Distributed Message Service for RabbitMQ

# User Guide

**Issue** 01

**Date** 2022-08-12

HUAWEI TECHNOLOGIES CO., LTD.

# Contents

# 1 Service Overview

## 1.1 What Is DMS for RabbitMQ?

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

- Immediate use

  DMS for RabbitMQ provides single-node and cluster instances with a range of specifications for you to choose from. Instances can be created with just a few clicks on the console, without requiring you to prepare servers.
- Rich features

  DMS for RabbitMQ supports Advanced Message Queuing Protocol (AMQP) and a variety of messaging features such as message broadcast, delayed delivery, and dead letter queues.
- Flexible routing

  In RabbitMQ, an exchange receives messages from producers and pushes the messages to queues. RabbitMQ provides direct, topic, headers, and fanout exchanges. You can also bind and customize exchanges.
- High availability

  In a RabbitMQ cluster, data is replicated to all nodes through mirrored queues, preventing service interruption and data loss in case of a node breakdown.
- Monitoring and alarm

  RabbitMQ cluster metrics are monitored and reported, including broker memory, CPU usage, and network flow. If an exception is detected, an alarm will be triggered.

## 1.2 Product Advantages

DMS for RabbitMQ provides easy-to-use message queuing based on RabbitMQ. Services can be quickly migrated to the cloud without any change, reducing maintenance and usage costs.

- Rapid deployment

  Simply set instance information on the DMS for RabbitMQ console, submit your order, and a complete RabbitMQ instance will be automatically created and deployed.

- Service migration without modifications

  DMS for RabbitMQ is compatible with open-source RabbitMQ APIs and supports all message processing functions of open-source RabbitMQ.

  If your application services are developed based on open-source RabbitMQ, you can easily migrate them to DMS for RabbitMQ after specifying a few authentication configurations.

  **NOTE**

  RabbitMQ instances are compatible with RabbitMQ 3.7.17.

- Exclusive experience

  RabbitMQ instances are physically isolated from each other and exclusively owned by the tenant.

- High performance

  Each queue can process up to 100,000 transactions per second (with default configurations). Performance can be increased simply by adding queues.

- Data security

  Operations on RabbitMQ instances are recorded and can be audited. Messages can be encrypted before storage.

  In addition to SASL, VPCs and security groups also provide security controls on network access.

- Simple O&M

  The cloud service platform provides a whole set of monitoring and alarm services, eliminating the need for 24/7 attendance. RabbitMQ instance metrics are monitored and reported, including the number of partitions, topics, and accumulated messages. You can configure alarm rules and receive SMS or email notifications on how your services are running in real time.

- Multi-language support

  RabbitMQ is an open-source service based on AMQP. It is used to store and forward messages in a distributed system. A RabbitMQ server is compiled in Erlang (supporting high concurrency, distribution, and robust fault tolerance), and a RabbitMQ client can be compiled in various programming languages, including Python, Ruby, .NET, Java, JMS, C, PHP, ActionScript, XMPP, STOMP, and AJAX.

## 1.3 Application Scenarios

RabbitMQ is popular message-oriented middleware that features highly reliable, asynchronous message delivery. It can be used for transmitting data between different systems in the enterprise application, payment, telecommunications, e-commerce, social networking, instant messaging, video, Internet of Things, and Internet of Vehicle industries.

## Asynchronous Communication

Non-core or less important messages are sent asynchronously to receiving systems, so that the main service process is not kept waiting for the results of other systems, allowing for faster responses.

For example, RabbitMQ can be used to send a notification email and SMS message after a user has registered with a website, providing fast responses throughout the registration process.

**Figure 1-1** Serial registration and notification

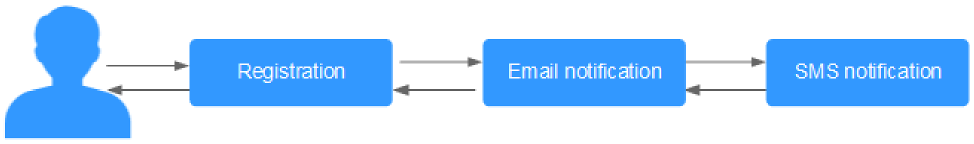
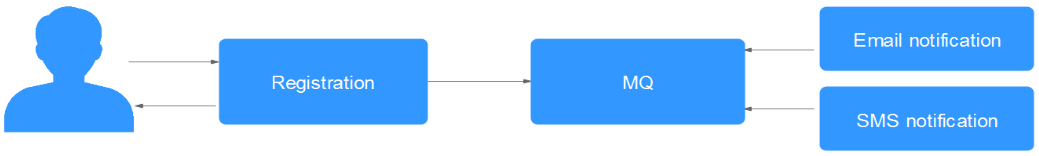


**Figure 1-2** Asynchronous registration and notification using message queues

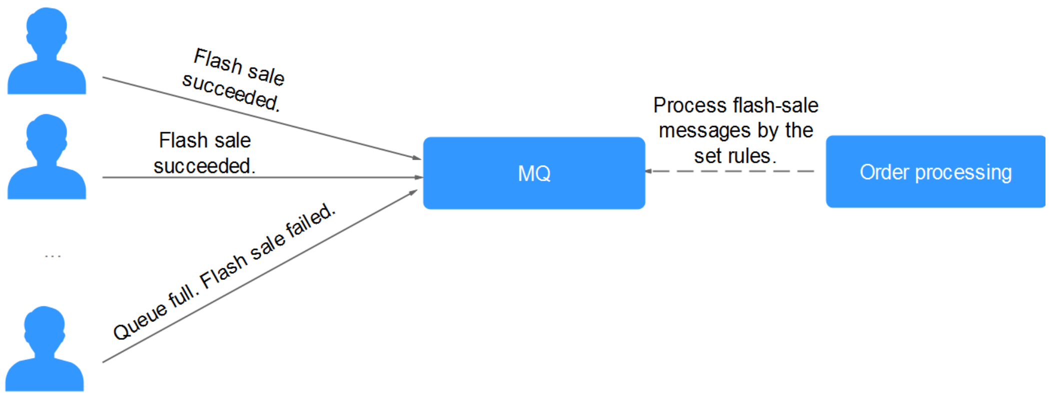


## Traffic Control

In e-commerce systems or large-scale websites, there is a processing capability gap between upstream and downstream systems. Traffic bursts from upstream systems with high processing capabilities may have a large impact on downstream systems with lower processing capabilities. For example, online sales promotions involve a huge amount of traffic flooding into e-commerce systems. RabbitMQ provides a three-day buffer by default for hundreds of millions of messages, such as orders and other information. In this way, message consumption systems can process the messages during off-peak periods.

In addition, flash sale traffic bursts originating from frontend systems can be handled with RabbitMQ, keeping the backend systems from crashing.

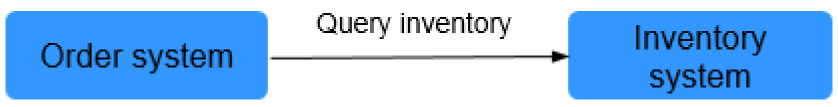
**Figure 1-3** Traffic burst handling using RabbitMQ


Flash sale succeeded.
Flash sale succeeded.
...
Queue full. Flash sale failed.
MQ
Process flash-sale messages by the set rules.
Order processing

## System Decoupling

Take e-commerce flash sales as an example. Traditionally, an order processing system sends order requests to the inventory system and waits for responses. If the inventory system goes down, the order processing system will not be able to get the data it wants, and the order will fail to be submitted. This means that the order processing system and the inventory system are closely coupled.

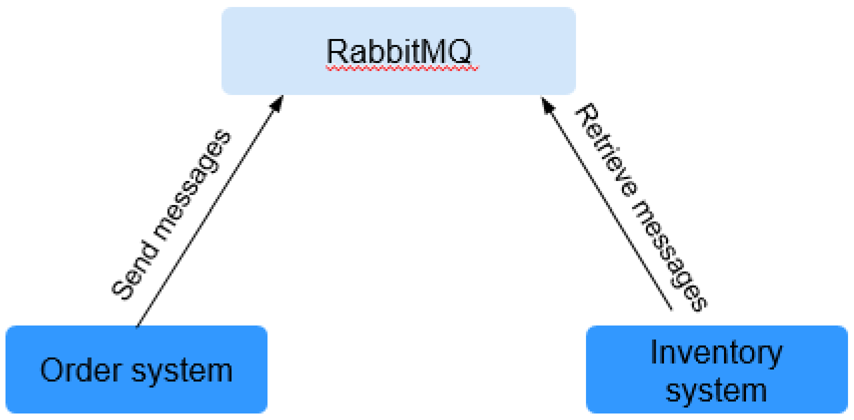
**Figure 1-4** Closely coupled systems



With RabbitMQ, order submission data will be stored in queues. Then, a response will be returned indicating that the order has been submitted.

The inventory system consumes the order submission message it has subscribed to. In this way, order submission will not be interrupted even if the inventory system breaks down.

**Figure 1-5** System decoupling



## High Availability

Queue mirroring is available since RabbitMQ 2.6.0. In a RabbitMQ cluster, queues can be mirrored across multiple brokers. In the event of a node failure, services are still available because the mirrors will take over.

In contrast, if there is only one broker, the queues will become unavailable if the broker fails.

Each mirrored queue consists of one master and multiple mirrors, which are distributed across the cluster brokers.

## 1.4 Specifications

### RabbitMQ Instance Specifications

DMS for RabbitMQ provides single-node and cluster instances compatible with RabbitMQ 3.7.17. Table 1-1 and Table 1-2 list the specifications of RabbitMQ instances.

NOTE

- To ensure stability, **the maximum size of a single message is 50 MB. Do not send a message larger than 50 MB**.
- In the following tables, the reference performance is represented by the number of messages (2 KB each) processed per second. In the tests, persistence and queue mirroring were not enabled. Messages were retrieved immediately after creation and were not accumulated in the queues. The data is for reference only and may differ from that in your production environment.
- Performance is related to the queue quantity, message accumulation, number of connections, number of channels, number of consumers, queue mirroring, priority queue, message persistence, and the exchange type. Select instance specifications based on the pressure test result of the service model.
- A maximum of 2047 channels can be opened on a connection.

**Table 1-1** Specifications of single-node RabbitMQ instances

| Instance Specifications | Reference Performance (TPS) | Queues (Recommended) | Maximum Connections |
|---|---|---|---|
| 2 vCPUs \| 4 GB | 10,000 | 100 | 2000 |
| 4 vCPUs \| 8 GB | 20,000 | 200 | 3000 |
| 8 vCPUs \| 16 GB | 35,000 | 400 | 5000 |
| 16 vCPUs \| 32 GB | 45,000 | 800 | 8000 |

**Table 1-2** Specifications of cluster RabbitMQ instances

| Instance Specifications | Brokers | Reference Performance (TPS) | Queues (Recommended) | Maximum Connections |
|---|---|---|---|---|
| 4 vCPUs \| 8 GB | 3 | 45,000 | 600 | 3000 x 3 |
| | 5 | 70,000 | 1000 | 3000 x 5 |
| | 7 | 80,000 | 1400 | 3000 x 7 |
| 8 vCPUs \| 16 GB | 3 | 85,000 | 1200 | 5000 x 3 |
| | 5 | 110,000 | 2000 | 5000 x 5 |

| Instance Specifications | Brokers | Reference Performance (TPS) | Queues (Recommended) | Maximum Connections |
|---|---|---|---|---|
| | 7 | 120,000 | 2800 | 5000 x 7 |
| 16 vCPUs \| 32 GB | 3 | 130,000 | 2400 | 8000 x 3 |
| | 5 | 160,000 | 4000 | 8000 x 5 |
| | 7 | 180,000 | 5600 | 8000 x 7 |

## Storage Space Selection

In cluster mode, RabbitMQ persists messages to disk. When creating a RabbitMQ instance, select a proper storage space size based on the estimated message size and the number of replicas in a mirrored queue, which can be maximally equal to the number of brokers in the cluster.

For example, if the estimated message size is 100 GB, the disk capacity must be at least: 100 GB x Number of mirrored replicas + 100 GB (reserved).

For single-node instances, select a storage space size based on the estimated message size and the reserved disk space.

You can change the number of brokers in a cluster, but cannot change the specifications of a single-node instance.

## 1.5 Comparing Kafka and RabbitMQ

Kafka is pull-based and provides higher throughput. It is suitable for collecting and delivering large volumes of data, such as collecting and analyzing logs. RabbitMQ does not provide as high throughput as Kafka, but it offers more message queuing functions.

The following is a comparison analysis on the performance, data reliability, service availability, and functions of Kafka and RabbitMQ.

### Performance

The performance of message-oriented middleware is measured by throughput. While RabbitMQ provides tens of thousands of QPS, Kafka provides millions.

However, if idempotency and transactions are enabled for Kafka, its performance will be compromised.

### Data Reliability

Both Kafka and RabbitMQ provide the replication mechanism to ensure high data reliability.

## Service Availability

Kafka runs in clusters and has partitions and replicas. Therefore, failure of a single broker does not affect services and the capacity of Kafka can be linearly scaled up.

## Functions

Both Kafka and RabbitMQ are popular open-source message-oriented middleware. They differ mainly in the functions, which are listed in the following table.

