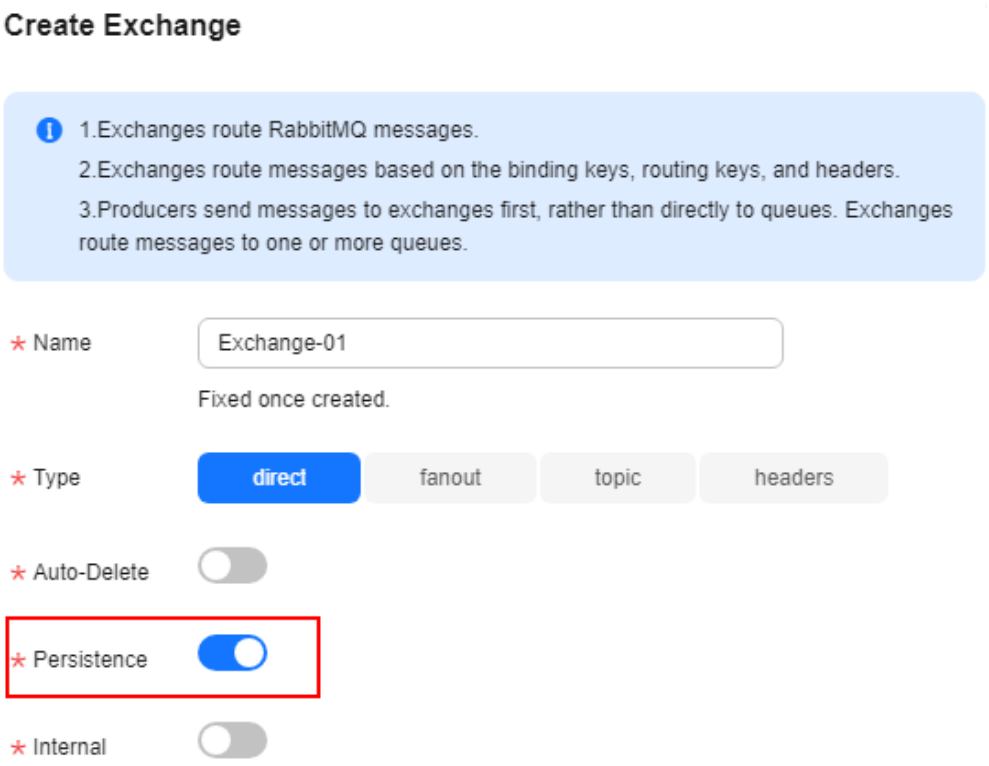


----End

Setting Exchange Persistence on the RabbitMQ Console

Step 1 Create an exchange by referring to **Creating a RabbitMQ Exchange** and configure exchange persistence, as shown in **Figure 7-3**.

Figure 7-3 Configuring exchange persistence (console)



Step 2 In the exchange list, check whether persistence is enabled for the new exchange. If **Yes** is displayed in the **Persistence** column, persistence is enabled, as shown in **Figure 7-4**.

Figure 7-4 Exchange persistence configured (console)

----End

Configuring Queue Persistence

Queue persistence can be configured on the management UI or RabbitMQ console.

Setting Queue Persistence on the RabbitMQ Management UI

Step 1 Log in to the **RabbitMQ management UI**.

Step 2 Click the **Queues** tab.

Step 3 Create a queue and set **Durability** to **Durable**.

Figure 7-5 Creating queues

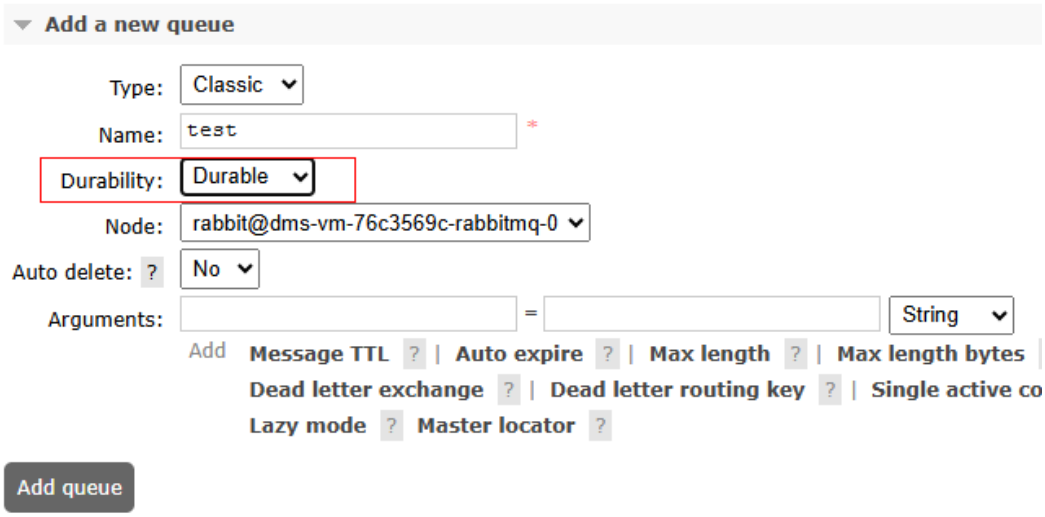


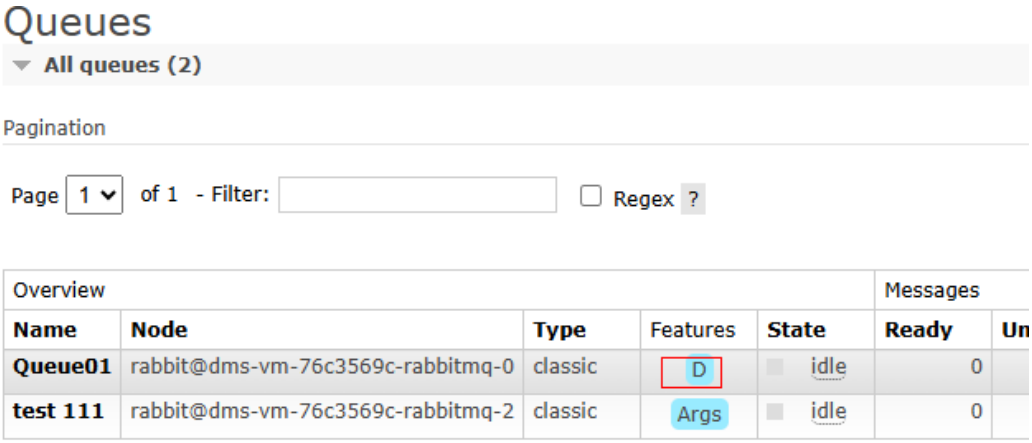
Table 7-2 Queue parameters

Parameter	Description
Type	Queue type.
Name	Name of the single active consumer queue, which is user-defined.
Durability	Whether to enable persistence. Set **Durability** to **Durable**. • Durable: This queue survives after server restart. • Transient: This queue will be deleted after server restart and needs to be recreated.
Node	Node where a single active consumer queue is deployed.
Auto delete	Whether to enable automatic deletion. • Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue. • No: This queue will not be deleted when the last consumer unsubscribes from the queue.
(Optional) Arguments	Queue attribute. You do not need to set it.

Step 4 Click **Add queue**.

Step 5 On the **Queues** page, check whether the new queue is a persistent queue. If **Features** is **D**, the queue is a persistent queue.

Figure 7-6 Viewing queue attributes

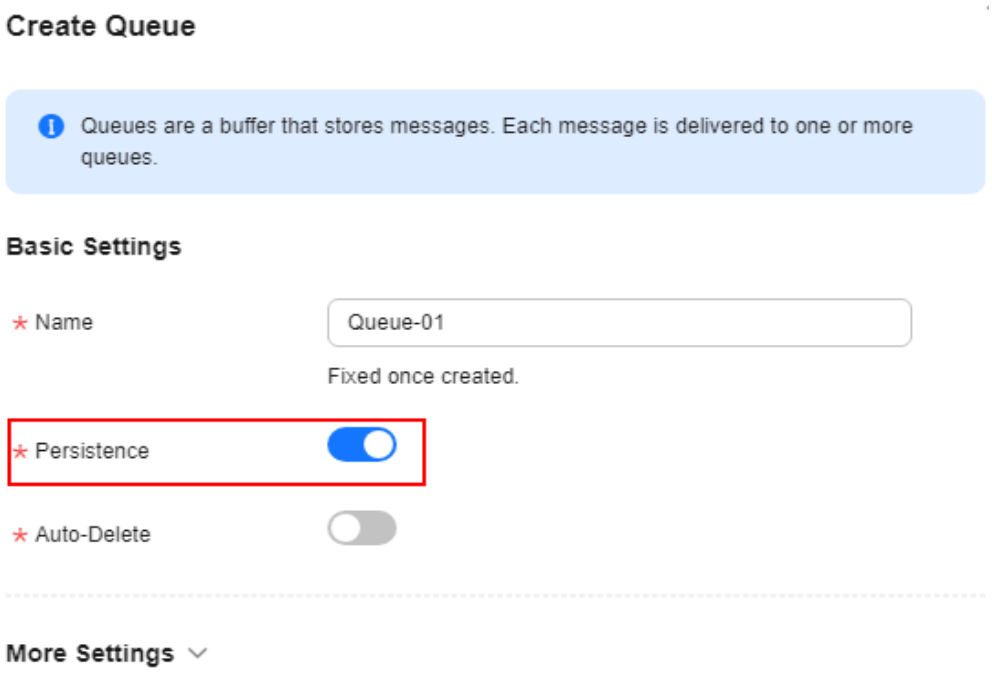


----End

Setting Queue Persistence on the RabbitMQ Console

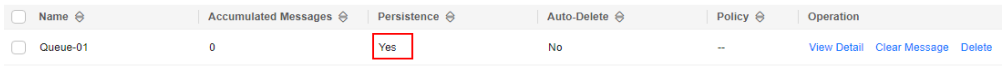
Step 1 Create a queue by referring to **Creating a RabbitMQ Queue** and set the queue persistence, as shown in **Figure 7-7**.

Figure 7-7 Configuring queue persistence (console)



Step 2 In the queue list, check whether persistence is enabled for the new queue. If **Yes** is displayed in the **Persistence** column, as shown in **Figure 7-8**, persistence is enabled.

Figure 7-8 Queue persistence configured (console)



----End

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**.

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

Notes and Constraints

If the queue TTL and message TTL are both configured, the smaller one takes effect.