**Table 1-3** Function differences between Kafka and RabbitMQ

| Function | Kafka 1.1.0/2.3.0 | RabbitMQ 3.7.17 |
|---|---|---|
| Priority queue | Not supported | Supported. It is recommended that the priority be set to 0–10. |
| Delayed queue | Not supported | Supported |
| Dead letter queue | Not supported | Supported |
| Retry | Not supported | Not supported |
| Retrieval mode | Pull-based | Pull-based and push-based |
| Message broadcasting | Supported | Supported |
| Message tracking | Supports offset and timestamp tracking. | Not supported. Once a message retrieval has been acknowledged, RabbitMQ will be notified that the message can be deleted. |
| Message accumulation | Supports higher accumulation performance than RabbitMQ thanks to high throughput. | Supported |
| Persistence | Supported | Supported |
| Message tracing | Not supported | Supported by the firehose feature or the rabbitmq_tracing plugin. However, rabbitmq_tracing reduces performance and should be used only for troubleshooting. |
| Message filtering | Supported | Not supported, but can be encapsulated. |

| Function | Kafka 1.1.0/2.3.0 | RabbitMQ 3.7.17 |
|---|---|---|
| Multi-tenancy | Not supported | Supported |
| Multi-protocol | Only Apache Kafka is supported. | RabbitMQ is based on AMQP and supports MQTT and STOMP. |
| Multi-language | Kafka is written in Scala and Java and supports clients in multiple programming languages. | RabbitMQ is written in Erlang and supports clients in multiple programming languages. |
| Throttling | Supports throttling on producer or consumer clients and users. | Supports credit-based throttling on producers, a mechanism that triggers protection from within. |
| Ordered message delivery | Supports partition-level FIFO. | Supports FIFO only for single-threaded message queuing without advanced features such as delayed queues or priority queues. |
| Security | Supports SSL and SASL authentication and read/ write permissions control. | Similar to Kafka. |
| Idempotency | Supports idempotency for a single producer session. | Not supported |
| Transaction messages | Supported | Supported |

**NOTE**

The comparison is made between open-source Kafka and RabbitMQ.

DMS for Kafka and DMS for RabbitMQ maintain open-source compatibility while supporting or enhancing features in the open-source versions.

## 1.6 Related Services

- Cloud Trace Service (CTS)

  Cloud Trace Service (CTS) generates traces to provide you with a history of operations performed on cloud service resources. The traces include operation requests sent using the cloud service platform management console or open APIs as well as the operation results. You can view all generated traces to query, audit, and backtrack performed operations.
- VPC

  RabbitMQ premium instances run in VPCs and use the IP addresses and bandwidth of VPC. Security groups of VPCs enhance the security of network access to the RabbitMQ premium instances.

- Cloud Eye

  Cloud Eye is an open platform that provides monitoring, alarm reporting, and alarm notification for your resources in real time.

  NOTE

  The values of all RabbitMQ instance metrics are reported to Cloud Eye every minute.

## 1.7 Notes and Constraints

DMS for RabbitMQ has the following constraints, as listed in **Table 1-4**.

**Table 1-4** RabbitMQ usage constraints

| Item | Constraint | Description |
|---|---|---|
| Version | 3.7.17 | AMQP 0-9-1 clients are supported. |
| Number of connections | The allowed number of connections differs by instance specifications and mode (single-node or cluster). For details, see **Specifications**. | - |
| Channels | ≤ 2047 | Number of channels that can be created for a single connection. |
| Message size | ≤ 50 MB per message | Do not send a message larger than 50 MB. Otherwise, the message will fail to be created. |
| Memory high watermark | ≤ 40% | If the memory usage exceeds 40%, the high memory watermark is triggered, blocking publishers. |
| Disk high watermark | ≥ 5 GB | If the remaining disk space is less than 5 GB, the high disk watermark is triggered, blocking publishers. |

| Item | Constraint | Description |
|---|---|---|
| cluster_partition_handling | pause_minority | When a network partition occurs in a cluster, cluster brokers will determine whether they are in a minority, that is, fewer than or equal to the total number of brokers. Minority brokers pause when a partition starts, detect the network status periodically, and start again when the partition ends. If queue mirroring is not enabled, queue replicas in the minority will no longer be available for message creation and retrieval.<br>This strategy sacrifices availability for data consistency. |
| rabbitmq_delayed_message_exchange | There may be an error of about 1%. The actual delivery time may be earlier or later than the scheduled delivery time. | Whether delayed message delivery is enabled for the instance |

## 1.8 Basic Concepts

DMS for RabbitMQ uses RabbitMQ as the messaging engine. In RabbitMQ, messages are sent by producers, stored in queues, and received by consumers. The following explains basic concepts of RabbitMQ.

### Message

A message has a message body and a label. The message body, in JSON or other formats, contains the content of the message. The label only describes the message.

Messages are sent by producers and retrieved by consumers, but a producer and a consumer are not directly linked to each other.

### Producer

A producer is an application that sends messages to queues. The messages are then delivered to other systems or modules for processing as agreed.

### Consumer

A consumer is an application that receives messages. Consumers subscribe to queues. During routing, only the message body will be stored in the queue, so only the message body will be consumed by consumers.

### Queue

A queue stores messages that are sent from producers and await retrievals by consumers. If different consumers subscribe to the same queue, the messages in that queue will be distributed across the consumers.

### Broker

Nodes that provide message middleware services.

## 1.9 Permissions Management

You can use Identity and Access Management (IAM) to manage DMS for RabbitMQ permissions and control access to your resources. IAM provides identity authentication, permissions management, and access control.

You can create IAM users for your employees, and assign permissions to these users on a principle of least privilege (PoLP) basis to control their access to specific resource types. For example, you can create IAM users for software developers and assign specific permissions to allow them to use DMS for RabbitMQ resources but prevent them from being able to delete resources or perform any high-risk operations.

If your account does not require individual IAM users for permissions management, skip this section.

### DMS for RabbitMQ Permissions

By default, new IAM users do not have any permissions assigned. To assign permissions to these new users, add them to one or more groups, and attach permissions policies or roles to these groups.

DMS for RabbitMQ is a project-level service deployed and accessed in specific physical regions. When assigning DMS for RabbitMQ permissions to a user group, specify region-specific projects where the permissions will take effect. If you select **All projects**, the permissions will be granted for all region-specific projects. When accessing DMS for RabbitMQ, the users need to switch to a region where they have been authorized to use this service.

You can grant permissions by using roles and policies.

- Roles: A type of coarse-grained authorization mechanism that provides only a limited number of service-level roles. When using roles to grant permissions, you also need to assign dependency roles. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- Policies: A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based

authorization for securer access control. For example, you can grant DMS for RabbitMQ users only the permissions for managing instances. Most policies define permissions based on APIs. For the API actions supported by DMS for RabbitMQ, see **Permissions Policies and Supported Actions**.

NOTE

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

**Table 1-5** lists all the system-defined roles and policies supported by DMS for RabbitMQ.

**Table 1-5** System-defined roles and policies supported by DMS for RabbitMQ

| Role/Policy Name | Description | Type | Dependency |
|---|---|---|---|
| DMS FullAccess | Administrator permissions for DMS. Users granted these permissions can perform all operations on DMS. | System-defined policy | None |
| DMS UserAccess | Common user permissions for DMS, excluding permissions for creating, modifying, deleting, and scaling up instances. | System-defined policy | None |
| DMS ReadOnlyAccess | Read-only permissions for DMS. Users granted these permissions can only view DMS data. | System-defined policy | None |
| DMS Administrator | Administrator permissions for DMS. | System-defined role | This role depends on the **Tenant Guest** and **VPC Administrator** roles. |

**Table 2** lists the common operations supported by each DMS for RabbitMQ system policy or role. Select the policies or roles as required.

**Table 1-6** Common operations supported by each system policy

| Operation | DMS FullAccess | DMS UserAccess | DMS ReadOnlyAccess |
|---|---|---|---|
| Creating instances | √ | × | × |

| Operation | DMS FullAccess | DMS UserAccess | DMS ReadOnlyAccess |
|---|---|---|---|
| Modifying instances | √ | × | × |
| Deleting instances | √ | × | × |
| Modifying instance specifications | √ | × | × |
| Restarting instances | √ | √ | × |
| Querying instance information | √ | √ | √ |

# 2 Permissions Management

## 2.1 Creating a User and Granting DMS for RabbitMQ Permissions

This section describes how to use Identity and Access Management (IAM) for fine-grained permissions control for your Distributed Message Service (DMS) for RabbitMQ resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing DMS for RabbitMQ resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust another account or cloud service to perform efficient O&M on your DMS for RabbitMQ resources.

If your account does not require individual IAM users, skip this chapter.

This section describes the procedure for granting the **DMS ReadOnlyAccess** permission (see **Figure 2-1**) as an example.

**NOTE**

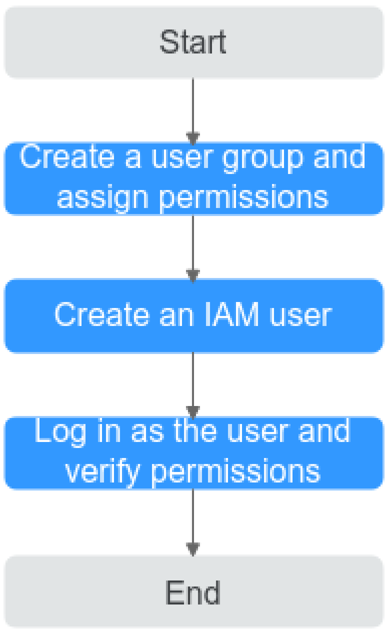
DMS for RabbitMQ permissions policies are based on DMS. Therefore, when assigning permissions, select DMS permissions policies.

### Prerequisites

Learn about the permissions (see *Service Overview* > *Permissions Management*) supported by DMS for RabbitMQ and choose policies according to your requirements.

### Process Flow

**Figure 2-1** Process of granting DMS for RabbitMQ permissions


Start
Create a user group and assign permissions
Create an IAM user
Log in as the user and verify permissions
End

1. Create a user group and assign permissions.

   Create a user group on the IAM console, and assign the **DMS ReadOnlyAccess** policy to the group.

2. Create an IAM user.

   Create a user on the IAM console and add the user to the group created in **1**.

3. Log in and verify permissions.

   Log in to the console using the user created in **2**, and verify that the user only has read permissions for DMS for RabbitMQ.

## 2.2 DMS for RabbitMQ Custom Policies

Custom policies can be created to supplement the system-defined policies of DMS for RabbitMQ. For the actions that can be added for custom policies, see **Permissions Policies and Supported Actions**.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For operation details, see "Creating a Custom Policy" in the *Identity and Access Management User Guide*. The following section contains examples of common DMS for RabbitMQ custom policies.

**NOTE**

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

## Example Custom Policies

- Example 1: Allowing users to delete and restart instances

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

- Example 2: Denying instance deletion

  A policy with only "Deny" permissions must be used in conjunction with other policies to take effect. If the permissions assigned to a 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"
                        ]
                }
        ]
}
```

# 3 Preparing the Environment

Before creating RabbitMQ instances, you must create a VPC and configure security groups and subnets for it. A VPC creates an isolated virtual network environment for you to configure and manage RabbitMQ instances, improving resource security and simplifying network deployment.

Once you have created a VPC, you can use it for all instances you subsequently create.

## Creating a VPC

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

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

NOTE

Select the same region as your application service.

**Step 3** Click and choose **Networking** > **Virtual Private Cloud**.

**Step 4** Click **Create VPC**.

**Step 5** Create a VPC as prompted, retaining the default values unless otherwise required. For details on how to create a VPC, see the *Virtual Private Cloud User Guide*.

After a VPC is created, a subnet is also created. If the VPC needs more subnets, go to **Step 6**. Otherwise, go to **Step 7**.

**Step 6** In the navigation pane, choose **Subnets**. Click **Create Subnet**. Create a subnet as prompted, retaining the default values unless otherwise required.

For details on how to create a subnet, see the *Virtual Private Cloud User Guide*.

**Step 7** In the navigation pane, choose **Access Control** > **Security Groups**. Create a security group as prompted, retaining the default values unless otherwise required.

For details on how to create a security group, see the *Virtual Private Cloud User Guide*.

**----End**

# 4 Buying an Instance

## Scenario

DMS provides RabbitMQ instances that are physically isolated for each tenant. You can customize the computing capabilities and storage space of a RabbitMQ instance based on service requirements.

RabbitMQ is an open-source service based on the Advanced Message Queuing Protocol (AMQP). It is used to store and forward messages in a distributed system. A RabbitMQ server is compiled in Erlang (supporting high concurrency, distributed deployment, and robust fault tolerance), and a RabbitMQ client can be compiled in various programming languages, including Python, Ruby, .NET, Java, JMS, C, PHP, ActionScript, XMPP, STOMP, and AJAX.

AMQP is an open standard application-layer protocol for message-oriented middleware. Its key features are message orientation, queuing, routing, reliability, and security.

## Prerequisites

A VPC configured with security groups and subnets is available.

## Procedure

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

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

**NOTE**

Select the same region as your application service.

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

**Step 4** Click **Buy Instance** in the upper right corner of the page.

**Step 5** Specify **Billing Mode**, **Region**, **Project**, and **AZ**.

**Step 6** Specify the instance name and the enterprise project.

**Step 7** Configure the following instance parameters.

1. **Version**: RabbitMQ version. Currently, only 3.7.17 is supported.
2. **Instance Type**: Select **Single-node** or **Cluster**.
   - **Single-node**: There is only one RabbitMQ broker.
   - **Cluster**: There are multiple RabbitMQ broker, achieving highly reliable message storage.
3. **CPU Architecture**: Currently, only x86 architecture is supported.
4. **Specifications**: Select specifications as required.

   NOTE

   To ensure service stability and reliability, DMS for RabbitMQ sets the default memory high watermark to 40%. Publishers will be blocked if the memory usage exceeds 40%. To avoid reaching the high watermark, retrieve messages stacked in the queue in time.
5. **Storage Space**: Indicates the disk type and total storage space of the RabbitMQ instance.
   - For a single-node instance, the value range is 200 GB to 90,000 GB.
   - For a cluster instance, the value range can be 100 GB x Number of brokers to 90,000 GB, 200 GB x Number of brokers to 90,000 GB, or 300 GB x Number of brokers to 90,000 GB.
6. **VPC**: Select a VPC and a subnet.

   A VPC provides an isolated virtual network for your RabbitMQ instances. You can configure and manage the network as required.
7. **Security Group**: Select a security group.

   A security group is a set of rules that control access to ECSs. It provides access policies for mutually trusted ECSs with the same security protection requirements in the same VPC.

   Click **Manage Security Group**. On the console that is displayed, view or create security groups.

**Step 8** Enter the username and password used for connecting to the RabbitMQ instance.

**Step 9** Click **More Settings** to configure more parameters.

1. Configure **Public Access**.

   Public access can be enabled or disabled.

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

   NOTE

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

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

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

3. Specify tags.

   Tags are used to identify cloud resources. When you have many cloud resources of the same type, you can use tags to classify cloud resources by dimension (for example, usage, owner, or environment).

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

   Up to 20 tags can be added to each RabbitMQ instance. For details about the requirements on tags, see **Managing Instance Tags**.

4. Enter a description of the instance.

**Step 10** Click **Buy Now**.

**Step 11** Confirm the instance information and 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 fails to be created, view **Instance Creation Failures**. Delete the instance by referring to **Deleting an Instance** and create another instance. If the RabbitMQ instance creation fails a second time, contact customer service.

**----End**

# 5 Accessing a RabbitMQ Instance

## 5.1 Accessing a RabbitMQ Instance Without SSL Encryption

RabbitMQ instances are compatible with the open-source RabbitMQ protocol. To access and use a RabbitMQ instance in different languages, see the tutorials at **https://www.rabbitmq.com/getstarted.html**.

The following demo shows how to access and use a RabbitMQ instance in a VPC, assuming that the RabbitMQ client is deployed in an ECS.

If SSL is enabled for the instance, see **Accessing a RabbitMQ Instance with SSL Encryption** for how to access the instance.

### Prerequisites

- A RabbitMQ instance has been created following the instructions in **Buying an Instance**, and the username and password used to create the instance have been obtained.
- The **Connection Address** of the instance has been recorded from the instance details.
- An ECS has been created, and its VPC, subnet, and security group configurations are the same as those of the RabbitMQ instance.

### Accessing the Instance in CLI Mode

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

**Step 2** Install JDK or JRE, and add the following lines to **.bash_profile** in the home directory to configure the environment variables **JAVA_HOME** and **PATH**:

```
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.

NOTE

Use Oracle JDK instead of ECS's default JDK (for example, OpenJDK), because ECS's default JDK may not be suitable for the sample project. Obtain Oracle JDK 1.8.111 or later from **Oracle's official website**.

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

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

**Step 4** Run the following command to decompress **RabbitMQ-Tutorial.zip**:

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

**Step 5** Run the following command to navigate to the **RabbitMQ-Tutorial** directory, which contains the precompiled JAR file:

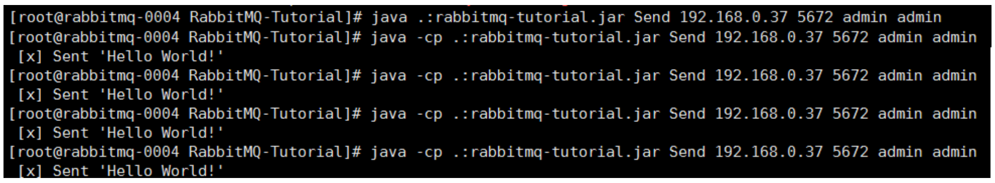
```
$ cd RabbitMQ-Tutorial
```

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

```
$ java -cp .:rabbitmq-tutorial.jar Send host port user password
```

*host* indicates the connection address for accessing the instance. *port* is the listening port of the instance, which is **5672** by default. *user* and *password* indicate the username and password used for accessing the instance.

**Figure 5-1** Sample project for creating messages



```
[root@rabbitmq-0004 RabbitMQ-Tutorial]# java .:rabbitmq-tutorial.jar Send 192.168.0.37 5672 admin admin
[root@rabbitmq-0004 RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.0.37 5672 admin admin
 [x] Sent 'Hello World!'
[root@rabbitmq-0004 RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.0.37 5672 admin admin
 [x] Sent 'Hello World!'
[root@rabbitmq-0004 RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.0.37 5672 admin admin
 [x] Sent 'Hello World!'
[root@rabbitmq-0004 RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.0.37 5672 admin admin
 [x] Sent 'Hello World!'
```

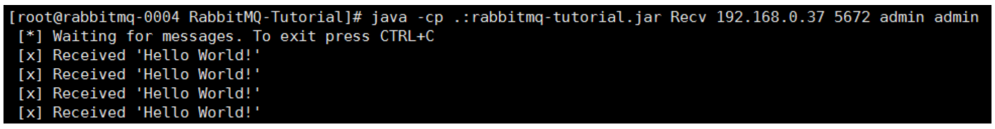
Press **Ctrl**+**C** to exit.

**Step 7** Retrieve messages using the sample project.

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

*host* indicates the connection address for accessing the instance. *port* is the listening port of the instance, which is **5672** by default. *user* and *password* indicate the username and password used for accessing the instance.

**Figure 5-2** Sample project for retrieving messages



```
[root@rabbitmq-0004 RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Recv 192.168.0.37 5672 admin admin
 [*] Waiting for messages. To exit press CTRL+C
 [x] Received 'Hello World!'
 [x] Received 'Hello World!'
 [x] Received 'Hello World!'
 [x] Received 'Hello World!'
```

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

**----End**

## Java Sample Code

Accessing a RabbitMQ instance and creating messages

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

 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();
```

Accessing a RabbitMQ instance and retrieving messages

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

## 5.2 Accessing a RabbitMQ Instance with SSL Encryption

If SSL is enabled, data will be encrypted before transmission for enhanced security.

This section describes intra-VPC access to a RabbitMQ instance with SSL enabled.

### Prerequisites

- A RabbitMQ instance has been created following the instructions in **Buying an Instance**, and the username and password used to create the instance have been obtained.
- The **Connection Address** of the instance has been recorded from the instance details.
- An ECS has been created, and its VPC, subnet, and security group configurations are the same as those of the RabbitMQ instance.

### Accessing the Instance in CLI Mode

**Step 1** Log in to the ECS. If public network access is enabled, log in to the server for running commands.

**Step 2** Install JDK or JRE, and add the following lines to **.bash_profile** in the home directory to configure the environment variables **JAVA_HOME** and **PATH**:

```
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.

**NOTE**

Use Oracle JDK instead of ECS's default JDK (for example, OpenJDK), because ECS's default JDK may not be suitable for the sample project. Obtain Oracle JDK 1.8.111 or later from **Oracle's official website**.

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

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

**Step 4** Run the following command to decompress **RabbitMQ-Tutorial-SSL.zip**:

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

**Step 5** Run the following command to navigate to the **RabbitMQ-Tutorial-SSL** directory, which contains the precompiled JAR file:

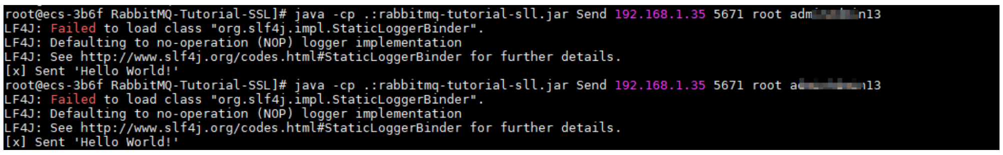
```
$ cd RabbitMQ-Tutorial-SSL
```

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

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

*host* indicates the connection address for accessing the instance. *port* is the listening port of the instance, which is **5671** by default. *user* and *password* indicate the username and password used for accessing the instance.

**Figure 5-3** Sample project for message creation



```
root@ecs-3b6f RabbitMQ-Tutorial-SSL]# java -cp .:rabbitmq-tutorial-sll.jar Send 192.168.1.35 5671 root adm[redacted]n13
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[x] Sent 'Hello World!'
root@ecs-3b6f RabbitMQ-Tutorial-SSL]# java -cp .:rabbitmq-tutorial-sll.jar Send 192.168.1.35 5671 root a[redacted]n13
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[x] Sent 'Hello World!'
```

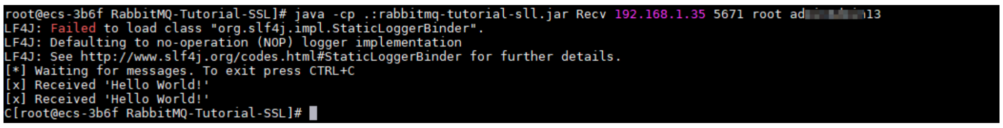
Press **Ctrl**+**C** to exit.

**Step 7** Retrieve messages using the sample project.

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

*host* indicates the connection address for accessing the instance. *port* is the listening port of the instance, which is **5671** by default. *user* and *password* indicate the username and password used for accessing the instance.

**Figure 5-4** Sample project for message retrieval



```
root@ecs-3b6f RabbitMQ-Tutorial-SSL]# java -cp .:rabbitmq-tutorial-sll.jar Recv 192.168.1.35 5671 root ad[redacted]13
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Hello World!'
[x] Received 'Hello World!'
C[root@ecs-3b6f RabbitMQ-Tutorial-SSL]#
```

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

**----End**

## Java Sample Code

Accessing a RabbitMQ instance and creating messages

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

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

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

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

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

Accessing a RabbitMQ instance and retrieving messages

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

## 5.3 Connecting to the Management Address of a RabbitMQ Instance

Access the management address of a RabbitMQ instance by using the browser-based, open-source RabbitMQ cluster management tool.

### Procedure

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

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

NOTE

Select the same region as your application service.

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

   NOTE

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

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

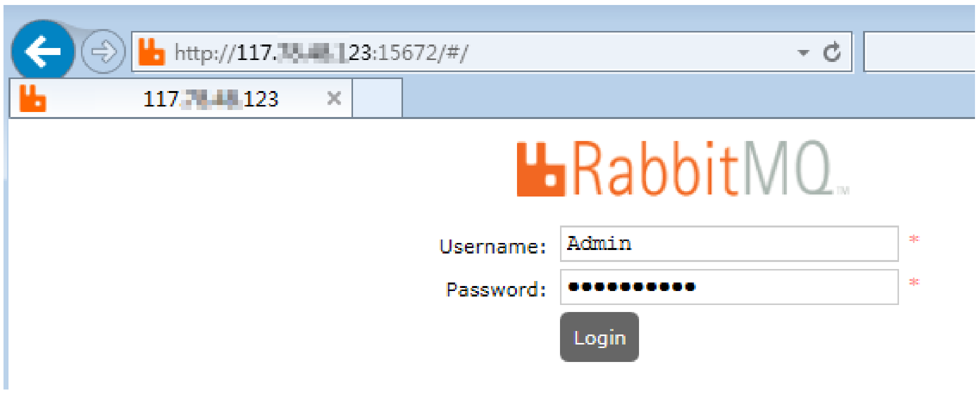
1. In the **Network** section on the **Basic Information** tab page, click the name of the security group.
2. Click the **Inbound Rules** tab to view the inbound rules of the security group.
   - SSL disabled
     - For intra-VPC access, inbound access through port 5672 must be allowed.
     - For public access, inbound access through port 15672 must be allowed.
   - SSL enabled
     - For intra-VPC access, inbound access through port 5671 must be allowed.
     - For public access, inbound access through port 15671 must be allowed.

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

NOTE

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

**Figure 5-5** Logging in to the management UI



http://117.[redacted]23:15672/#/

117.[redacted].123

RabbitMQ

Username: Admin

Password: ●●●●●●●●●●●

**Step 4** Click **Login**.

**----End**

## 5.4 Enabling Heartbeats

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

### What Is a Heartbeat?

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

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

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

After the heartbeat timeout is configured, the RabbitMQ server and client send AMQP heartbeat frames to each other at an interval of half the heartbeat timeout. After a client misses two heartbeats, it is considered unreachable and the TCP connection is closed. If the client detects that the server cannot be accessed due to heartbeats, the client needs to reconnect to the server.

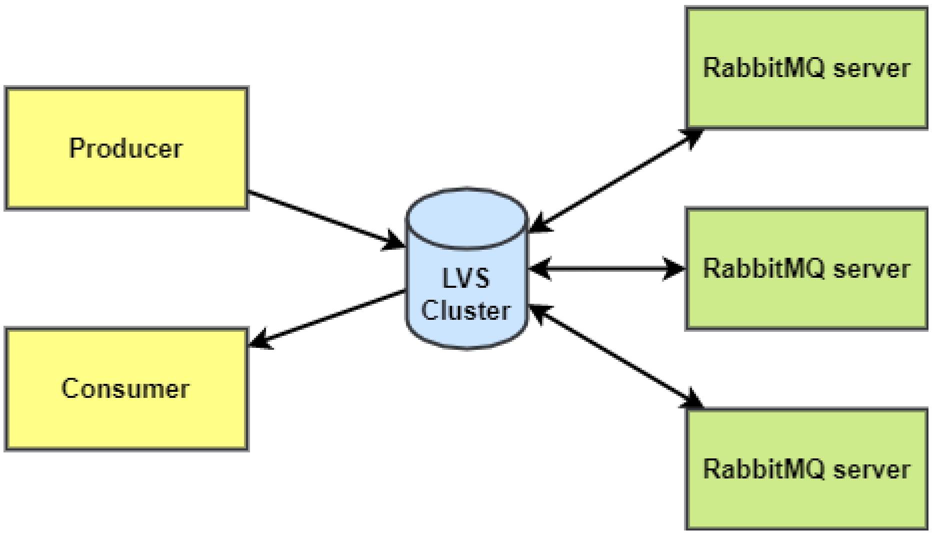
NOTICE

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 5-6**. Single-node instances do not have LVSs.

**Figure 5-6** Load balancing of a cluster instance



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

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

## Configuring Heartbeat Timeout on the Client

- Java client

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

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

- .NET client

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

- Python Pika

  ```
  // The heartbeat timeout is 60 seconds.
  params = pika.ConnectionParameters(host='host', heartbeat=60,
  ```

```
credentials=pika.PlainCredentials('username', 'passwd'))
connection = pika.BlockingConnection(params)

while True:
    channel.basic_publish(exchange='', routing_key='hello', body='Hello World!')
    print(" [x] Sent 'Hello World!'")
    # The producer needs to use connection.sleep() to trigger a heartbeat. time.sleep() cannot trigger
heartbeats.
    connection.sleep(200)
```

## 5.5 Viewing Client Connection Addresses

View client connection addresses on the RabbitMQ management UI.