Procedure

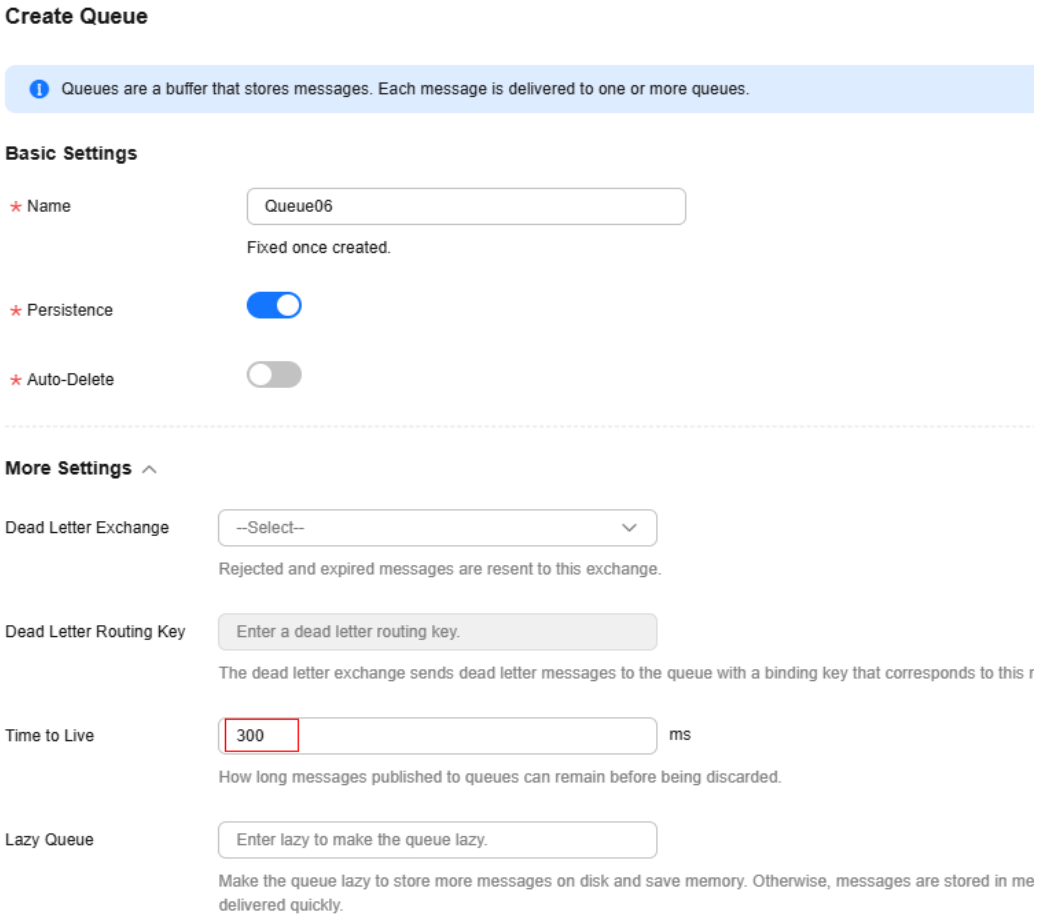
You can configure TTL for messages and queues. Message TTL can be configured in the following ways:

- 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.

Setting Queue TTL on the RabbitMQ Console

Step 1 Create a queue by referring to **Creating a RabbitMQ Queue** and set **Time to Live**.

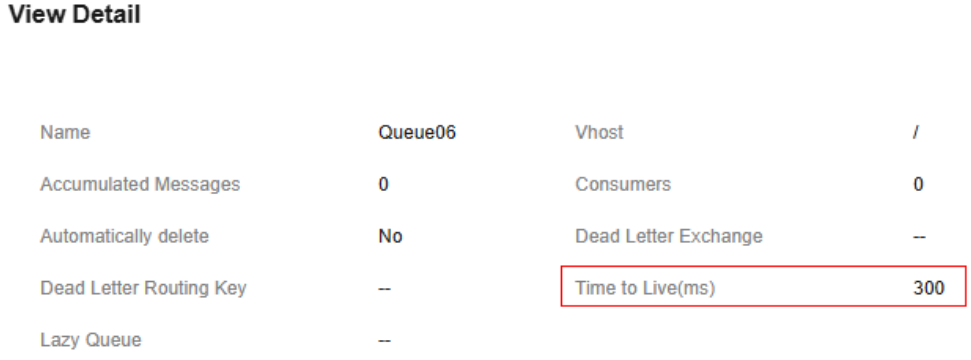
Figure 7-9 Setting the time to live



Step 2 In the queue list, click **View Detail** next to the created queue.

If **Time to Live(ms)** is the same as that set in **Step 1**, the TTL is set successfully.

Figure 7-10 Viewing the TTL



----End

Setting Queue TTL on a Java Client

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 **RabbitMQ console**.

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

Step 3 Search for a RabbitMQ instance by specifying the filters. Current filters include the name, status, version, instance type, specification, used/available storage space, billing mode, AZ, instance ID, private connection address, public connection address, enterprise project, and resource tag. Only enterprise users can use enterprise projects to filter resources. **Table 8-1** describes the various possible statuses of a RabbitMQ instance.

Table 8-1 RabbitMQ instance status description

Status	Description
Creating	The instance is being created.
Creation failed	The instance failed to be created.

Status	Description
Running	The instance is running properly. Only instances in the **Running** state can provide services.
Faulty	The instance is not running properly.
Starting	The status between **Frozen** and **Running**.
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 **Running** and **Frozen**.
Upgrading	The instance is being upgraded.
Rolling back	The instance is being rolled back.
Binning	The instance is being moved to Recycle Bin.
Binned	The instance is in Recycle Bin.
Recovering	The instance is being recovered from Recycle Bin.

Step 4 Click the name of the RabbitMQ instance and view the instance details.

Table 8-2 and Table 8-3 describe the parameters for connecting to an instance. For details about other parameters, see the **Basic Information** tab page of the instance on the console.

Table 8-2 Connection parameters (RabbitMQ 3.x.x)

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.

Table 8-3 Connection parameters (RabbitMQ AMQP-0-9-1)

Parameter	Description
Instance Address (Private Network)	Address for connecting to the instance when public access is disabled.
ACL	Whether to enable ACL. When ACL is enabled, message production and consumption require authentication.

----End

Modifying Basic Information of a RabbitMQ Instance

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

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

Step 3 Click a RabbitMQ instance name to go to the instance details page.

Step 4 Modify the following parameters if needed:

Table 8-4 Modifiable RabbitMQ parameters

Parameter	How to Modify	Result
Instance Name	Click , enter a new name, and click . 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 . 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 . 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 . Modifying this parameter does not restart the instance.	The modification result is displayed in the upper right corner of the page.

Parameter	How to Modify	Result
Public Access	See **Configuring RabbitMQ Public Access**.	You will be redirected to the **Background Tasks** page, which displays the modification progress and result.
ACL	Click  or  to configure ACL. This parameter is supported only for RabbitMQ AMQP-0-9-1 instances. **WARNING** Enabling ACL will disconnect clients without authentication configuration.	The modification result is displayed in the upper right corner of the page.

----End

Related Documents

- To query RabbitMQ instance information by calling an API, see **Querying an Instance**.
- To modify basic information of a RabbitMQ instance by calling an API, see **Modifying Instance Information**.

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.

Notes and Constraints

- The client connection addresses cannot be viewed on the management UI of RabbitMQ AMQP-0-9-1.
- A client's connection addresses can be viewed only when the client is connected to a RabbitMQ instance.

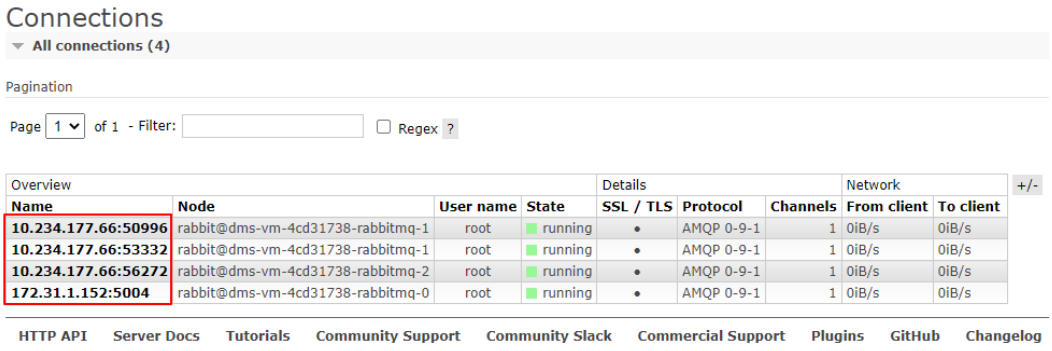
Procedure

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



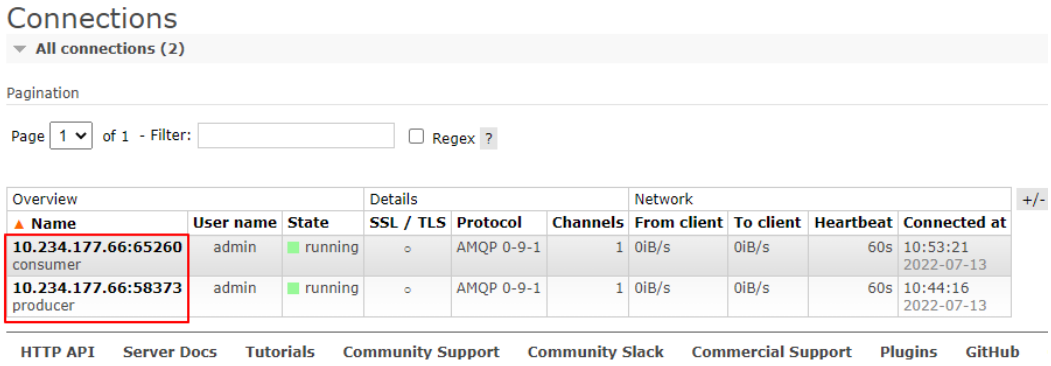
A client can function as a producer to create messages and as a consumer to retrieve messages. The producer and consumer IP addresses are the same, as shown in **Figure 8-1**, and are difficult to distinguish. To differentiate between producer and consumer IP addresses, you can set the **clientProperties** 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)



----End

8.3 Viewing RabbitMQ Background Tasks

Performing an instance operation listed in **Table 8-5** starts a background task. The tasks are displayed on the **Background Tasks** page. Tasks can be cleared there.

Table 8-5 Background tasks

Task Name	Description
Creating an Instance	Creates a RabbitMQ instance.
Plug-in change	• Enables a plug-in. • Disables a plug-in.
Enable public access	Enables public access.
Disable public access	Disables public access.
Modify Specifications	• Increases the storage space. • Increases brokers. • Increases a broker flavor. • Decreases a broker flavor.

Procedure

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

Step 2 Click in the upper left corner of the console and select the region where the RabbitMQ instance is located.

Step 3 Click the name of a RabbitMQ instance to go to the **Overview** page.

Step 4 In the navigation pane on the left, choose **Instance** > **Background Tasks**.

Step 5 On the **Current Tasks** tab page, select a time period from the drop-down box, enter a keyword in the search box, and press **Enter**. Tasks started in the specified time period are displayed.

On this page, you can also perform the following operations:

- To refresh task statuses, click .
- To clear a task, click **Delete**. In the displayed **Delete Task** dialog box, click **OK**.

  Only tasks in the **Successful** or **Failed** state can be deleted.

----End

Related Document

To view a background task list by calling an API, see **Listing Background Tasks**.

8.4 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 **Instance** > **Tags** page of the created instance. Tags can be deleted.