NOTE

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

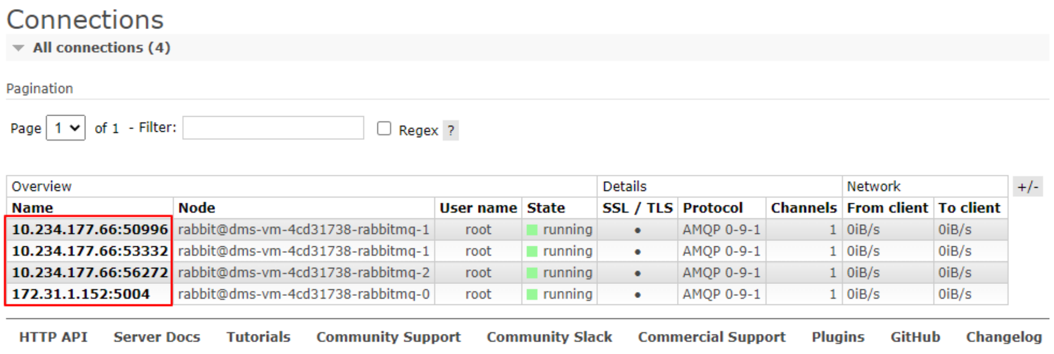
### Procedure

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

**Step 2** In the navigation pane, choose **Connections**.

**Step 3** View client connection addresses, as shown in **Figure 5-7**.

**Figure 5-7** Client connection addresses



Connections

All connections (4)

Pagination

Page 1 of 1 - Filter: Regex ?

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

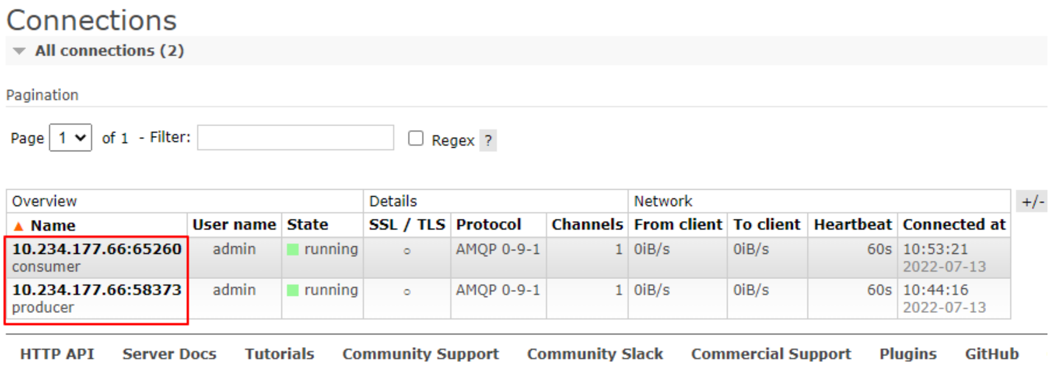
A client can function as a producer to create messages and as a consumer to retrieve messages. The producer and consumer IP addresses are the same, as shown in **Figure 5-7**, and are difficult to distinguish. To differentiate between producer and consumer IP addresses, you can set the **clientProperties** 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 5-8**.

**Figure 5-8** Client connection addresses (with producer/consumer differentiated)



Connections

All connections (2)

Pagination

Page 1 of 1 - Filter: Regex ?

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

**----End**

# 6 Operating RabbitMQ Instances

## 6.1 Viewing an Instance

### Scenario

View detailed information about a RabbitMQ instance on the console, for example, the connection address and port number for accessing the instance.

### Prerequisites

A RabbitMQ instance has been created.

### Procedure

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

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

NOTE

Select the same region as your application service.

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

**Step 4** Search for a RabbitMQ instance by specifying the project, status, name, connection address, or ID. **Table 6-1** describes the various possible statuses of a RabbitMQ instance.

**Table 6-1** RabbitMQ instance status description

| Status | Description |
|---|---|
| Creating | The instance is being created. |

| Status | Description |
|---|---|
| Running | The instance is running properly.<br>Only instances in the **Running** state can provide services. |
| Faulty | The instance is not running properly. |
| Starting | The status between **Frozen** and **Running**. |
| Restarting | The instance is being restarted. |
| 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. |

**Step 5** Click the name of the chosen RabbitMQ instance and view the instance details on the page that is displayed.

**Table 6-2** Instance parameters

| Section | Parameter | Description |
|---|---|---|
| Instance Information | Instance Name | Name of the RabbitMQ instance. To modify the instance name, click ✎. |
| | Status | Status of the RabbitMQ instance. |
| | Instance Type | Type of the RabbitMQ instance, which can be **Single-node** or **Cluster**. |
| | Specifications | Specifications of the RabbitMQ instance. |
| | Mgmt. UI Address | Address for connecting to the RabbitMQ management UI when public access is disabled. |
| | Time Window | Time range for any scheduled maintenance activities on the instance. To modify the time window, click ✎. |
| | Enterprise Project | Enterprise project to which the instance belongs. To modify the enterprise project, click ✎. |

| Section | Parameter | Description |
|---|---|---|
| | Instance ID | ID of the RabbitMQ instance. |
| | Version | RabbitMQ version used by the instance. Currently, only version 3.7.17 is supported. |
| | Created | Time when the RabbitMQ instance is created. |
| | Username | Username for logging in to the RabbitMQ cluster management tool. If you forget the password, click **Reset Password** to change the password. |
| | Description | Description of the RabbitMQ instance. To modify the description, click ✎. |
| Storage Informatio n | Used/ Available Storage Space (GB) | Storage space that has been used by the RabbitMQ instance and the maximum storage space that can be used. |
| | Disk Type | Type of the disk used by the instance. Currently, high I/O and ultra-high I/O are supported. |
| Public Access | Public Access | Whether public access has been enabled. Click ✎ to change the public network setting. |
| | Public Access Address | Address for accessing the RabbitMQ instance over public networks. |
| | Mgmt. UI Address | Address for connecting to the RabbitMQ management UI over public networks. |
| Network | AZ | AZ to which the RabbitMQ instance belongs. |
| | Security Group | Security group that controls access to the RabbitMQ instance. |
| | VPC | VPC in which the instance resides. |
| | Subnet | Subnet in which the instance resides. |

**----End**

## 6.2 Restarting a RabbitMQ Instance

### Scenario

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

NOTICE

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

## Prerequisites

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

## Procedure

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

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

NOTE

Select the same region as your application service.

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

**Step 4** Restart RabbitMQ instances using one of the following methods:

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

**Step 5** Click **Yes**.

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

NOTE

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

To restart a single RabbitMQ instance, you can also click **Restart** in the row containing the chosen RabbitMQ instance on the **RabbitMQ Premium** page.

**----End**

## 6.3 Deleting an Instance

### Scenario

With a few clicks on the DMS (for RabbitMQ) console, you can delete one or more RabbitMQ instances that have been created or multiple RabbitMQ instances that failed to be created.

NOTICE

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

## Prerequisites

The status of the RabbitMQ instance you want to delete is **Running** or **Faulty**.

## Deleting a RabbitMQ Instance

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

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

NOTE

Select the same region as your application service.

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

**Step 4** Delete RabbitMQ instances using one of the following methods:

- 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**.

NOTE

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

**Step 5** Click **Yes**.

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

**----End**

## Deleting a RabbitMQ Instance That Failed to Be Created

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

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

NOTE

Select the same region as your application service.

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

If there are RabbitMQ instances that failed to be created, **Instance Creation Failures** and quantity information will be displayed.

**Step 4** Click the icon or quantity next to **Instance Creation Failures**.

**Step 5** Delete RabbitMQ instances that failed to be created in either of the following ways:

- To batch delete all RabbitMQ instances that failed to be created, click **Clear Failed Instance**.
- To delete a single RabbitMQ instance that failed to be created, click **Delete** in the row containing the chosen RabbitMQ instance.

**----End**

## 6.4 Modifying the Instance Information

After creating a RabbitMQ instance, you can adjust some parameters of the instance based on your service requirements.

### Procedure

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

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

NOTE

Select the same region as your application service.

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

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

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

- Instance Name
- Time Window
- Description
- Enterprise Project
- Security Group
- Public Access (For details about how to change the public access configuration, see **Configuring Public Access**.)

After the parameters are modified, view the modification result in the following ways:

- If **Public Access** has been modified, you will be redirected to the **Background Tasks** page, which displays the modification progress and result.
- If **Instance Name**, **Time Window**, **Description**, **Enterprise Project**, or **Security Group** has been modified, the modification result will be displayed in the upper right corner of the page.

**----End**

## 6.5 Resetting the Instance Password

### Scenario

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

**NOTE**

You can reset the password of a RabbitMQ instance only if the instance in the **Running** state.

### Procedure

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

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

**NOTE**

Select the same region as your application service.

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

**Step 4** In the row containing the desired instance, choose **More** > **Reset Password**.

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

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

**NOTE**

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

**----End**

## 6.6 Modifying Instance Specifications

### Scenario

After creating a RabbitMQ instance in cluster mode, you can modify the number of cluster brokers, but cannot modify the specifications themselves. This operation is not supported by RabbitMQ instances in single-node mode.

NOTICE

- To ensure that the instance runs properly, do not perform other operations on the instance during the modification.
- During the specification modification, the LVS node is restarted and the client is disconnected. The service code must be able to handle this situation to prevent service interruption.

## Prerequisites

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

## Procedure

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

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

NOTE

Select the same region as your application service.

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

**Step 4** In the row containing the instance for which you want to modify the specifications, choose **More** > **Modify Specifications** in the **Operation** column.

**Step 5** Specify the desired number of cluster nodes and click **Submit**.

View the number of cluster nodes on the **RabbitMQ Premium** page.

**----End**

## 6.7 Configuring Public Access

### Scenario

Enable public access to a RabbitMQ instance so that you can access the instance over a public network. If you no longer need public access to the instance, you can disable it as required.

NOTICE

- You can enable public access to a RabbitMQ instance only if its status is **Running**.
- 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.
- After enabling public access, configure the following settings:
  - If SSL has been disabled:

    Add an inbound rule to the security group, allowing access to ports 5672 and 15672.

    To access the RabbitMQ management plane, visit http://*{public IP address of the RabbitMQ instance}*:15672, and enter your username and password.

    To access the instance through clients, use port 5672.
  - If SSL has been enabled:

    Add an inbound rule to the security group, allowing access to ports 5671 and 15671.

    To access the RabbitMQ management plane, visit https://*{public IP address of the RabbitMQ instance}*:15671, and enter your username and password.

    To access the instance through clients, use port 5671.

## Prerequisites

A RabbitMQ instance has been created.

## Enabling Public Access

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

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

NOTE

Select the same region as your application service.

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

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

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

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

If no EIP exists in the **Elastic IP Address** drop-down list box, click **Create Elastic IP** to create an EIP on the page that is displayed.

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

**----End**

## Disabling Public Access

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

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

NOTE

Select the same region as your application service.

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

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

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

**Step 6** Click ✓.

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

**----End**

## 6.8 Configuring Queue Mirroring

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

To learn more about the RabbitMQ web UI, visit the **RabbitMQ official website**. The following procedure describes how to configure queue mirroring on the RabbitMQ web UI.

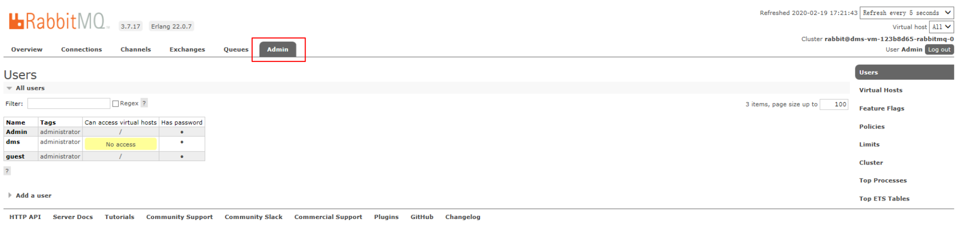
### Procedure

**Step 1** Log in to the management UI of a RabbitMQ instance.

For details on how to log in to the management UI, see **Connecting to the Management Address of a RabbitMQ Instance**.

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

**Figure 6-1** Admin tab page



Refreshed 2020-02-19 17:21:43 Refresh every 5 seconds
Virtual host All
Cluster rabbit@dms-vm-123b8d65-rabbitmq-0
User Admin

Users

All users

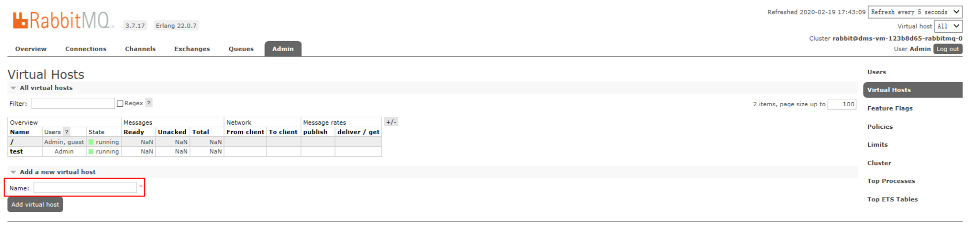
Filter: ☐ Regex ?　　3 items, page size up to 100

| Name | Tags | Can access virtual hosts | Has password |
|---|---|---|---|
| Admin | administrator | / | ● |
| dms | administrator | No access | ● |
| guest | administrator | / | ● |

**Step 3** (Optional) In the navigation tree on the right, choose **Virtual Hosts**, specify **Name**, and click **Add virtual host** to create a virtual host.

Perform this step only if you need to specify a virtual host. Otherwise, go to **Step 4**.