If your organization has configured tag policies for DMS for RabbitMQ, add tags to RabbitMQ instances based on the tag policies. If a tag added on the **Instance** > **Tags** page does not comply with the tag policies, the tag fails to be added.

A tag consists of a tag key and a tag value. **Table 8-6** lists the tag key and value requirements.

Table 8-6 Tag key and value requirements

Name	Rules
Tag key	• Cannot be left blank. • Must be unique for the same instance. • Can contain 1 to 128 characters. • Can contain letters, digits, spaces, and special characters _.:=+-@ • Cannot start or end with a space. • Cannot start with _**sys**_.
Tag value	• Can contain 0 to 255 characters. • Can contain letters, digits, spaces, and special characters _.:=+-@ • Cannot start or end with a space in instance creation.

Notes and Constraints

A maximum of 20 tags can be added to a RabbitMQ instance.

Procedure

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

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

Step 3 Click the desired instance to view its details.

Step 4 In the navigation pane, choose **Instance** > **Tags**.

Step 5 Add, edit, or delete tags as required.

Table 8-7 Tag operations

Operation	Procedure
Adding a tag	1. Click **Edit Tag**. 2. Click **Add Tag** to set tags with **Tag key** and **Tag value**. If you have predefined tags, select a predefined pair of tag key and value, and click **Add**. A maximum of **20 tags** can be added. 3. Click **OK**. View the new tag on the tag list page.
Deleting a tag	1. Click **Edit Tag**. 2. Modify the tag key and value, and click **OK**. On the tag list page, view the new tag key and value.
Editing a tag	1. Click **Edit Tag**. 2. In the row containing the tag to be deleted, click **Delete**. Then, click **OK** to delete the tag. The tags are deleted when they are no longer displayed in the tag list.

----End

8.5 Configuring RabbitMQ Recycling Policies

If recycling is enabled, deleted instances and their data are retained in Recycle Bin, and can be recovered during the retention period. Once the retention period expires, instances in Recycle Bin will be deleted permanently.

Recycling is disabled by default.

Constraints

- Pay-per-use instance in Recycle Bin will not generate fees, but their storage will.
- Yearly/Monthly instances will be moved to Recycle Bin upon unsubscription. After that, they will not generate fees, but their storage will.
- Yearly/Monthly instances will be changed to pay-per-use ones upon successful recovery.
- Removing or unsubscribing instances in the grace or retention period deletes them permanently.

Enabling Recycling

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

Step 2 Click  in the upper left corner to select a region.

Step 3 In the navigation pane, choose **Recycle Bin**.

Step 4 Click **Modify Recycling Policy** and the **Modify Recycling Policy** dialog box is displayed.

Step 5 Enable **Recycle Bin**, specify **Retention Days** (1–7), and click **OK**.

----End

Recovering RabbitMQ Instances

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

Step 2 Click  in the upper left corner to select a region.

Step 3 In the navigation pane, choose **Recycle Bin**.

Step 4 Recover RabbitMQ instances using either of the following methods:

- Select one or more RabbitMQ instances and click **Recover** in the upper left corner.
- In the row containing the desired RabbitMQ instance, click **Recover**.

Step 5 In the displayed **Recover Instance** dialog box, click **OK**.

It takes 3 to 10 minutes to recover an instance. You can view recovered instances on the **RabbitMQ Instances** page.

----End

Modifying Retention Days

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

Step 2 Click  in the upper left corner to select a region.

Step 3 In the navigation pane, choose **Recycle Bin**.

Step 4 Click **Modify Recycling Policy** and the **Modify Recycling Policy** dialog box is displayed.

Step 5 Modify the retention days (1–7) and click **OK**.

Changes to the retention period apply only to instances deleted after the changes.

----End

Exporting Instances in the Recycle Bin

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

Step 2 In the navigation pane, choose **Recycle Bin**.

Step 3 Choose **Export** > **Export all data to an XLSX file** or **Export** > **Export selected data to an XLSX file**.

----End

Deleting Instances Permanently

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

Step 2 Click  in the upper left corner to select a region.

Step 3 In the navigation pane, choose **Recycle Bin**.

Step 4 Delete instances using either of the following methods:

- Select one or more RabbitMQ instances and click **Delete** in the upper left corner.
- In the row containing the desired RabbitMQ instance, click **Delete**.

Step 5 In the displayed **Delete Instance** dialog box, enter **DELETE** and click **OK**.

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

----End

Disabling Recycling

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

Step 2 Click  in the upper left corner to select a region.

Step 3 In the navigation pane, choose **Recycle Bin**.

Step 4 Click **Modify Recycling Policy** and the **Modify Recycling Policy** dialog box is displayed.

Step 5 Disable **Recycle Bin** and click **OK**.

----End

8.6 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.

Notes and Constraints

Unavailable for RabbitMQ AMQP-0-9-1.

Prerequisite

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

Procedure

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

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

Step 3 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. Choose ··· > **Reset Password** in the upper right corner.

Step 4 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.

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

----End

8.7 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 8-8 lists plug-ins supported by RabbitMQ. **The ports of the plug-ins cannot be changed.**

Table 8-8 Plug-ins

Name	Function	Port
rabbitmq_federation	Federation	-
rabbitmq_shovel	Message moving	-
rabbitmq_consistent_hash_exchange	Support for x-consistent-hash. x-consistent-hash exchanges can be created after this plugin is enabled.	-

Notes and Constraints

- Unavailable for RabbitMQ AMQP-0-9-1.
- To enable plug-ins for RabbitMQ instances created before July 1, 2020, submit a service ticket.
- To enable plug-in rabbitmq_consistent_hash_exchange for RabbitMQ instances created before November 14, 2020, submit a service ticket.
- Enabling plug-ins does not restart instances.
- The rabbitmq_shovel and rabbitmq_federation plug-ins can be enabled only for specific instances. For details, see **Table 8-9**.

Table 8-9 Instances for which plug-ins can be enabled

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

Enabling RabbitMQ Plug-ins

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

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

Step 3 Click the desired instance to view its details.

Step 4 In the navigation pane, choose **Instance** > **Plug-ins**. Then, click **Enable** in the row containing the desired plug-in.

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

----End

8.8 Exporting the RabbitMQ Instance List

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

Procedure

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

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

Step 3 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

8.9 Deleting a RabbitMQ Instance

For pay-per-use RabbitMQ instances, you can delete one or more of them in batches on the console. For yearly/monthly RabbitMQ instances, if you no longer

need them, choose **More** > **Unsubscribe** in the **Operation** column. RabbitMQ instances will be automatically deleted upon unsubscription.

Manage deleted instances using recycling policies. When no recycling policies are enabled, deleting instances clears their data permanently. Recycle bin policies are disabled by default. To enable them, see **Enabling Recycling**.

Prerequisite

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

Procedure

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

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

Step 3 Delete pay-per-use RabbitMQ instances in one of the following ways:

- Select one or more RabbitMQ instances and click **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 ··· > **Delete**.

WARNING

When no recycling policies are enabled, deleting instances clears their data permanently.

Step 4 In the **Delete Instance** dialog box, enter **DELETE** and click **OK** to delete the RabbitMQ instance.

It takes 1–60s to delete a RabbitMQ instance.

The RabbitMQ instances are deleted when they are no longer displayed in the instance list.

----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.

Notes and Constraints

Unavailable for RabbitMQ AMQP-0-9-1 instances.

Prerequisite

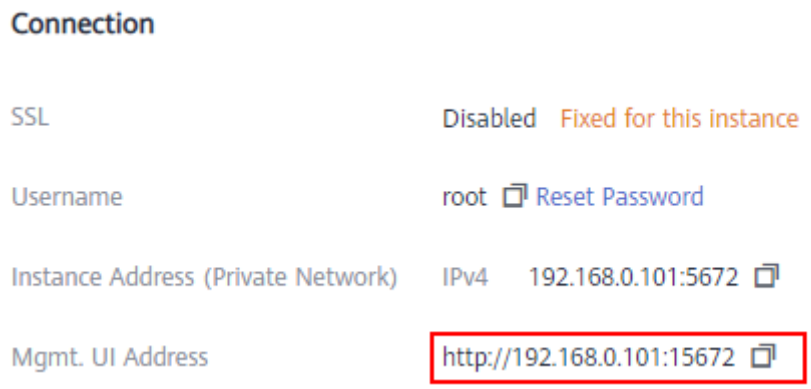
You have obtained the username and password set in instance creation. If you forget the password, reset the password by referring to **Resetting the RabbitMQ Instance Password**.

Procedure

Step 1 Obtain the management address of an instance.

1. Log in to the **RabbitMQ console**.
2. In the upper left corner, click and select a region.
3. 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**.

Figure 8-3 Viewing the management UI address (without SSL)



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

1. In the **Network** section on the **Overview** 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.

- 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 a Custom ECS**.

Figure 8-4 Logging in to the management UI

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

----End

9 Modifying Instance Specifications

9.1 Modifying RabbitMQ Instance Specifications

After creating a RabbitMQ instance, you can increase or decrease its specifications. For details, see **Table 9-1** and **Table 9-2**.

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

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

Table 9-2 Supported specification changes (for RabbitMQ AMQP-0-9-1)

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

Impact

Table 9-3 Impact of specification modification

Modified Object	Impact
Brokers	• During the broker increase, the RabbitMQ workload balance process is restarted, triggering a **master/standby switchover** and causing **services to temporarily stutter**. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours. • The change duration depends on the number of added brokers, and is 5–10 minutes per broker. The total change duration = **Single change duration × Number of added brokers**.
Storage space	• Storage space expansion **does not affect services**. • You can expand the storage space **20 times** for each instance. • 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. • The total change duration is **within 5 minutes**.

Modified Object	Impact
Broker/ Instance flavor	• For **single-node** RabbitMQ 3.x.x instances, nodes are restarted and services may temporarily **stutter in minutes** during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours. • For cluster RabbitMQ 3.x.x instances **not configured with mirrored/quorum queues**, nodes are restarted in sequence and services may **stutter in minutes** during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours. • For cluster RabbitMQ 3.x.x instances **not configured with mirrored/quorum queues**, nodes are restarted in sequence and services may **stutter for several seconds** during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours. • During the modification of cluster RabbitMQ 3.x.x instances, connections to the changed nodes will be switched to other nodes, posing overload risks (for example, connections excess or memory high watermark) on them. You are advised to use them within the instance specifications. For more information, see **Specifications**. • RabbitMQ AMQP-0-9-1: For single-node and cluster instances, **intermittent disconnections may occur for several seconds** during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours. • **Persistent exchanges, queues, and messages are required.** Otherwise, messages may be lost after node restarts. For details, see **Configuring RabbitMQ Persistence**. • Brokers will be restarted in sequence during an instance change, which depends on the number of brokers. The change duration per single broker is 5–10 minutes. The total change duration = **Single change duration × Number of brokers**.