**Figure 6-2** Creating a virtual host



Refreshed 2020-02-19 17:43:09 Refresh every 5 seconds
Virtual host All
Cluster rabbit@dms-vm-123b8d65-rabbitmq-0
User Admin

Virtual Hosts

All virtual hosts

Filter: ☐ Regex ?　　2 items, page size up to 100

| Overview | | | Messages | | | Network | | Message rates | |
|---|---|---|---|---|---|---|---|---|---|
| Name | Users ? | State | Ready | Unacked | Total | From client | To client | publish | deliver / get |
| / | Admin, guest | 🟩 running | NaN | NaN | NaN | | | | |
| test | Admin | 🟩 running | NaN | NaN | NaN | | | | |

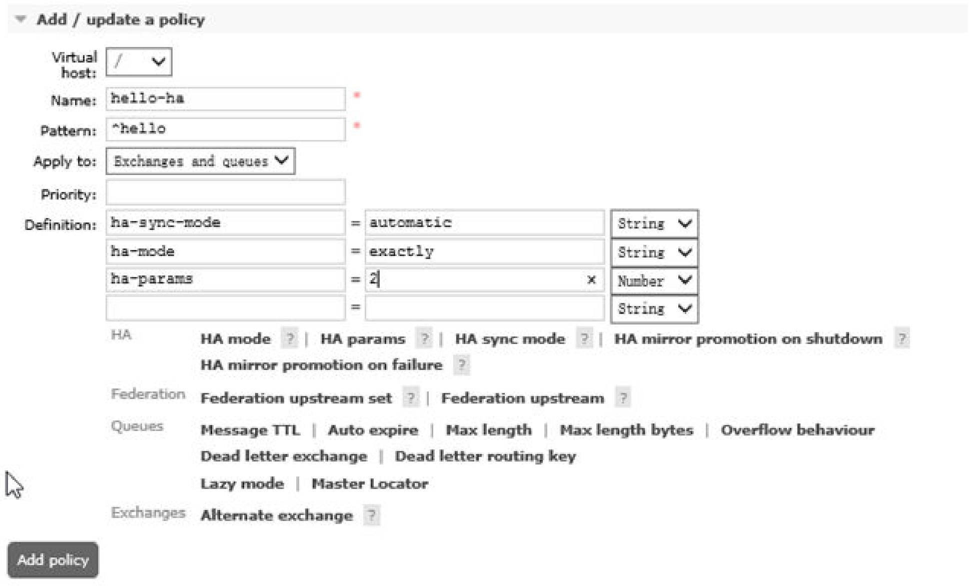
Add a new virtual host

Name: *

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

To set rules for a specific virtual host, select the virtual host created in **Step 3** from the **Virtual host** drop-down list box. If no virtual host has been created, the default value **/** is used.

**Figure 6-3** Setting rules for a virtual host


HA
HA mode ? | HA params ? | HA sync mode ? | HA mirror promotion on shutdown ?
HA mirror promotion on failure ?
Federation
Federation upstream set ? | Federation upstream ?
Queues
Message TTL | Auto expire | Max length | Max length bytes | Overflow behaviour
Dead letter exchange | Dead letter routing key
Lazy mode | Master Locator
Exchanges
Alternate exchange ?
Add policy

Add / update a policy

Virtual host: /
Name: hello-ha *
Pattern: ^hello *
Apply to: Exchanges and queues
Priority:
Definition:

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

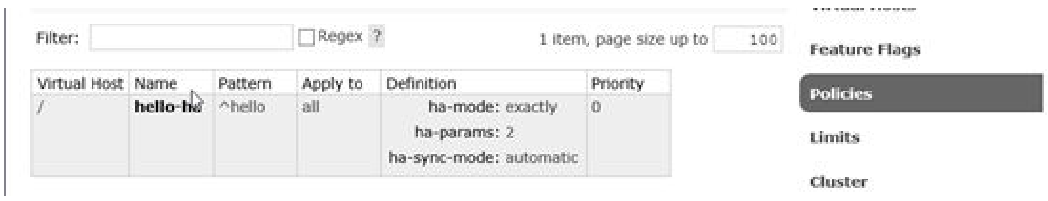
Parameter description:

- **Name**: customized name of the policy.
- **Pattern**: regular expression that defines the pattern for matching queues.
- **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 **manually**.
    - **automatic**: Data is automatically synchronized from the master.
    - **manually**: Data is manually synchronized from the master.
  - **ha-mode**: queue mirroring mode. Options: **all**, **exactly**, and **nodes**.
    - **all**: Mirror the queue across all brokers in the cluster.
    - **exactly**: Mirror the queue to a specific number (determined through **ha-params**) of brokers in the cluster.
    - **nodes**: Mirror the queue to specific brokers (determined through **ha-params**).
  - **ha-params**: This parameter is used for specifying the nodes in the **ha-mode** parameter.
- (Optional) **Priority**: priority of the policy.

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

The following figure shows a successfully added policy.

**Figure 6-4** Virtual host policy



| Virtual Host | Name | Pattern | Apply to | Definition | Priority |
|---|---|---|---|---|---|
| / | hello-ha | ^hello | all | ha-mode: exactly<br>ha-params: 2<br>ha-sync-mode: automatic | 0 |

**----End**

## 6.9 Enabling Plug-ins

After creating a RabbitMQ instance, you can enable the plug-ins listed in the following table. The plug-ins are disabled by default when the instance is created.

**NOTE**

When plug-ins are enabled, the instance will not be restarted. However, enabling plug-ins rabbitmq_mqtt, rabbitmq_web_mqtt, rabbitmq_stomp, and rabbitmq_web_stomp will restart Keepalived and disconnect the instance. After the instance is disconnected, it may be automatically reconnected depending on the service logic.

**Table 6-3** Plug-ins that can be enabled or disabled

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

**NOTE**

The ports of the plug-ins cannot be changed.

## Procedure

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

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

**NOTE**

Select the same region as your application service.

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

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

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

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

**----End**

## 6.10 Managing Instance Tags

Tags facilitate RabbitMQ instance identification and management.

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

A tag consists of a tag key and a tag value. **Table 6-4** lists the tag key and value requirements.

**Table 6-4** Tag key and value requirements

| Name | Rules |
|---|---|
| Tag key | ● Cannot be left blank.<br>● Must be unique for the same instance.<br>● Can contain a maximum of 36 characters.<br>● Cannot contain the following characters: =*<>\,\|/<br>● Cannot start or end with a space. |
| Tag value | ● Can contain a maximum of 43 characters.<br>● Cannot contain the following characters: =*<>\,\|/<br>● Cannot start or end with a space. |

### Procedure

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

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

NOTE

Select the same region as your application service.

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

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

**Step 5** Click the **Tags** tab.

View the tags of the instance.

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

- Adding a tag
  a. Click **Add Tag**.
     If you have created predefined tags, select a predefined pair of tag key and value. To view or create predefined tags, click **View predefined tags**. Then you will be directed to the TMS console.

You can also create new tags by entering **Tag key** and **Tag value**.

b. Click **OK**.

- Modifying a tag

  In the row containing the tag to be modified, click **Edit** in the **Operation** column. Enter the new tag value and click **OK**.
- Deleting a tag

  In the row containing the tag to be deleted, click **Delete** in the **Operation** column. Then click **Yes**.

**----End**

## 6.11 Deleting Queues

Delete queues on the RabbitMQ management UI or by calling APIs.

- **Method 1: Deleting a Single Queue on the Management UI**: Delete a single queue on the **Queues** tab page of the management UI.
- **Method 2: Deleting Queues in Batches Using a Policy**: 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 time-to-live (TTL) is 1 ms.
- **Method 3: Deleting a Single Queue Using an API**: Call an API to delete a queue from a RabbitMQ instance with SSL disabled.
- **Method 4: Deleting Queues in Batches Using an API**: Compile a shell script to repeatedly call an API to delete queues in batches from a RabbitMQ instance with SSL disabled.

NOTICE

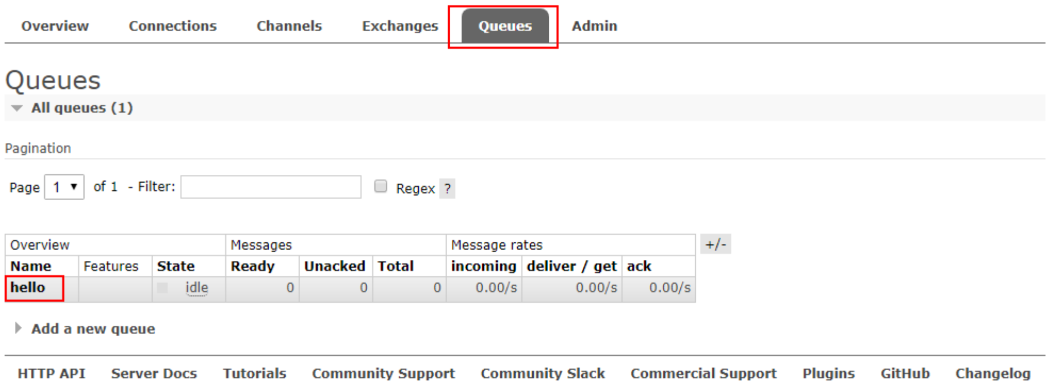
Before deleting a queue, ensure that all messages in the queue have been retrieved. Otherwise, unretrieved messages will be deleted together with the queue.

### Method 1: Deleting a Single Queue on the 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 6-5** Queue list



Queues

All queues (1)

Pagination

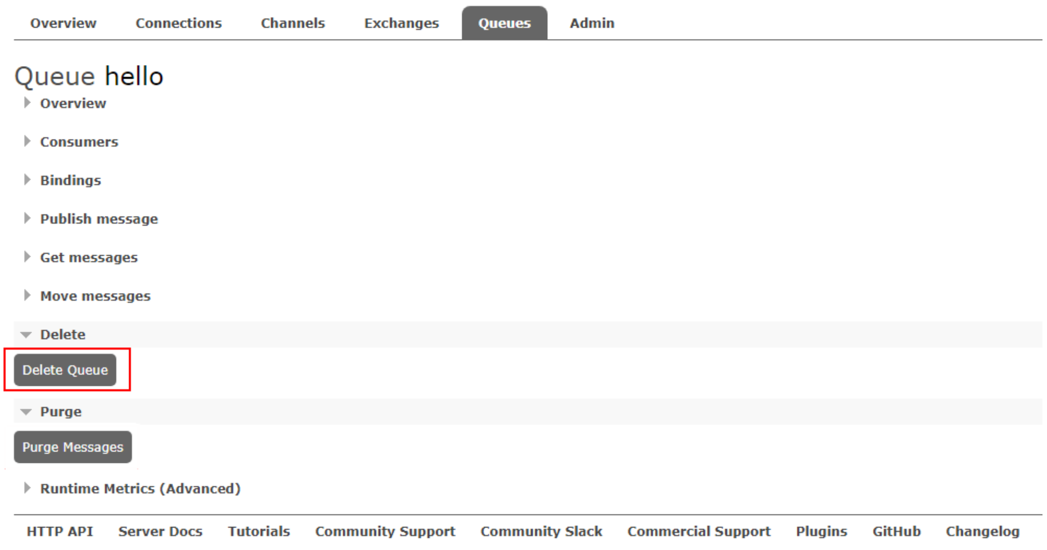
Page 1 of 1 - Filter: Regex ?

| Overview | | | Messages | | | Message rates | | | +/- |
|---|---|---|---|---|---|---|---|---|---|
| Name | Features | State | Ready | Unacked | Total | incoming | deliver / get | ack | |
| hello | | idle | 0 | 0 | 0 | 0.00/s | 0.00/s | 0.00/s | |

Add a new queue

**Step 3** Click **Delete Queue** to delete the queue.

**Figure 6-6** Deleting a single queue



**----End**

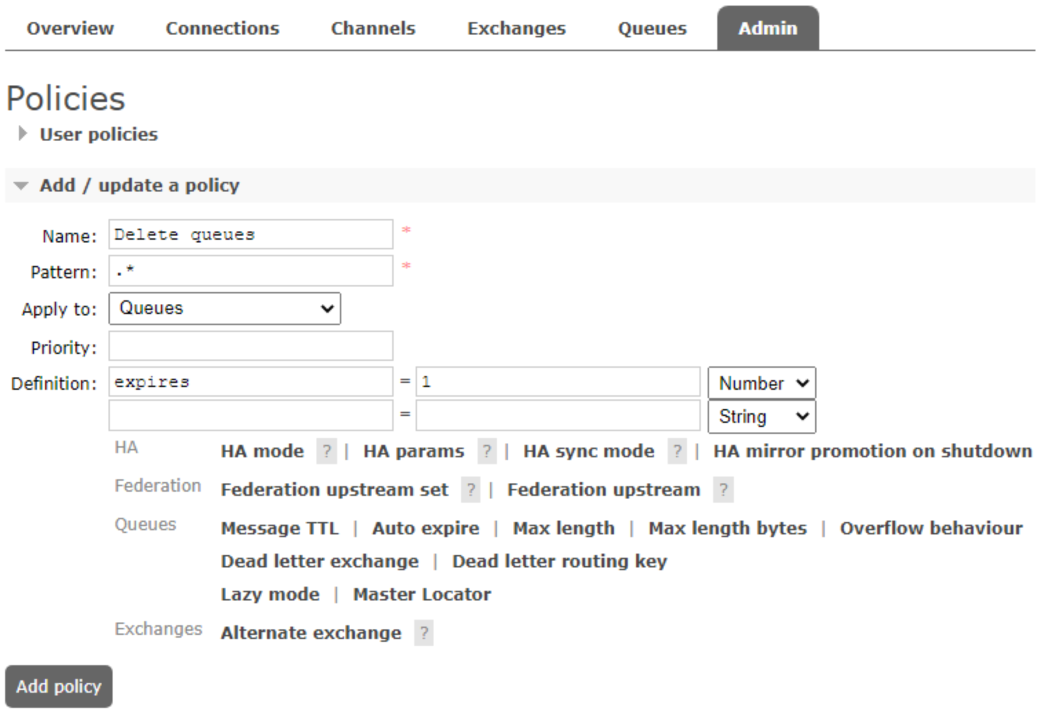
## Method 2: Deleting Queues in Batches Using a Policy

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 6-7** Adding a policy to delete queues in batches



| | | | |
|---|---|---|---|
| Name: | Delete queues | | |
| Pattern: | .* | | |
| Apply to: | Queues | | |
| Priority: | | | |
| Definition: | expires | = 1 | Number |
| | | = | String |

- **Name**: Enter a policy name.
- **Pattern**: queue matching mode. Enter a queue name. Queues with the same prefix will be matched. If this parameter is set to **.***, all queues are matched. If this parameter is set to **.*queue-name**, all queues whose name prefix is **queue-name** are matched.
- **Apply to**: Select **Queues**.
- **Priority**: policy priority. A larger value indicates a higher priority. This parameter is optional.
- **Definition**: TTL, in milliseconds. Set **expires** to **1**, indicating that the queue expiration time is 1 ms.