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

```
ConnectionFactory connectionFactory = new ConnectionFactory();
//Set a service address.
connectionFactory.setHost("localhost");
//Set a port.
connectionFactory.setPort(5672);
//Auto retry:
connectionFactory.setAutomaticRecoveryEnabled(true);
connectionFactory.setNetworkRecoveryInterval(5);
connectionFactory.setTopologyRecoveryEnabled(true);
```

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 Specifications of a RabbitMQ 3.x.x Instance

The following describes how to change specifications of a RabbitMQ 3.x.x instance.

Increasing the Storage Space (for 3.x.x)

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

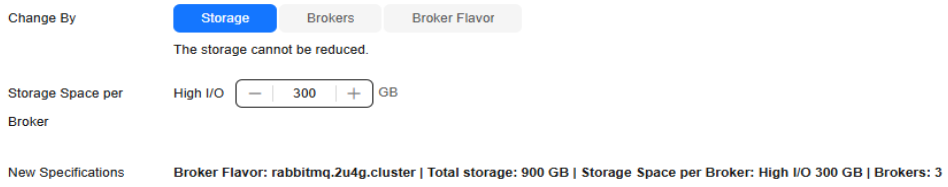
Step 2 In the upper left corner, click and select a region.

Step 3 Modify the instance specifications using either of the following methods:

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

Step 4 Specify the required storage space.

Figure 9-1 Expanding a storage space (RabbitMQ 3.x.x)



1. For **Change By**, select **Storage**. For **Storage Space per Broker**, specify a new storage space. The storage space varies by instance specifications. For details, see **Specifications**. Click **Next**.
2. Confirm the configurations and click **Submit**.
3. Return to the instance list and check whether the change succeeded.
   - If the instance status has changed from **Changing** to **Running**, the change succeeded. View the new storage space (Storage space per broker × Number of brokers) in the **Used/Available Storage Space (GB)** column in the instance list.
   - If the instance status has changed from **Changing** to **Change failed**, the change failed. Move the cursor over **Change failed** to check the failure cause.

     Instances in the **Change failed** state cannot be modified, or deleted. After the instance status automatically changes from **Change failed** to **Running**, you can continue to perform operations. If the status does not change to **Running**, contact customer service.

----End

Increasing the Broker Quantity (V3.x.x)

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

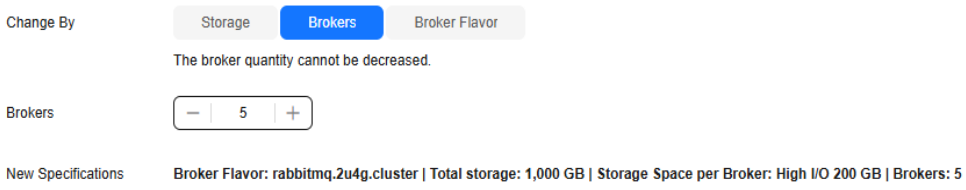
Step 2 In the upper left corner, click and select a region.

Step 3 Modify the instance specifications using either of the following methods:

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

Step 4 Specify the required number of brokers.

Figure 9-2 Increasing brokers (RabbitMQ 3.x.x)



1. For **Modify By**, select **Brokers**. Then, enter the number of brokers. The broker quantity range varies by instance specifications. For details, see **Specifications**. If public access has been enabled, configure EIPs for the new brokers. Then click **Next**.
2. Confirm the configurations and click **Submit**.
3. Return to the instance list and check whether the change succeeded.
   - If the instance status has changed from **Changing** to **Running**, the change succeeded. You can check the new broker quantity in the **Flavor** column.
   - If the instance status has changed from **Changing** to **Change failed**, the change failed. Move the cursor over **Change failed** to check the failure cause.

     Instances in the **Change failed** state cannot be modified, or deleted. After the instance status automatically changes from **Change failed** to **Running**, you can continue to perform operations. If the status does not change to **Running**, contact customer service.

----End

Increasing/Decreasing a Broker Flavor (V3.x.x)

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

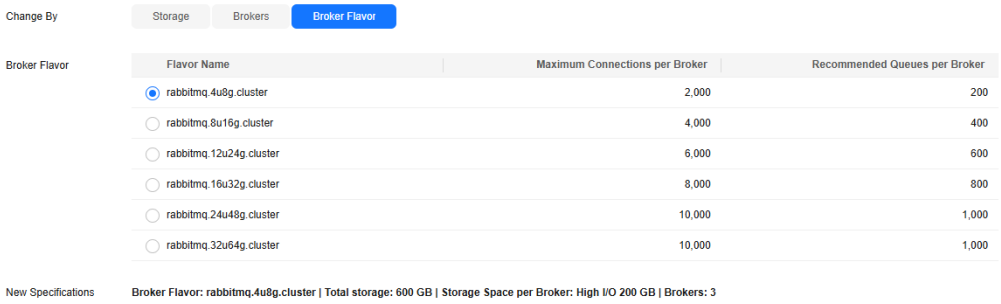
Step 2 In the upper left corner, click and select a region.

Step 3 Modify the instance specifications in either of the following ways:

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

Step 4 Specify the required broker flavor.

Figure 9-3 Increasing/Decreasing a broker flavor (RabbitMQ 3.x.x)



1. For **Modify By**, select **Broker Flavor**. Then, select a new flavor and click **Next**.
2. Confirm the configurations and click **Submit**.
3. Return to the instance list and check whether the change succeeded.
   - If the instance status has changed from **Changing** to **Running**, the change succeeded. You can check the new broker flavor in the **Flavor** column.
   - If the instance status has changed from **Changing** to **Change failed**, the change failed. Move the cursor over **Change failed** to check the failure cause.

     Instances in the **Change failed** state cannot be modified, or deleted. After the instance status automatically changes from **Change failed** to **Running**, you can continue to perform operations. If the status does not change to **Running**, contact customer service.

----End

Modifying Specifications of a RabbitMQ AMQP-0-9-1 Instance

The following describes how to change specifications of a RabbitMQ AMQP-0-9-1 instance.

Expanding a Storage Space (AMQP-0-9-1)

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

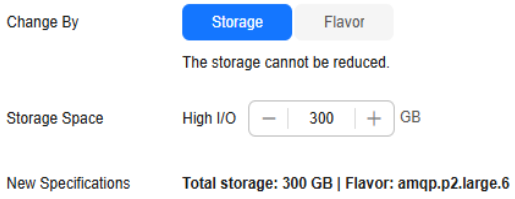
Step 2 In the upper left corner, click  and select a region.

Step 3 Modify the instance specifications in either of the following ways:

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

Step 4 Specify the required storage space.

Figure 9-4 Expanding a storage space (RabbitMQ AMQP-0-9-1)



1. For **Modify By**, select **Storage**. For **Storage**, specify a new storage space. The storage space varies by instance specifications. For details, see **Specifications**. Click **Next**.
2. Confirm the configurations and click **Submit**.
3. Return to the instance list and check whether the change succeeded.
   - If the instance status has changed from **Changing** to **Running**, the change succeeded. View the new storage space (Storage space per broker x Number of brokers) in the **Used/Available Storage Space (GB)** column in the instance list.
   - If the instance status has changed from **Changing** to **Change failed**, the change failed. Move the cursor over **Change failed** to check the failure cause.

     Instances in the **Change failed** state cannot be modified, or deleted. After the instance status automatically changes from **Change failed** to **Running**, you can continue to perform operations. If the status does not change to **Running**, contact customer service.

----End

Increasing an Instance Flavor (AMQP-0-9-1)

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

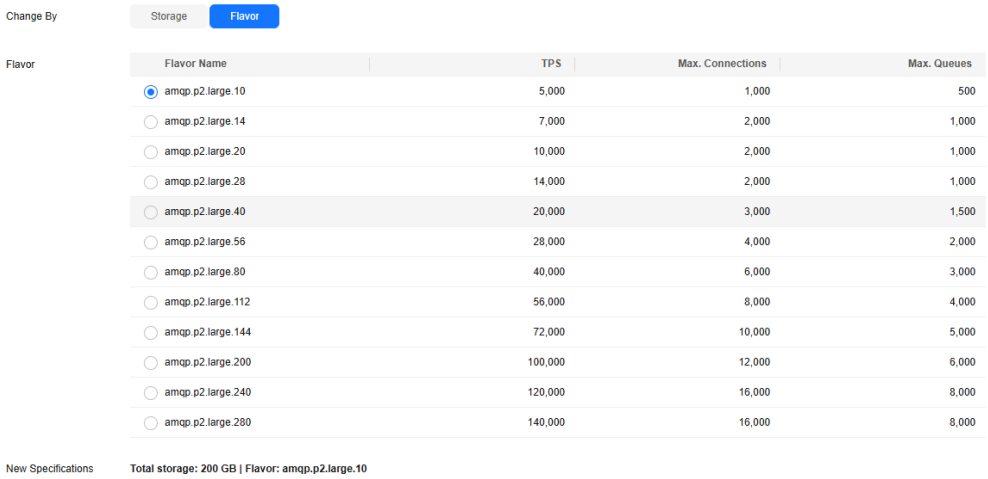
Step 2 In the upper left corner, click  and select a region.

Step 3 Modify the instance specifications in either of the following ways:

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

Step 4 Increase the instance flavor as required.

Figure 9-5 Increasing an instance flavor (RabbitMQ AMQP-0-9-1)



1. For **Change By**, select **Flavor**. Then, select a new flavor and click **Next**.
2. Confirm the configurations and click **Submit**.
3. Return to the instance list and check whether the change succeeded.
   - If the instance status has changed from **Changing** to **Running**, the change succeeded. You can check the new broker flavor in the **Flavor** column.
   - If the instance status has changed from **Changing** to **Change failed**, the change failed. Move the cursor over **Change failed** to check the failure cause.

     Instances in the **Change failed** state cannot be modified, or deleted. After the instance status automatically changes from **Change failed** to **Running**, you can continue to perform operations. If the status does not change to **Running**, contact customer service.

----End

Related Document

To change specifications of a RabbitMQ instance by calling an API, see **Modifying Specifications of Instances with New Flavors**.

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.

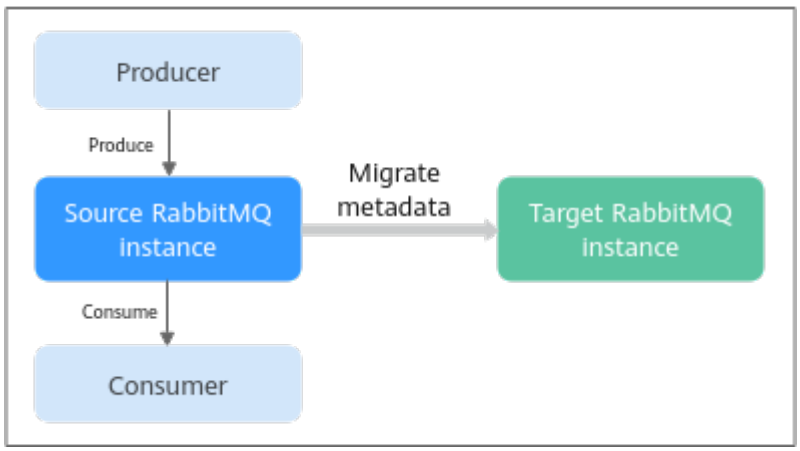
Prerequisite

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.