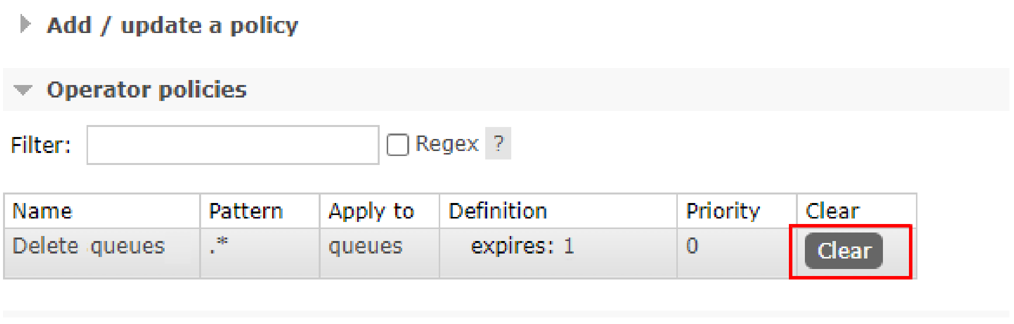
**Step 3** Click **Add policy**.

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

**Step 4** After the queues are deleted, choose **Admin** > **Policies**, locate the row that contains 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 6-8** Deleting the policy



| Name | Pattern | Apply to | Definition | Priority | Clear |
|---|---|---|---|---|---|
| Delete queues | .* | queues | expires: 1 | 0 | Clear |

**----End**

## Method 3: Deleting a Single Queue Using an API

If SSL is not enabled for a RabbitMQ instance, you can call an API to delete a queue.

**Step 1** Connect to the instance in Linux. For details, see **Accessing a RabbitMQ Instance Without SSL Encryption**.

**Step 2** Run the following command to delete a queue:

```
curl -i -XDELETE http://${USERNAME}:${PASSWORD}@${HOST}:${PORT}/api/queues/${VHOST_NAME}/$
{QUEUE_NAME}
```

Parameter description:

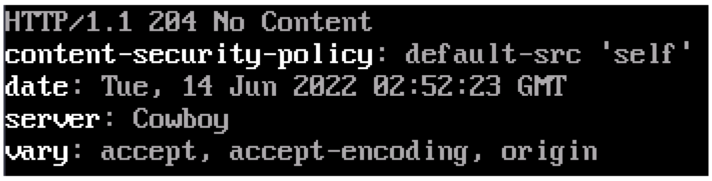
- **USERNAME**: username set when the instance was created.
- **PASSWORD**: password set when the instance was created. If you forget the password, reset it by referring to **Resetting the Instance Password**.
- **HOST**: management UI address queried on the instance details page.
- **PORT**: management UI port number queried on the instance details page.
- **VHOST_NAME**: vhost name. The default value is **/**. Change it to **%2F** in the command.
- **QUEUE_NAME**: name of the queue to be deleted.

Example:

```
curl-i -XDELETE http://test:Zsxxxdx@192.168.0.241:15672/api/queues/%2F/hello
```

If the deletion is successful, the following information is displayed.

**Figure 6-9** Queue deleted



```
HTTP/1.1 204 No Content
content-security-policy: default-src 'self'
date: Tue, 14 Jun 2022 02:52:23 GMT
server: Cowboy
vary: accept, accept-encoding, origin
```

You can also check whether the queue is successfully deleted on the **Queues** tab page of the management UI.

**----End**

## Method 4: Deleting Queues in Batches Using an API

If SSL is not enabled for a RabbitMQ instance, you can compile a shell script to repeatedly call an API to delete queues in batches.

**Step 1** Connect to the instance in Linux. For details, see **Accessing a RabbitMQ Instance Without SSL Encryption**.

**Step 2** Create the **delete_queues.sh** script.

```
touch delete_queues.sh
```

**Step 3** Edit the script.

```
vim delete_queues.sh
```

Copy the following content to the script. Change the values of **USERNAME**, **PASSWORD**, **HOST**, and **QUEUES_LIST** as required.

```
#!/usr/bin/env bash

USERNAME=root
PASSWORD=Zsxxxdx
HOST=192.168.0.241
PORT=15672
VHOST='%2F'


QUEUES_LIST="test1 test2 test3";
for QUEUE_NAME in $QUEUES_LIST :
do
   curl -i -XDELETE http://$USERNAME:$PASSWORD@$HOST:$PORT/api/queues/$VHOST/$QUEUE_NAME
done
```

Parameter description:

- **USERNAME**: username set when the instance was created.
- **PASSWORD**: password set when the instance was created. If you forget the password, reset it by referring to **Resetting the Instance Password**.
- **HOST**: management UI address queried on the instance details page.
- **PORT**: management UI port number queried on the instance details page.
- **VHOST**: vhost name. The default value is **/**. Change it to **%2F** in the command.
- **QUEUES_LIST**: names of the queues to be deleted. Use spaces to separate queue names.

**Step 4** Save the script.

**Step 5** Configure the script permissions.

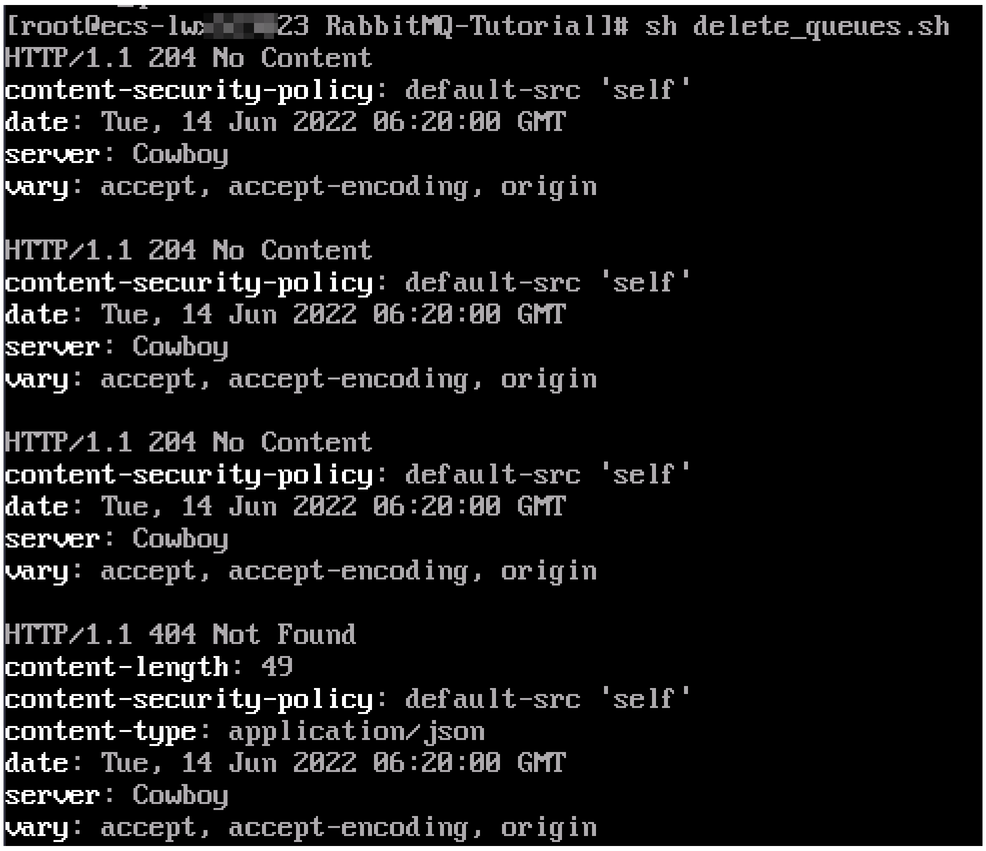
```
chmod 777 delete_queues.sh
```

**Step 6** Run the script.

```
sh delete_queues.sh
```

If the deletion is successful, the following information is displayed.

**Figure 6-10** Queues deleted



```
[root@ecs-lw[redacted]23 RabbitMQ-Tutorial]# sh delete_queues.sh
HTTP/1.1 204 No Content
content-security-policy: default-src 'self'
date: Tue, 14 Jun 2022 06:20:00 GMT
server: Cowboy
vary: accept, accept-encoding, origin

HTTP/1.1 204 No Content
content-security-policy: default-src 'self'
date: Tue, 14 Jun 2022 06:20:00 GMT
server: Cowboy
vary: accept, accept-encoding, origin

HTTP/1.1 204 No Content
content-security-policy: default-src 'self'
date: Tue, 14 Jun 2022 06:20:00 GMT
server: Cowboy
vary: accept, accept-encoding, origin

HTTP/1.1 404 Not Found
content-length: 49
content-security-policy: default-src 'self'
content-type: application/json
date: Tue, 14 Jun 2022 06:20:00 GMT
server: Cowboy
vary: accept, accept-encoding, origin
```

You can also check whether the queues are successfully deleted on the **Queues** tab page of the management UI.

**----End**

# 7 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 management console.
2. Click in the upper left corner to select a region and a project.
3. Click (the **My Quota** icon) in the upper right corner.

   The **Service Quota** page is displayed.
4. On the **Service Quota** 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?

The system does not support online quota adjustment. To increase a quota, contact customer service by calling the hotline or sending an email. We will process your request as soon as possible and will inform you of the processing progress by phone or email.

Before you contact customer service, prepare the following information:

- Account name, project name, and project ID

  To obtain the preceding information, log in to the management console, click the username in the upper-right corner, and choose **My Credentials** from the drop-down list.
- Quota information, including:
  – Service name

- Quota type
- Required quota

To increase a quota, contact the administrator.

# 8 Monitoring

## 8.1 RabbitMQ Metrics

### Introduction

This section describes metrics reported by DMS for RabbitMQ to Cloud Eye as well as their namespaces and dimensions. You can use the Cloud Eye console to query the RabbitMQ metrics and alarms.

### Namespace

SYS.DMS

### Instance Metrics

**Table 8-1** Instance metrics

| Metric ID | Metric Name | Description | Value Range | Monitored Object | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| connections | Connections | Number of connections in the RabbitMQ instance<br>Unit: count | ≥ 0 | RabbitMQ instance | 1 minute |
| channels | Channels | Number of channels in the RabbitMQ instance<br>Unit: count | 0–2047 | RabbitMQ instance | 1 minute |
| queues | Queues | Number of queues in the RabbitMQ instance<br>Unit: count | 0–1200 | RabbitMQ instance | 1 minute |

| Metric ID | Metric Name | Description | Value Range | Monitored Object | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| consumers | Consumers | Number of consumers in the RabbitMQ instance<br>Unit: count | 0–1200 | RabbitMQ instance | 1 minute |
| messages_ready | Available Messages | Number of messages that can be retrieved in the RabbitMQ instance<br>Unit: count | 0–10,000,000 | RabbitMQ instance | 1 minute |
| messages_unacknowledged | Unacked Messages | Number of messages that have been retrieved but not acknowledged in the RabbitMQ instance<br>Unit: count | 0–10,000,000 | RabbitMQ instance | 1 minute |
| publish | Publish | Rate at which messages are created in the RabbitMQ instance<br>Unit: Count/s | 0–25,000 | RabbitMQ instance | 1 minute |
| deliver | Deliver (manual ack) | Rate at which messages are retrieved (and manually acknowledged) in a RabbitMQ instance<br>Unit: Count/s | 0–25,000 | RabbitMQ instance | 1 minute |
| deliver_no_ack | Deliver (auto ack) | Rate at which messages are retrieved (and automatically acknowledged) in a RabbitMQ instance.<br>Unit: Count/s | 0–50,000 | RabbitMQ instance | 1 minute |

## Node Metrics

**Table 8-2** Node metrics

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

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

## Queue Metrics

**Table 8-3** Queue metrics

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

## Dimensions

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

## 8.2 Setting RabbitMQ Alarm Rules

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 8-4** Alarm rules for RabbitMQ instances

| Metric | Alarm Policy | Description | Solution |
|---|---|---|---|
| Memory High Watermark | Alarm threshold: Raw data ≥ 1<br>Number of consecutive periods: 1<br>Alarm severity: Critical | A threshold of 1 indicates that the memory high watermark is reached, blocking message publishing. | ● Accelerate message retrieval.<br>● 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<br>Number of consecutive periods: 1<br>Alarm severity: Critical | A threshold of 1 indicates that the disk high watermark is reached, blocking message publishing. | ● Reduce the number of messages accumulated in lazy queues.<br>● Reduce the number of messages accumulated in durable queues.<br>● Delete queues. |
| Memory Usage | Alarm threshold: Raw data > Expected usage (30% is recommended)<br>Number of consecutive periods: 3–5<br>Alarm severity: Major | To prevent high memory watermarks from blocking publishing, configure an alarm for this metric on each node. | ● Accelerate message retrieval.<br>● 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)<br>Number of consecutive periods: 3–5<br>Alarm severity: Major | A high CPU usage may slow down publishing rate. Configure an alarm for this metric on each node. | ● Reduce the number of mirrored queues.<br>● For a cluster instance, add nodes and rebalance queues between all nodes. |

| Metric | Alarm Policy | Description | Solution |
|---|---|---|---|
| Available Messages | Alarm threshold: Raw data > Expected number of available messages<br>Number of consecutive periods: 1<br>Alarm severity: Major | If the number of available messages is too large, messages are accumulated. | See **the solution to preventing message accumulation**. |
| Unacked Messages | Alarm threshold: Raw data > Expected number of unacknowledged messages<br>Number of consecutive periods: 1<br>Alarm severity: Major | If the number of unacknowledged messages is too large, messages may be accumulated. | • Check whether the consumer is abnormal.<br>• Check whether the consumer logic is time-consuming. |
| Connections | Alarm threshold: Raw data > Expected number of connections<br>Number of consecutive periods: 1<br>Alarm severity: Major | A sharp increase in the number of connections may be a warning of a traffic increase. | The services may be abnormal. Check whether other alarms exist. |
| Channels | Alarm threshold: Raw data > Expected number of channels<br>Number of consecutive periods: 1<br>Alarm severity: Major | A sharp increase in the number of channels may be a warning of a traffic increase. | The services may be abnormal. Check whether other alarms exist. |

| Metric | Alarm Policy | Description | Solution |
|---|---|---|---|
| Erlang Processes | Alarm threshold: Raw data > Expected number of processes<br>Number of consecutive periods: 1<br>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. |

**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.

## Procedure

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

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

**NOTE**

Select the same region as your application service.

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

**Step 4** In the row containing the desired RabbitMQ instance, click **View Metric**.

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

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

For more information about creating alarm rules, see the *Cloud Eye User Guide*.

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**

## 8.3 Viewing Metrics

### Scenario

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

### Prerequisites

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

### Procedure

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

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

**NOTE**

Select the same region as your application service.

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

**Step 4** In the row containing the desired RabbitMQ 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.

**----End**

# 9 Auditing

## 9.1 Operations Logged by CTS

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

**Table 9-1** DMS for RabbitMQ operations that can be recorded 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 | modifyDMSInstanceOrderSuc-cess |
| Failing to create an order for modifying an instance | rabbitmq | modifyDMSInstanceOrderFailure |
| Successfully scaling up an instance | rabbitmq | extendDMSInstanceSuccess |
| Failing to scale up an instance | rabbitmq | extendDMSInstanceFailure |

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

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

## 9.2 Viewing Audit Logs

See **Querying Traces on the CTS Console**.

# 10 FAQs

## 10.1 Instances

### 10.1.1 What RabbitMQ Version Does DMS for RabbitMQ Use?

The RabbitMQ version on the server is 3.7.17.

### 10.1.2 What SSL Version Does DMS for RabbitMQ Use?

TLS 1.2.

### 10.1.3 Why Can't I View the Subnet and Security Group Information During Instance Creation?

This may be because you do not have the permissions of the server administrator and VPC administrator. For details on how to add user permissions, see the *Identity and Access Management User Guide*.

### 10.1.4 What If One RabbitMQ VM Fails to Be Restarted When a Cluster RabbitMQ Instance Is Being Restarted?

Restarting a RabbitMQ instance will restart only the RabbitMQ processes instead of VMs on which the instance runs.

For a cluster RabbitMQ instance, if the RabbitMQ process fails to be restarted on one VM, the instance will be still in the **Running** state after the restart and a message will be displayed indicating that some nodes are faulty. A RabbitMQ daemon process runs on each VM and periodically checks whether the RabbitMQ process exists. If the RabbitMQ process does not exist, the RabbitMQ process will be automatically started.

If a RabbitMQ instance exception lasts for more than 1 minute, an alarm will be reported.

## 10.1.5 How Are Requests Evenly Distributed to Each VM of a Cluster RabbitMQ Instance?

A cluster uses Linux virtual servers (LVSs) inside for load balancing, which evenly distribute requests to each VM.

## 10.1.6 Do Queues Inside a Cluster RabbitMQ Instance Have Any Redundancy Backup?

Whether queue mirroring (that is, redundancy backup) is implemented depends on your service requirements. If you configure mirroring, queue replicas are stored on multiple brokers in a cluster. If a broker is faulty, queue data is synchronized from another normal broker.

## 10.1.7 Does DMS for RabbitMQ Support Data Persistence? How Do I Perform Scheduled Data Backups?

DMS for RabbitMQ supports data persistence, which can be configured by connecting to a RabbitMQ instance using a client, or by configuring persistence when creating queues using the RabbitMQ Management interface.

Unfortunately, data backup scheduling and backup operations on the console are not supported.

## 10.1.8 How Do I Obtain the Certificate After SSL Has Been Enabled?

DMS for RabbitMQ uses one-way authentication and does not involve any certificates.

## 10.1.9 Can I Change the SSL Setting of a RabbitMQ Instance?

No. Once the instance has been created, you cannot enable or disable SSL. You are advised to enable SSL when creating the instance.

## 10.1.10 Can RabbitMQ Instances Be Scaled Up?

Single-node RabbitMQ instances: The storage space cannot be increased.

Cluster RabbitMQ instances: The storage space cannot be increased, but the node quantity can be increased.

## 10.1.11 Does DMS for RabbitMQ Support MQTT?

Yes. The MQTT plug-in can be enabled. For details about the MQTT plug-in, see **https://www.rabbitmq.com/mqtt.html#enabling-plugin**.

The supported plug-ins are listed as follows.

In the following list, an empty square bracket indicates that the plug-in has not been installed. **[E*]** indicates that the plug-in has been explicitly installed. **[e*]** indicates that the plug-in has been implicitly installed, that is, the plug-in is installed as the dependency of other plug-ins.

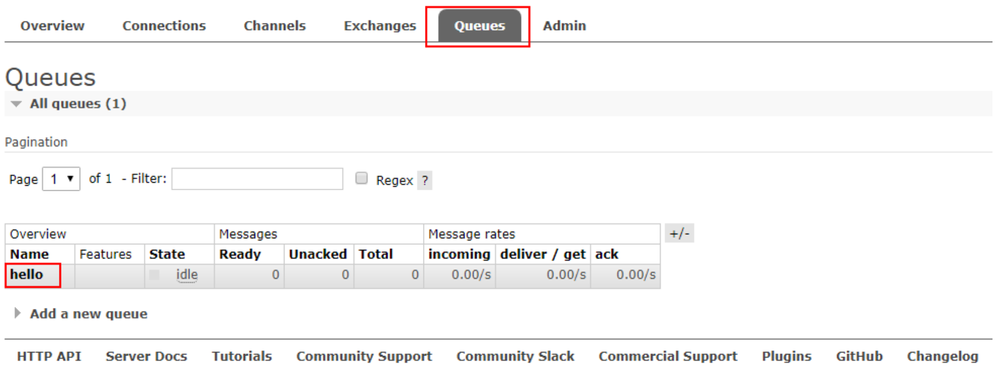
```
[  ] rabbitmq_amqp1_0                  3.7.17
[  ] rabbitmq_auth_backend_cache       3.7.17
[  ] rabbitmq_auth_backend_http        3.7.17
[  ] rabbitmq_auth_backend_ldap        3.7.17
[  ] rabbitmq_auth_mechanism_ssl       3.7.17
[  ] rabbitmq_consistent_hash_exchange 3.7.17
[  ] rabbitmq_delayed_message_exchange 3.8.0
[  ] rabbitmq_event_exchange           3.7.17
[  ] rabbitmq_federation               3.7.17
[  ] rabbitmq_federation_management    3.7.17
[  ] rabbitmq_jms_topic_exchange       3.7.17
[E*] rabbitmq_management               3.7.17
[e*] rabbitmq_management_agent         3.7.17
[  ] rabbitmq_mqtt                     3.7.17
[  ] rabbitmq_peer_discovery_aws       3.7.17
[  ] rabbitmq_peer_discovery_common    3.7.17
[  ] rabbitmq_peer_discovery_consul    3.7.17
[  ] rabbitmq_peer_discovery_etcd      3.7.17
[  ] rabbitmq_peer_discovery_k8s       3.7.17
[  ] rabbitmq_random_exchange          3.7.17
[  ] rabbitmq_recent_history_exchange  3.7.17
[  ] rabbitmq_sharding                 3.7.17
[  ] rabbitmq_shovel                   3.7.17
[  ] rabbitmq_shovel_management        3.7.17
[  ] rabbitmq_stomp                    3.7.17
[E*] rabbitmq_top                      3.7.17
[  ] rabbitmq_tracing                  3.7.17
[  ] rabbitmq_trust_store              3.7.17
[e*] rabbitmq_web_dispatch             3.7.17
[  ] rabbitmq_web_mqtt                 3.7.17
[  ] rabbitmq_web_mqtt_examples        3.7.17
[  ] rabbitmq_web_stomp                3.7.17
[  ] rabbitmq_web_stomp_examples       3.7.17
```

The plug-ins that you can enable on the RabbitMQ console are rabbitmq_amqp1_0, rabbitmq_delayed_message_exchange, rabbitmq_federation, rabbitmq_sharding, rabbitmq_shovel, rabbitmq_tracing, rabbitmq_mqtt, rabbitmq_web_mqtt, rabbitmq_stomp, rabbitmq_web_stomp, and rabbitmq_consistent_hash_exchange. When an instance is created, these plug-ins are disabled by default. To enable them, go to the **Plug-ins** page of an instance on the RabbitMQ console.

To install plug-ins that cannot be enabled on the console (such as rabbitmq_random_exchange), contact customer service. Services are not affected during the installation.

## 10.1.12 How Do I Clear Queue Data?

1. Log in to the Management Web UI. For details, see the "Connecting to the Management Address of a RabbitMQ Instance" section.
2. On the **Queues** tab page, click the name of a queue.



Queues

All queues (1)

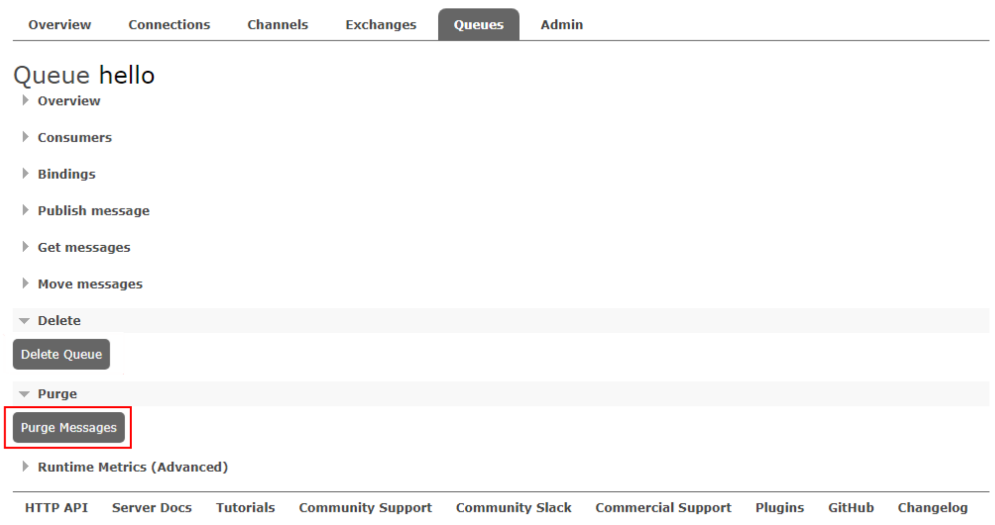
Pagination

Page 1 of 1 - Filter: Regex ?

| Overview | | | Messages | | | Message rates | | |
|---|---|---|---|---|---|---|---|---|
| Name | Features | State | Ready | Unacked | Total | incoming | deliver / get | ack |
| hello | | idle | 0 | 0 | 0 | 0.00/s | 0.00/s | 0.00/s |

Add a new queue

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



## 10.1.13 Does DMS for RabbitMQ Support CPU and Memory Upgrades?

No. DMS for RabbitMQ does not support CPU and memory upgrades.

## 10.1.14 How Do I Disable the RabbitMQ Management UI?

If you want to disable login to the management UI of a RabbitMQ instance, do not allow inbound access over port 15672 (if SSL is disabled for the instance) or 15671 (if SSL is enabled for the instance) in the security group rules.

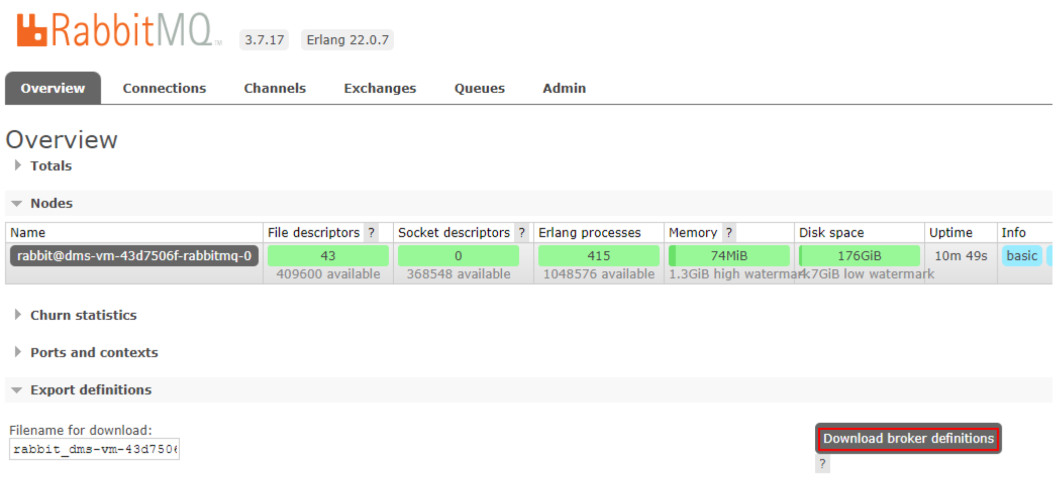
## 10.1.15 Can I Change the AZ for an Instance?

No. To use a different AZ, create another instance and migrate metadata to it.