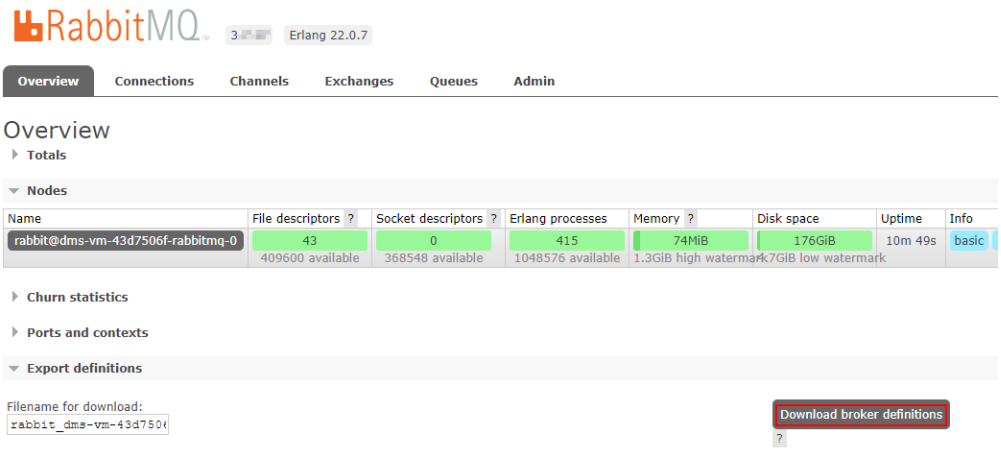
Figure 10-1 Migrating metadata



Do as follows:

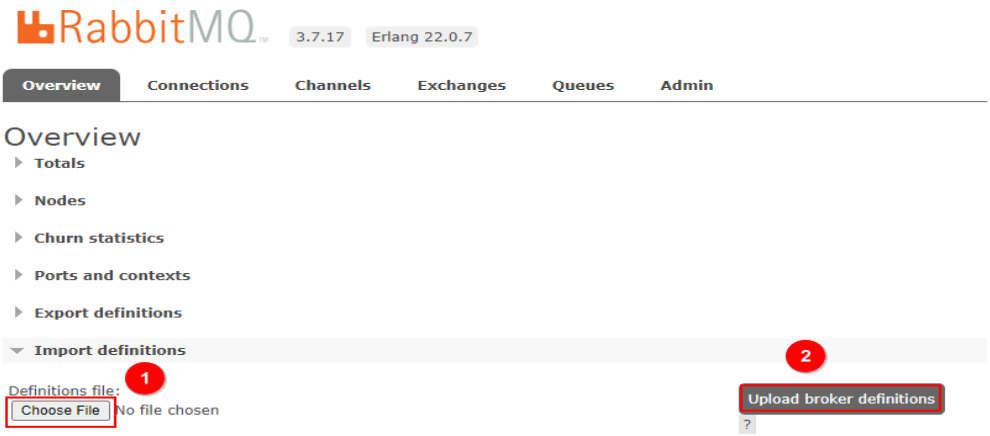
1. Log in to the management UI of the source RabbitMQ. On the **Overview** tab page, click **Download broker definitions** to export the metadata.

   Figure 10-2 Exporting metadata

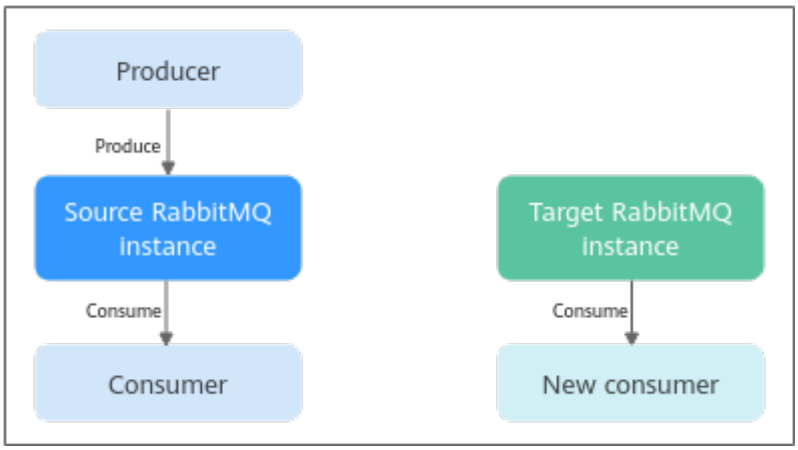
2. Log in to the management UI of the target RabbitMQ. On the **Overview** tab page, click **Choose File** and select the metadata exported in **Step 1.1**, and click **Upload broker definitions** to upload the metadata.

   Figure 10-3 Importing metadata

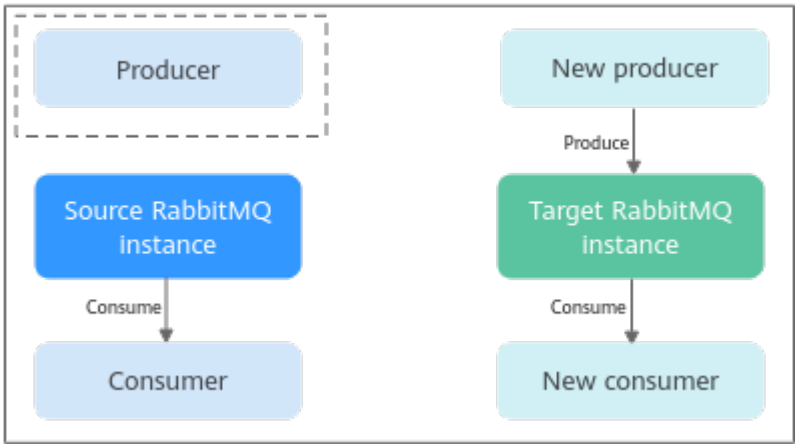
Step 2 Add new consumers for the target RabbitMQ instance.

Figure 10-4 Adding new consumers



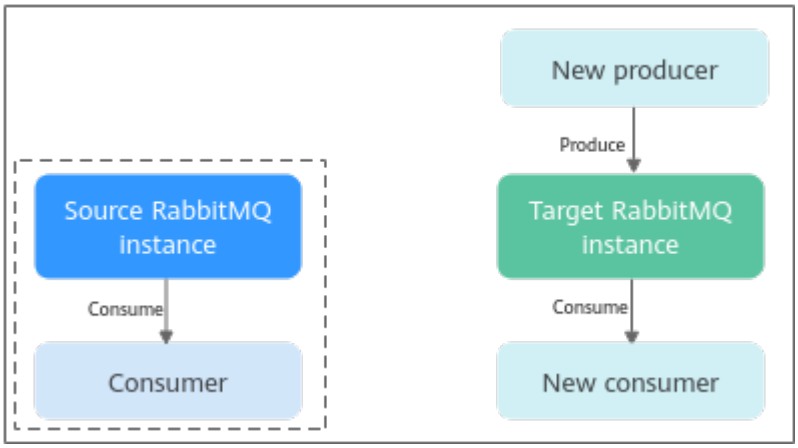
Step 3 Add new producers for the target RabbitMQ instance and remove the producers of the source RabbitMQ instance. The old consumers continue consuming messages from the source RabbitMQ instance.

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

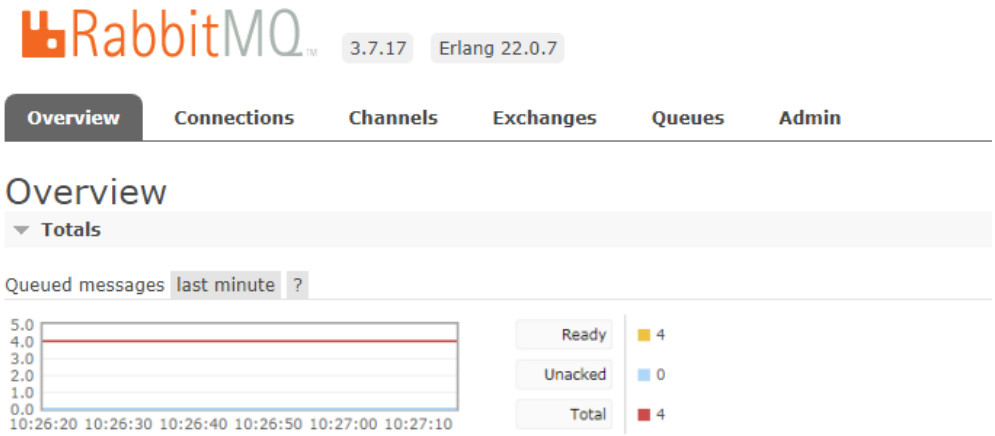
Check After Migration

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

  Table 10-1 Parameters

Parameter	Description
username	Account of the source instance to log in to the RabbitMQ management UI

Parameter	Description
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 Testing Instance Performance

11.1 Testing RabbitMQ Production and Consumption Rate

This section describes performance tests on Distributed Message Service (DMS) for RabbitMQ. The performance is measured by the instance flavors, SSL, producer/consumer quantity, queue quantity and type, and exchange type. The tests cover the following scenarios:

- Test scenario 1 (instance flavors): same exchange, queue, number of producers and consumers, but different instance flavors
- Test scenario 2 (whether SSL is enabled): same exchange, queue, number of producers and consumers, instance flavors, but SSL enabled/disabled
- Test scenario 3 (number of producers/consumers): same exchange, queue, instance flavors, but different numbers of producers and consumers
- Test scenario 4 (single-queue and multi-queue): same exchange, number of producers and consumers, instance flavors, but different number of queues
- Test scenario 5 (queue type): same exchange, number of producers and consumers, instance flavors, but different queue types
- Test scenario 6 (fanout exchange): same instance flavors, fanout exchange, number of queues, but different number of producers and consumers

NOTE

The test result may vary by the network or client conditions.

Test Environment

Perform the following steps to set up the test environment.

1. Purchase cluster RabbitMQ 3.8.35 instances with parameters described in **Table 11-1**. For details about how to purchase them, see **Buying a RabbitMQ Instance**.

CAUTION

- You are not advised to set a **RabbitMQ password** with **special characters**. If special characters are used, they need to be translated before using test scripts. Otherwise, errors occur.
- **Enable public access** and **allow port 15672** in the inbound rules of the security group when purchasing a **rabbitmq-2u4g** instance. In this way, the management UI can be accessed in a browser.

Table 11-1 Instance parameters

Name	Brokers	Flavor	SSL Enabled	Disk Type
rabbitmq-ssl	3	rabbitmq.2u4g.cluster	Yes	Ultra-high I/O
rabbitmq-2u4g	3	rabbitmq.2u4g.cluster	No	Ultra-high I/O
rabbitmq-4u8g	3	rabbitmq.4u8g.cluster	No	Ultra-high I/O
rabbitmq-8u16g	3	rabbitmq.8u16g.cluster	No	Ultra-high I/O
rabbitmq-16u32g	3	rabbitmq.16u32g.cluster	No	Ultra-high I/O

Obtain the private connection addresses on the RabbitMQ instance details page and record the username and password. In addition to a **rabbitmq-2u4g** instance, record the management UI address for later use.

2. For the **rabbitmq-2u4g** instance, **log in to the management UI** and set **queue mirroring**, **lazy queues**, and **quorum queues**.
3. In the **/** virtual host of the **rabbitmq-2u4g** instance, create a fanout exchange. For details, see **Creating a RabbitMQ Exchange**.
4. Obtain the test tool **rabbitmq-perf-test-2.18.0-bin.tar.gz**.

5. Purchase a client server.

   The region, AZ, VPC, subnet, and security group should be consistent with the RabbitMQ instance. The specification is 16 vCPUs | 32 GB. The server is a Linux ECS. For details, see **Purchasing a Custom ECS**.

   Perform the following operations on the ECSs:

   - Install **Java JDK** and configure the environment variables **JAVA_HOME** and **PATH**.
     ```
     export JAVA_HOME=/root/jdk1.8.0_231
     export PATH=$JAVA_HOME/bin:$PATH
     ```
   - Download **rabbitmq-perf-test-2.18.0-bin.tar.gz** and decompress it.
     ```
     tar -zxvf rabbitmq-perf-test-2.18.0-bin.tar.gz
     ```

Test Script

CAUTION

- The test script creates a "**direct**" exchange and a **non-persistence** queue with **automatic deletion** enabled. To test fanout exchanges and quorum queues, append "**--predeclared**" in the script which indicates that custom exchanges and queues are used.
- To test instances with **SSL enabled**, change "**amqp://**" to "**amqps://**" to encrypt data transmission.

Single-queue test script:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x {producer count} -y {consumer count} -z {running time}
```

Table 11-2 Test script parameters

Parameter	Description
Instance username	Username set in **1**.
Instance password	Password set in **1**.
Private connection address	Private connection address obtained in **1**.
Exchange name	Name of the exchange to be tested.
Queue name	Name of the queue to be tested.
Producer count	Number of producers.
Consumer count	Number of consumers.
Running time	Running duration of the script, in seconds.

Multi-queue test script:


```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -x {producer count} -y {consumer count} -z {running time} --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to x
```

Table 11-3 Test script parameters

Parameter	Description
Instance username	Username set in **1**.
Instance password	Password set in **1**.
Private connection address	Private connection address obtained in **1**.
Exchange name	Name of the exchange to be tested.
Producer count	Number of producers.
Consumer count	Number of consumers.
Running time	Running duration of the script, in seconds.
queue-%d	Indicates multiple queues. The queue name prefix is **queue-**. **%d** is a variable ranging from the **--queue-pattern-from** value to the **--queue-pattern-to** value, in integers. For example: **--queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to 3** indicates three queues. The queue names are queue-1, queue-2, and queue-3.

Test Procedure

Step 1 Log in to the client server and go to the **rabbitmq-perf-test-2.18.0/bin** directory.

Step 2 Run the following script to test and record the production and consumption rates of different instance flavors:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x 3 -y 3 -z 300
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -u queue-1 -x 3 -y 3 -z 300
```

Step 3 Run the following script to test and record the production and consumption rates of the "rabbitmq-ssl" instance based on the number of producers, consumers, and queues.

Single-queue test script:

```
./runjava com.rabbitmq.perf.PerfTest -h amqps://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x {producer count} -y {consumer count} -z 300
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqps://test:******@192.168.0.150:5671 -e exchange-direct -s 1024 -u queue-1 -x 1 -y 1 -z 300
```


Multi-queue test script:

```
./runjava com.rabbitmq.perf.PerfTest -h amqps://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -x 3 -y 3 -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to x
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqps://test:******@192.168.0.150:5671 -e exchange-direct -s 1024 -x 3 -y 3 -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to 3
```

Step 4 Run the following script to test and record the production and consumption rates of the "rabbitmq-2u4g" instance based on the number of producers, consumers, and queues.

Single-queue test script:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x {producer count} -y {consumer count} -z 300
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -u queue-1 -x 1 -y 1 -z 300
```

Multi-queue test script:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -x {producer count} -y {consumer count} -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to x
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -x 3 -y 3 -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to 3
```

Step 5 Run the following script to test and record the number of producers, consumers, queues, and production and consumption rates of the "rabbitmq-2u4g" instance. The queue types are lazy, mirroring, and quorum.

Single-queue test script (excluding quorum queues):

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x {producer count} -y {consumer count} -z 300
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -u queue-1 -x 1 -y 1 -z 300
```

Multi-queue test script (excluding quorum queues):

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -x {producer count} -y {consumer count} -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to x
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -x 3 -y 3 -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to 3
```

Single-queue test script (quorum queue):

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x {producer count} -y {consumer count} -z 300 --predeclared
```

Example:


```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -u queue-1 -x 1 -y 1 -z 300 --predeclared
```

Multi-queue test script (quorum queue):

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -x {producer count} -y {consumer count} -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to x --predeclared
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-direct -s 1024 -x 3 -y 3 -z 300 --queue-pattern 'queue-%d' --queue-pattern-from 1 --queue-pattern-to 3 --predeclared
```

Step 6 Run the following script to test and record the production and consumption rates of a fanout exchange of the "rabbitmq-2u4g" instance based on the number of consumers and queues:

Single-queue test script:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://{instance username}:{instance password}@{private connection address} -e {exchange name} -s 1024 -u {queue name} -x {producer count} -y {consumer count} -z 300 --predeclared
```

Example:

```
./runjava com.rabbitmq.perf.PerfTest -h amqp://test:******@192.168.0.150:5672 -e exchange-fanout -s 1024 -u queue-1 -x 1 -y 3 -z 300 --predeclared
```

----End

Test Result

The test results of the six test scenarios are as follows:

Test scenario 1 (instance flavors): same exchange, queue, number of producers and consumers, but different instance flavors

The test parameters are as follows:

- Exchange: The type is direct, which is not persistent and cannot be automatically deleted.
- Queue: The type is classic. The number of queues is 3. The queue is not persistent and will be automatically deleted.
- Producer: The number is 3.
- Consumer: The number is 3.

Table 11-4 Test result

Instance Flavor	Disk Type	Brokers	Production Rate	Consumption Rate
rabbitmq.2u4g.cluster	Ultra-high I/O	3	32,052	25,219
rabbitmq.4u8g.cluster	Ultra-high I/O	3	53,774	47,370

Instance Flavor	Disk Type	Brokers	Production Rate	Consumption Rate
rabbitmq.8u16g.cluster	Ultra-high I/O	3	54,727	45,730
rabbitmq.16u32g.cluster	Ultra-high I/O	3	66,896	51,061

Based on the test results, the following conclusions are drawn (for reference only): **The bigger the instance flavors, the higher the instance performance.**

Test scenario 2 (whether SSL is enabled): same exchange, queue, number of producers and consumers, instance flavors, but SSL enabled/disabled

The test parameters are as follows:

- Flavor: rabbitmq.2u4g.cluster × 3.
- Exchange: The type is direct, which is not persistent and cannot be automatically deleted.
- Queue: The type is classic. The queue is not persistent and will be automatically deleted.

Table 11-5 Test result

SSL Enabled	Producers	Consumers	Queues	Production Rate	Consumption Rate
Yes	1	1	1	7,631	6,291
No	1	1	1	17,104	17,091
Yes	3	3	1	21,819	21,819
No	3	3	1	26,050	26,050
Yes	6	6	1	19,617	19,617
No	6	6	1	24,720	24,720
Yes	1	3	1	13,909	13,909
No	1	3	1	18,732	18,732
Yes	1	6	1	20,006	20,006
No	1	6	1	20,371	20,371
Yes	3	1	1	7,537	5,472
No	3	1	1	28,743	28,743
Yes	6	1	1	7,813	5,310

SSL Enabled	Producers	Consumers	Queues	Production Rate	Consumption Rate
No	6	1	1	26,663	26,662
Yes	3	3	3	32,052	25,219
No	3	3	3	39,951	37,790
Yes	3	3	6	32,972	26,440
No	3	3	6	38,686	37,464
Yes	3	3	10	31,778	25,375
No	3	3	10	39,809	37,912

Based on the test results, the following conclusions are drawn (for reference only): **Connecting to a RabbitMQ instance with SSL enabled provides high security, but the production and consumption rates decrease significantly.** You are advised to use resources with higher flavors to support the SSL encryption and decryption performance. Increasing the number of producers, consumers, and queues can alleviate the performance deterioration caused by SSL connections.

Test scenario 3 (number of producers/consumers): same exchange, queue, instance flavors, but different numbers of producers and consumers

The test parameters are as follows:

- Flavor: rabbitmq.2u4g.cluster × 3.
- Exchange: The type is direct, which is not persistent and cannot be automatically deleted.
- Queue: The type is classic. The number of queue is 1. The queue is not persistent and will be automatically deleted.

Table 11-6 Test result

Producers	Consumers	Production Rate	Consumption Rate
1	1	17,104	17,091
3	3	26,050	26,050
6	6	24,720	24,720
1	3	18,732	18,732
1	6	20,371	20,371
3	1	28,743	28,743
6	1	26,663	26,662

Based on the test results, the following conclusions are drawn (for reference only): **Proper increases of producers and consumers improve production and consumption rates.** In this test, the production rate is the highest when the number of consumer is 1 and producer is 3. When 3 producers are added, the production rate decreases slightly because of performance bottleneck of resources and memory high watermark.

Test scenario 4 (single-queue and multi-queue): same exchange, number of producers and consumers, instance flavors, but different number of queues

The test parameters are as follows:

- Flavor: rabbitmq.2u4g.cluster × 3.
- Exchange: The type is direct, which is not persistent and cannot be automatically deleted.
- Queue: The type is classic. The queue is not persistent and will be automatically deleted.

Table 11-7 Test result

Producers	Consumers	Queues	Production Rate	Consumption Rate
3	3	1	26,050	26,050
3	3	3	39,951	37,790
3	3	6	38,686	37,464
3	3	10	39,809	37,912
6	6	1	24,720	24,720
6	6	10	54,768	45,847

Based on the test results, the following conclusions are drawn (for reference only): **Given the same number of producers and consumers, the production and consumption rates of multi-queue is higher than those of a single queue.** The cause is that the workload among queues is balanced, improving message processing efficiency.

Test scenario 5 (queue type): same exchange, number of producers and consumers, instance flavors, but different queue types

The test parameters are as follows:

- Flavor: rabbitmq.2u4g.cluster × 3.
- Exchange: The type is direct, which is not persistent and cannot be automatically deleted.
- Queue: Classic, lazy, and mirrored queues are set to be non-persistent, and will be automatically deleted. Persistent quorum queues will not be automatically deleted.

Table 11-8 Test result

Producers	Consumers	Queues	Queue Type	Production Rate	Consumption Rate
3	3	1	Classic	26,050	26,050
3	3	1	Lazy	12,291	12,291
3	3	1	Mirrored	16,513	16,100
3	3	1	Quorum	6,771	15,568
3	3	10	Classic	39,809	37,912
3	3	10	Lazy	30,543	29,991
3	3	10	Mirrored	11,130	8,220
3	3	10	Quorum	52,466	51,772
6	6	1	Classic	24,720	24,720
6	6	1	Lazy	9,934	8,728
6	6	1	Mirrored	16,000	14,888
6	6	1	Quorum	6,318	26,841
6	6	10	Classic	54,768	45,847
6	6	10	Lazy	42,556	33,994
6	6	10	Mirrored	54,625	45,853
6	6	10	Quorum	47,133	41,011

Based on the test results, the following conclusions are drawn (for reference only):

- Lazy queue: **The performance of a lazy queue is lower than that of a classic queue. After the queues are added, the performance is significantly improved.**
- Mirrored queue: **The production rate of a mirrored queue is lower than that of a classic queue.** Each time a message is published, the mirrored queue replicates the message to all mirrored nodes, impacting the network transmission and node processing.
- Quorum queue: **The consumption rate of a single queue is higher than the production rate.** The possible cause is that the quorum queue checks consistency each time data is written, which affects the production rate. **The production and consumption rates of multi-queue is significantly higher than those of a single queue.** Therefore, multi-queue applies to scenarios requiring high consistency and concurrency.

Test scenario 6 (fanout exchange): same instance flavors, fanout exchange, number of queues, but different number of producers and consumers

The test parameters are as follows:

- Flavor: rabbitmq.2u4g.cluster × 3.
- Exchange: The type is fanout, which is not persistent and cannot be automatically deleted.
- Queue: The type is classic. The number of queue is 1. The queue is not persistent and will be automatically deleted.

Table 11-9 Test result

Producers	Consumers	Production Rate	Consumption Rate
1	1	17,233	17,232
3	3	35,539	35,231
6	6	32,635	31,682
1	3	27,864	27,864
1	6	31,518	31,518

Based on the test results, the following conclusions are drawn (for reference only): **Proper increases of producers and consumers improve production and consumption rates.**

12 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.

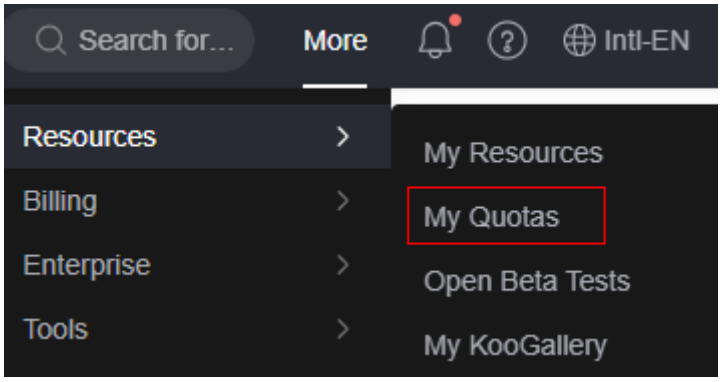
If the current resource quota cannot meet your service requirements, you can apply for a higher quota.

How Do I View My Quota?

1. **Log in to the console**.
2. Click in the upper left corner to select a region and a project.
3. In the upper right corner of the page, choose **Resources** > **My Quotas**.

   The **Quotas** page is displayed.

   Figure 12-1 My Quotas



4. 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**.

13 Viewing Metrics and Configuring Alarms

13.1 Viewing RabbitMQ Metrics

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

Notes and Constraints

On the **Monitoring** > **Monitoring Details** page of the RabbitMQ console, you can select a maximum of 50 resources from the drop-down list at a time. To view the monitoring data of more than 50 resources, do so in batches.

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 **RabbitMQ console**.

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

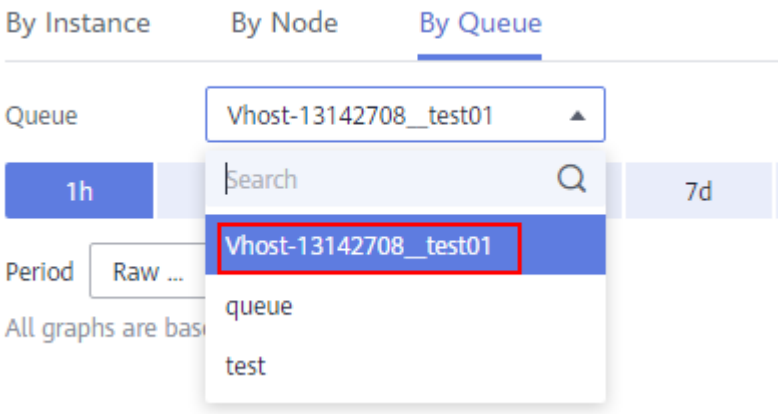
Step 3 View the instance metrics using either of the following methods:

- 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** > **Monitoring Details**. 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**.

Figure 13-1 Queue monitoring



----End

13.2 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 or **APIs** 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 13-1 Instance metrics (RabbitMQ 3.x.x)

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
connections	Connections	Number of connections in the RabbitMQ instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
channels	Channels	Number of channels in the RabbitMQ instance	0–2047	Count	N/A	RabbitMQ instance	1 minute
queues	Queues	Number of queues in the RabbitMQ instance	0–7,000	Count	N/A	RabbitMQ instance	1 minute
consumers	Consumers	Number of consumers in the RabbitMQ instance	0–280,000	Count	N/A	RabbitMQ instance	1 minute
messages_ready	RabbitMQ instance Available Messages	Number of messages that can be consumed in the RabbitMQ instance	0–10,000,000	Count	N/A	RabbitMQ instance	1 minute
messages_unacknowledged	Unacknowledged Messages	Total number of messages that have been consumed but not acknowledged in a RabbitMQ instance	0–10,000,000	Count	N/A	RabbitMQ instance	1 minute
publish	Production Rate	Rate at which messages are produced in the RabbitMQ instance	0–25,000	Count/s	N/A	RabbitMQ instance	1 minute
deliver	Retrieval Rate (Manual Ack)	Rate at which messages are consumed (in the manual acknowledgment scenario) in a RabbitMQ instance	0–25,000	Count/s	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
deliver_no_ack	Retrieval Rate (Auto Ack)	Rate at which messages are consumed (in the automatic acknowledgment scenario) in a RabbitMQ instance	0–50,000	Count/s	N/A	RabbitMQ instance	1 minute
connections_states_running	Normal Connections	Number of **starting**, **tuning**, **opening**, and **running** connections in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
connections_states_flow	Flow Connections	Number of **flow** connections in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
connections_states_block	Blocked/Blocking Connections	Number of **blocking** and **blocked** connections in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
connections_states_close	Closed/Closing Connections	Number of **closing** and **closed** connections in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels_states_runnning	Normal Channels	Number of **starting**, **tuning**, **opening**, and **running** channels in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels_states_flow	Flow Channels	Number of **flow** channels in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
channels_states_block	Blocked/ Blocking Channels	Number of **blocking** and **blocked** channels in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels_states_close	Closed/ Closing Channels	Number of **closing** and **closed** channels in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
queues_states_running	Normal Queues	Number of **running** queues in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute
queues_states_flow	Flow Queues	Number of **flow** queues in the instance This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
connections_states_full	CONNECTIONS States Full	Number of connections that reach the upper limit of channels in the instance	0–1,000,000	Count	N/A	RabbitMQ instance	1 minute

Table 13-2 Instance metrics (RabbitMQ AMQP-0-9-1)

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
connections	Connections	Number of connections in the RabbitMQ instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels	Channels	Number of channels in the RabbitMQ instance	0–2,000	Count	N/A	RabbitMQ instance	1 minute
queues	Queues	Number of queues in the RabbitMQ instance	0–1,000	Count	N/A	RabbitMQ instance	1 minute
consumers	Consumers	Number of consumers in the RabbitMQ instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
messages_ready	Available Messages	Number of messages that can be consumed in the RabbitMQ instance	0–10,000,000	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
publish	Production Rate	Rate at which messages are produced in the RabbitMQ instance	≥ 0	Count/s	N/A	RabbitMQ instance	1 minute
instance_bytes_in_rate	Message Production	Number of bytes produced in the RabbitMQ instance per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance	1 minute
instance_bytes_out_rate	Message Consumption	Number of bytes consumed from the RabbitMQ instance per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance	1 minute
deliver_get	Consumption Rate	Number of messages consumed in real time by the RabbitMQ instance per second	≥ 0	Count/s	N/A	RabbitMQ instance	1 minute
instance_disk_usage	Instance Disk Usage	Instance disk usage	0~100	%	N/A	RabbitMQ instance	1 minute
instance_tps	Instance Transactions Per Second	Number of requests processed by a RabbitMQ instance per second	0–10,000,000	Count	N/A	RabbitMQ instance	1 minute

Broker Metrics

Available only for RabbitMQ 3.x.x.

Table 13-3 Broker metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
fd_used	File Handles	Number of file handles used by RabbitMQ in the node	0–65,535	Count	N/A	RabbitMQ instance node	1 minute
socket_used	Socket Connections	Number of socket connections used by RabbitMQ in the node	0–50,000	Count	N/A	RabbitMQ instance node	1 minute
proc_used	Erlang Processes	Number of Erlang processes used by RabbitMQ in the node	0–1,048,576	Count	N/A	RabbitMQ instance node	1 minute
mem_used	Memory Usage	Memory usage of RabbitMQ in the node	0–32,000,000,000	Byte	1024(IEC)	RabbitMQ instance node	1 minute
disk_free	Available Memory	Available memory of RabbitMQ in the node	0–500,000,000,000	Byte	1024(IEC)	RabbitMQ instance node	1 minute
rabbitmq_alive	Node Alive	Whether the RabbitMQ node is alive This metric is supported for instances purchased in April 2020 or later.	1: alive 0: not alive	N/A	N/A	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
rabbitmq_disk_usage	Disk Capacity Usage	Disk usage of the RabbitMQ VM This metric is supported for instances purchased in April 2020 or later.	0~100	%	N/A	RabbitMQ instance node	1 minute
rabbitmq_cpu_usage	CPU Usage	CPU usage of the RabbitMQ VM This metric is supported for instances purchased in April 2020 or later.	0~100	%	N/A	RabbitMQ instance node	1 minute
rabbitmq_cpu_core_load	Average Load per CPU Core	Average load of each CPU core of the RabbitMQ VM This metric is supported for instances purchased in April 2020 or later.	> 0	N/A	N/A	RabbitMQ instance node	1 minute
rabbitmq_memory_usage	Memory usage of the RabbitMQ VM	Memory usage of the RabbitMQ VM This metric is supported for instances purchased in April 2020 or later.	0~100	%	N/A	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
rabbitmq_disk_read_await	Average Disk Read Time	Average time for each disk I/O read in the monitoring period This metric is supported for instances purchased in June 2020 or later.	> 0	ms	N/A	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 This metric is supported for instances purchased in June 2020 or later.	> 0	ms	N/A	RabbitMQ instance node	1 minute
rabbitmq_node_bytes_in_rate	Inbound Traffic	Inbound traffic per second This metric is supported for instances purchased in June 2020 or later.	> 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute
rabbitmq_node_bytes_out_rate	Outbound Traffic	Outbound traffic per second This metric is supported for instances purchased in June 2020 or later.	> 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
rabbitmq_node_queues	Queues	Number of queues in the node This metric is supported for instances purchased in June 2020 or later.	> 0	Count	N/A	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 This metric is supported for instances purchased in June 2020 or later.	1: yes 0: no	N/A	N/A	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 This metric is supported for instances purchased in June 2020 or later.	1: yes 0: no	N/A	N/A	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
rabbitmq_disk_read_rate	Disk Read Speed	Number of bytes read from the disk of the node each second This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute
rabbitmq_disk_write_rate	Disk Write Speed	Number of bytes written to the disk of the node each second This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute
connections_usage	Connection Usage	Percentage of current connections to the maximum number of connections This metric is supported for instances purchased since February 18, 2024.	≥ 0	%	N/A	RabbitMQ instance node	1 minute

Queue Metrics

Table 13-4 Queue metrics (RabbitMQ 3.x.x)

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_messages_unacknowledged	Unacknowledged Messages	Number of messages that have been consumed but not acknowledged in the RabbitMQ queue	0–10,000,000	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_ready	Queue Available Messages	Number of messages that can be retrieved in a RabbitMQ queue	0–10,000,000	Count	N/A	RabbitMQ instance queue	1 minute
queue_consumers	Consumers	Number of consumers that are subscribed to the queue This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_publish_rate	Production Rate	Number of messages sent to the queue each second This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_messages_ack_rate	Consumption Rate (Manual)	Number of acknowledged messages sent from the queue to clients each second This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute
queue_messages_deliver_get_rate	Consumption Rate	Number of messages sent from the queue each second This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute
queue_messages_redeliver_rate	Redelivery Rate	Number of messages in the queue redelivered each second This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_messages_persistent	Total Persisted Messages	Total number of persisted messages in the queue. This is always **0** for transient queues. This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_ram	Total Messages in RAM	Total number of messages in the queue that are kept in RAM This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_memory	Bytes Consumed by Erlang	Bytes of memory consumed by the Erlang process associated with the queue, including stack, heap, and internal structures This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Byte	1024(IEC)	RabbitMQ instance queue	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_message_bytes	Total Message Size	Total number of bytes of all messages in the queue This metric is supported for instances purchased on or after May 16, 2022.	≥ 0	Byte	1024(IEC)	RabbitMQ instance queue	1 minute

Table 13-5 Queue metrics (RabbitMQ AMQP-0-9-1)

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_messages_ready	Queue Available Messages	Number of messages that can be consumed in a RabbitMQ queue	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_consumers	Consumers	Number of consumers that are subscribed to the queue	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_publish_rate	Production Rate	Number of messages sent to the queue each second	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute
queue_messages_deliver_get_rate	Consumption Rate	Number of messages sent from the queue each second	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_bytes_in_rate	Message Production	Number of messages produced in real time in the queue per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance queue	1 minute
queue_bytes_out_rate	Message Consumption	Number of messages consumed in real time in the queue per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance queue	1 minute

Virtual Host Metrics

Available only for RabbitMQ AMQP-0-9-1.

Table 13-6 Virtual host metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
vhost_connections	Connections	Total number of connections in the virtual host	≥ 0	Count	N/A	RabbitMQ instance virtual host	1 minute
vhost_channels	Channels	Total number of channels in the virtual host	≥ 0	Count	N/A	RabbitMQ instance virtual host	1 minute
vhost_queues	Queues	Total number of queues in the virtual host	≥ 0	Count	N/A	RabbitMQ instance virtual host	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
vhost_consumers	Consumers	Total number of consumers in the virtual host	≥ 0	Count	N/A	RabbitMQ instance virtual host	1 minute
vhost_messages_ready	Vhost Available Messages	Total number of messages that can be consumed from the virtual host	≥ 0	Count	N/A	RabbitMQ instance virtual host	1 minute
vhost_messages_publish_rate	Production Rate	Number of messages produced in real time in the virtual host per second	≥ 0	Count/s	N/A	RabbitMQ instance virtual host	1 minute
vhost_messages_deliver_get_rate	Consumption Rate	Number of messages consumed in real time in the virtual host per second	≥ 0	Count/s	N/A	RabbitMQ instance virtual host	1 minute
vhost_bytes_in_rate	Message Creation	Number of bytes produced in the virtual host per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance virtual host	1 minute
vhost_bytes_out_rate	Message Retrieval	Number of bytes consumed from the virtual host per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance virtual host	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
vhost_tps	Vhost Transaction Per Second	Number of requests processed by the current virtual host per second	≥ 0	Count	N/A	RabbitMQ instance virtual host	1 minute

Exchange Metrics

Available only for RabbitMQ AMQP-0-9-1.

Table 13-7 Exchange metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
exchange_messages_publish_rate	Production Rate	Number of messages produced in real time in the exchange per second	≥ 0	Count/s	N/A	RabbitMQ instance exchange	1 minute
exchange_bytes_in_rate	Message Creation	Number of bytes produced in the exchange per second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance exchange	1 minute

Dimensions

Key	Value
rabbitmq_instance_id	RabbitMQ instance
rabbitmq_node	RabbitMQ instance node
rabbitmq_queue	RabbitMQ instance queue
rabbitmq_vhost	RabbitMQ instance virtual host

Key	Value
rabbitmq_vhost_exchange	RabbitMQ instance exchange
rabbitmq_vhost_queue	RabbitMQ instance queue

13.3 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 13-8 RabbitMQ instance metrics and alarm policies (RabbitMQ 3.x.x)

Metric	Alarm Policy	Description	Solution
Memory High Watermark	Alarm threshold: Raw data ≥ 1 Number of consecutive periods: 1 Alarm severity: Critical	A threshold of 1 indicates that the memory high watermark is reached, blocking message publishing.	• Accelerate message retrieval. • Use publisher confirms and monitor the publishing rate and duration on the publishing end. When the duration increases significantly, apply flow control.
Disk High Watermark	Alarm threshold: Raw data ≥ 1 Number of consecutive periods: 1 Alarm severity: Critical	A threshold of 1 indicates that the disk high watermark is reached, blocking message publishing.	• Reduce the number of messages accumulated in lazy queues. • Reduce the number of messages accumulated in durable queues. • Delete queues.

Metric	Alarm Policy	Description	Solution
Memory Usage	Alarm threshold: Raw data > Expected usage (30% is recommended) Number of consecutive periods: 3–5 Alarm severity: Major	To prevent high memory watermarks from blocking publishing, configure an alarm for this metric on each node.	• Accelerate message retrieval. • Use publisher confirms and monitor the publishing rate and duration on the publishing end. When the duration increases significantly, apply flow control.
CPU Usage	Alarm threshold: Raw data > Expected usage (70% is recommended) Number of consecutive periods: 3–5 Alarm severity: Major	A high CPU usage may slow down publishing rate. Configure an alarm for this metric on each node.	• Reduce the number of mirrored queues. • For a cluster instance, add nodes and rebalance queues between all nodes.
Available Messages	Alarm threshold: Raw data > Expected number of available messages Number of consecutive periods: 1 Alarm severity: Major	If the number of available messages is too large, messages are accumulated.	See **the solution to preventing message accumulation**.
Unacked Messages	Alarm threshold: Raw data > Expected number of unacknowledged messages Number of consecutive periods: 1 Alarm severity: Major	If the number of unacknowledged messages is too large, messages may be accumulated.	• Check whether the consumer is abnormal. • Check whether the consumer logic is time-consuming.

Metric	Alarm Policy	Description	Solution
Connections	Alarm threshold: Raw data > Expected number of connections Number of consecutive periods: 1 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: Raw data > Expected number of channels Number of consecutive periods: 1 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.
Erlang Processes	Alarm threshold: Raw data > Expected number of processes Number of consecutive periods: 1 Alarm severity: 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.

Table 13-9 RabbitMQ instance metrics and alarm policies (RabbitMQ AMQP-0-9-1)

Metric	Alarm Policy	Description	Solution
Available Messages	Alarm threshold: Raw data > Expected number of available messages Number of consecutive periods: 1 Alarm severity: Major	If the number of available messages is too large, messages are accumulated.	See **Solutions to Message Accumulation**

Metric	Alarm Policy	Description	Solution
Connections	Alarm threshold: Raw data > Expected number of connections Number of consecutive periods: 1 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: Raw data > Expected number of channels Number of consecutive periods: 1 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.
Instance Disk Usage	Alarm threshold: Raw data > 85% Number of consecutive periods: 1 Alarm severity: Critical	Large instance disk usage may be message accumulation.	See **Solutions to Message Accumulation**

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 **RabbitMQ console**.

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

Step 3 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** > **Monitoring Details**. On the displayed page, view

the metrics of the instance, nodes, and queues. Metric data is updated every minute.

Step 4 Hover the mouse pointer over a metric and click  to create an alarm rule for the metric.

Step 5 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

14 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 14-1 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 creating an order for creating an instance	rabbitmq	createDMSInstanceOrderSuccess
Failing to create an order for creating an instance	rabbitmq	createDMSInstanceOrderFailure
Successfully creating an order for modifying an instance	rabbitmq	modifyDMSInstanceOrderSuccess
Failing to create an order for modifying an instance	rabbitmq	modifyDMSInstanceOrderFailure
Successfully scaling up an instance	rabbitmq	extendDMSInstanceSuccess

Operation	Resource Type	Trace Name
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
Successfully deleting multiple instances at a time	rabbitmq	batchDeleteDMSInstanceSuccess
Failing to delete multiple instances at a time	rabbitmq	batchDeleteDMSInstanceFailure
Successfully modifying instance information	rabbitmq	modifyDMSInstanceInfoSuccess
Failing to modify instance information	rabbitmq	modifyDMSInstanceInfoFailure
Successfully deleting multiple instance tasks at a time	rabbitmq	batchDeleteDMSInstanceTask
Successfully unfreezing an instance	rabbitmq	unfreezeDMSInstanceTaskSuccess
Failing to unfreeze an instance	rabbitmq	unfreezeDMSInstanceTaskFailure
Successfully freezing an instance	rabbitmq	freezeDMSInstanceTaskSuccess
Failing to freeze an instance	rabbitmq	freezeDMSInstanceTaskFailure
Successfully deleting an instance task	rabbitmq	deleteDMSInstanceTaskSuccess
Failing to delete an instance task	rabbitmq	deleteDMSInstanceTaskFailure

Operation	Resource Type	Trace Name
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
Successfully submitting a request for modifying instance information	rabbitmq	modifyDMSInstanceInfoTaskSuccess
Failing to submit a request for modifying instance information	rabbitmq	modifyDMSInstanceInfoTaskFailure
Successfully restoring the instance from Recycle Bin	rabbitmq	out_recycleTaskSuccess
Failing to restore the instance from Recycle Bin	rabbitmq	out_recycleTaskFailure

Viewing Audit Logs

See **Querying Real-Time Traces**.