To migrate instance metadata, perform the following steps:

**Step 1** Log in to the management UI of the original instance.

**Step 2** On the **Overview** tab page, click **Download broker definitions** to export the metadata.

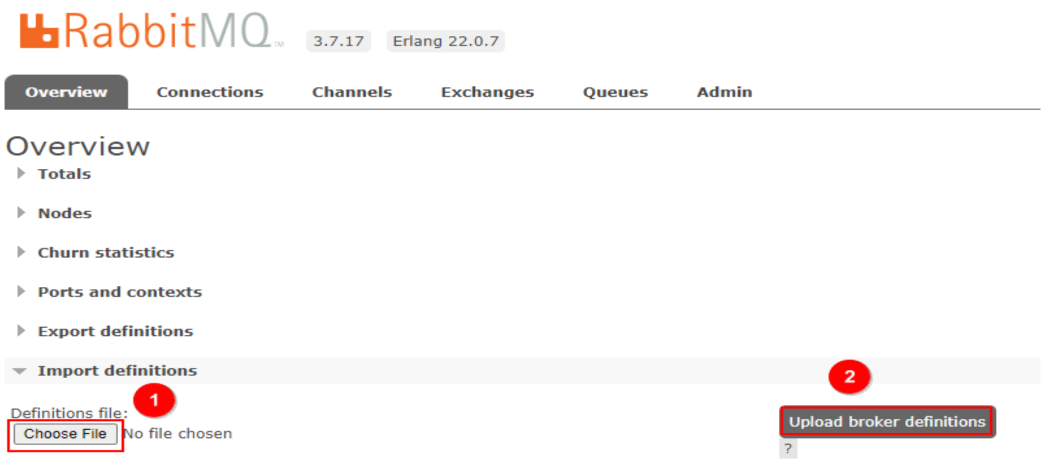


| Name | File descriptors ? | Socket descriptors ? | Erlang processes | Memory ? | Disk space | Uptime | Info |
|---|---|---|---|---|---|---|---|
| rabbit@dms-vm-43d7506f-rabbitmq-0 | 43<br>409600 available | 0<br>368548 available | 415<br>1048576 available | 74MiB<br>1.3GiB high watermark | 176GiB<br>47GiB low watermark | 10m 49s | basic |

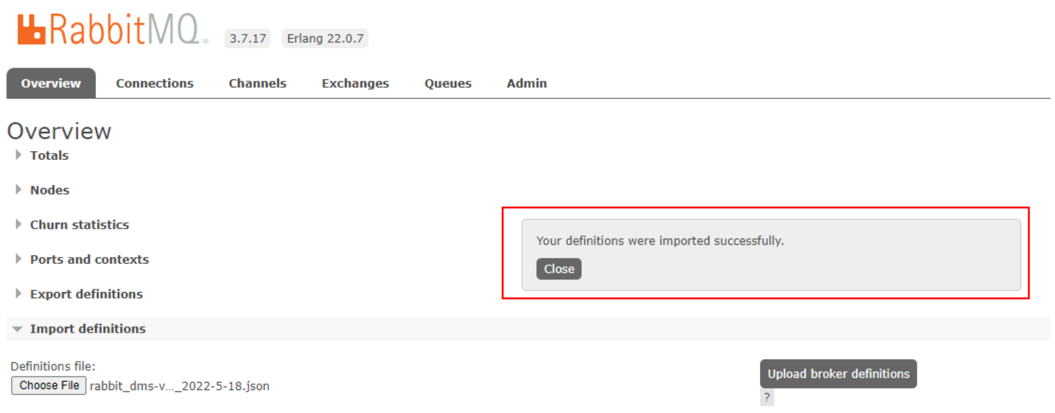
Filename for download:
rabbit_dms-vm-43d750

**Step 3** Log in to the management UI of the new instance. On the **Overview** tab page, click **Choose File** and select the metadata exported from **Step 2**.

**Step 4** Click **Upload broker definitions** to upload the metadata.



If the upload is successful, the following information is displayed:



Your definitions were imported successfully.

**----End**

## 10.2 Connections

### 10.2.1 How Do I Configure a Security Group?

To access a RabbitMQ instance within a VPC or over public networks, configure the security group rules as follows.

- Intra-VPC Access

  To access a RabbitMQ instance, you must deploy your client on an ECS in the same VPC and subnet as the instance.

  In addition, before you can access the instance through your client, you must configure correct rules for the security groups of both the ECS and RabbitMQ instance.

  a. You are advised to configure the same security group for the ECS and RabbitMQ instance. After a security group is created, network access in the group is not restricted by default.
  b. If different security groups are configured, you may need to refer to the following configurations:

  NOTE

  - Assume that security groups **sg-53d4**, **sg-RabbitMQ**, and **Default_All** are configured respectively for your ECS and RabbitMQ instance.
  - You can specify a security group or IP address as the remote end in the following rules.

  Add the following security group rule to allow the ECS to access the RabbitMQ instance.

  **Table 10-1** Security group rule

| Direction | Protocol & Port | Destination |
|---|---|---|
| Outbound | All | Default_All |

  To ensure that your client can access the RabbitMQ instance, add the following rule to the security group configured for the RabbitMQ instance.

  **Table 10-2** Security group rule

| Direction | Protocol & Port | Source |
|---|---|---|
| Inbound | All | sg-53d4 |

- Public access:

  A client can access a RabbitMQ instance only after rules are correctly configured for the security group of the instance.

For example, for security group **sg-RabbitMQ**, you need to configure either of the following rules in the inbound direction:

a. Protocol: **TCP**; port number: **5672**; source IP address: **0.0.0.0/0**
b. Protocol: **Any**; source IP address: **0.0.0.0/0**

## 10.2.2 Why Does a Client Fail to Connect to a RabbitMQ Instance?

This problem occurs when the connection address, port number, username, or password is incorrect, or when the maximum allowed number of connections is exceeded.

- **Incorrect connection address**

  Error message displayed for an incorrect connection address in intra-VPC access:

```
[root@ecs-heru RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.125.110 5672 user
*******
Exception in thread "main" java.net.NoRouteToHostException: No route to host (Host unreachable)
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
```

  Error message displayed for an incorrect connection address in public access:

```
[root@ecs-heru RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 139.xxx.178 5672 user *******
Exception in thread "main" java.net.SocketTimeoutException: connect timed out
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
```

- **Incorrect port number**

  Error message displayed for an incorrect port number in intra-VPC access:

```
[root@ecs-heru RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.125.111 5673 user
*******
Exception in thread "main" java.net.ConnectException: Connection refused (Connection refused)
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
```

  Error message displayed for an incorrect port number in public access:

```
[root@ecs-heru RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 139.xxx.179 5673 user *******
Exception in thread "main" java.net.SocketTimeoutException: connect timed out
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
```

- **Incorrect username or password**

```
[root@ecs-heru RabbitMQ-Tutorial]# java -cp .:rabbitmq-tutorial.jar Send 192.168.125.111 5672 user
*******
Exception in thread "main" com.rabbitmq.client.AuthenticationFailureException: ACCESS_REFUSED -
Login was refused using authentication mechanism PLAIN. For details
 see the broker logfile.
at com.rabbitmq.client.impl.AMQConnection.start(AMQConnection.java:351)
at
com.rabbitmq.client.impl.recovery.RecoveryAwareAMQConnectionFactory.newConnection(RecoveryAwa
reAMQConnectionFactory.java:64)
```

- **Maximum number of connections exceeded**

## 10.2.3 Does DMS for RabbitMQ Support Public Access?

Yes. You can enable public access on the instance creation page when creating the instance, or on the instance details page after the instance has been created.

## 10.2.4 Does DMS for RabbitMQ Support Cross-Region Deployment?

No. Currently, only cross-AZ deployment is supported. Cross-region deployment is not supported.

## 10.2.5 Does DMS for RabbitMQ Support Cross-VPC Access?

Yes. RabbitMQ instances support cross-VPC and cross-subnet access. By establishing a peering connection between two VPCs, ECSs in one VPC can access instances in the other VPC.

## 10.2.6 Does DMS for RabbitMQ Support Cross-Subnet Access?

Yes.

If the client and the instance are in the same VPC, cross-subnet access is supported.

If the client and the instance are in different VPCs, establish a VPC peering connection.

You can also access an instance through the public access address bound to the instance.

## 10.2.7 What Should I Do If I Fail to Access a RabbitMQ Instance with SSL Encryption?

1. Check the inbound rule of the security group. You must allow access using port 5671 (with SSL encryption) or 5672 (without SSL encryption).

2. Configure one-way SSL authentication as follows:

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

## 10.2.8 Can I Access a RabbitMQ Instance Using DNAT?

No. You can only use a proxy, VPN, Direct Connect, FullNAT, or reverse proxy to access a RabbitMQ instance.

## 10.2.9 Why Can't I Open the Management Web UI?

Possible cause: The security group of the instance is incorrectly configured.

Solution: Do as follows to reconfigure the security group.

1. In the **Network** section on the **Basic Information** tab page, click the name of the security group.
2. Click the **Inbound Rules** tab to view the inbound rules of the security group.

- SSL disabled
  - For intra-VPC access, inbound access through port 5672 must be allowed.
  - For public access, inbound access through port 15672 must be allowed.
- SSL enabled
  - For intra-VPC access, inbound access through port 5671 must be allowed.
  - For public access, inbound access through port 15671 must be allowed.

### 10.2.10 Can a Client Connect to Multiple Virtual Hosts of a RabbitMQ Instance?

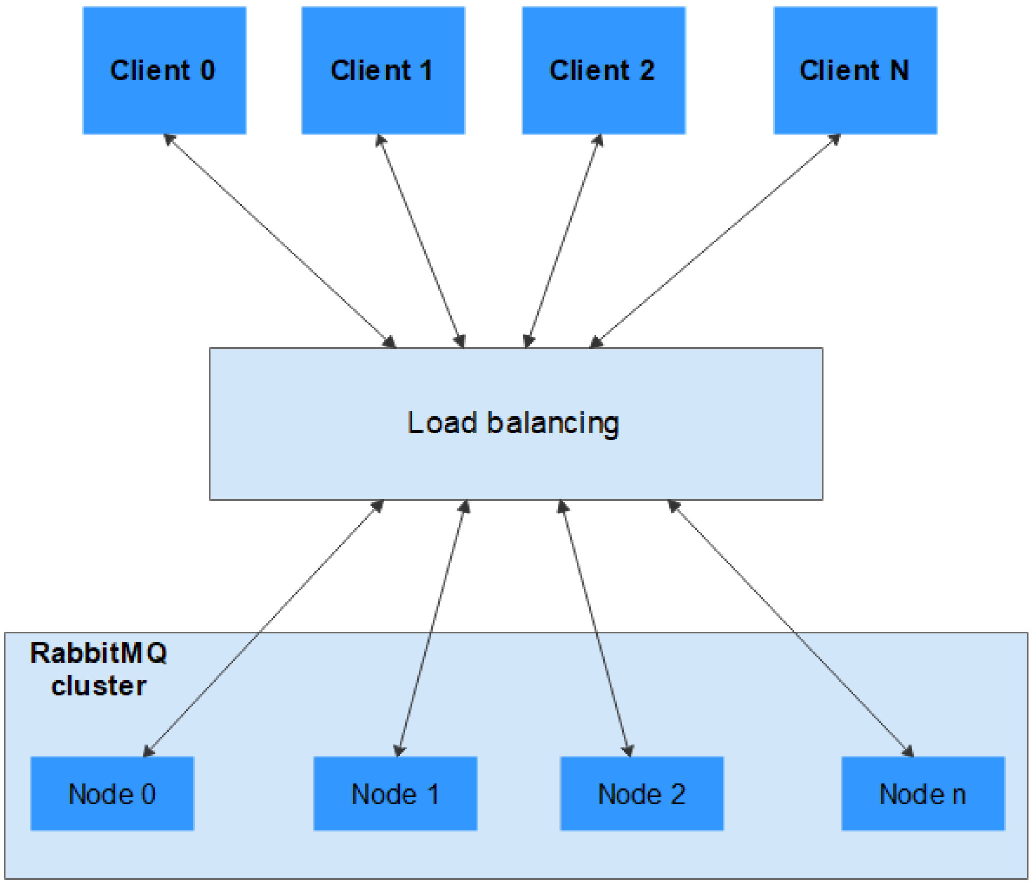
Yes.

Virtual hosting is a basic feature of RabbitMQ. Each virtual host serves as an independent RabbitMQ server. Different virtual hosts have different data directories but share the same process. Connecting to multiple virtual hosts does not differ much in performance from connecting to one virtual host. The only difference is that the RabbitMQ process has more objects. You are advised to test the performance by using the service model.

For details, see **Virtual Hosts** on the official RabbitMQ website.

### 10.2.11 Why Does a RabbitMQ Cluster Have Only One Connection Address?

The connection address of a cluster RabbitMQ instance is actually the LVS node address (load balancing address) of the instance. When clients connect to the instance, the load balancer distributes the client request to each node of the cluster instance.



## 10.3 Plug-ins

### 10.3.1 What Plug-ins Does DMS for RabbitMQ Support?

#### What Plug-ins Does DMS for RabbitMQ Support?

The supported plug-ins are listed as follows. In the following list, an empty square bracket indicates that the plug-in is not installed; **[E*]** indicates that the plug-in has been explicitly installed; **[e*]** indicates that the plug-in has been implicitly installed, that is, the plug-in is installed as the dependency of other plug-ins.

```
[  ] rabbitmq_amqp1_0                  3.7.17
[  ] rabbitmq_auth_backend_cache       3.7.17
[  ] rabbitmq_auth_backend_http        3.7.17
[  ] rabbitmq_auth_backend_ldap        3.7.17
[  ] rabbitmq_auth_mechanism_ssl       3.7.17
[  ] rabbitmq_consistent_hash_exchange 3.7.17
[  ] rabbitmq_delayed_message_exchange 3.8.0
[  ] rabbitmq_event_exchange           3.7.17
[  ] rabbitmq_federation               3.7.17
[  ] rabbitmq_federation_management    3.7.17
[  ] rabbitmq_jms_topic_exchange       3.7.17
[E*] rabbitmq_management               3.7.17
[e*] rabbitmq_management_agent         3.7.17
[  ] rabbitmq_mqtt                     3.7.17
[  ] rabbitmq_peer_discovery_aws       3.7.17
[  ] rabbitmq_peer_discovery_common    3.7.17
[  ] rabbitmq_peer_discovery_consul    3.7.17
[  ] rabbitmq_peer_discovery_etcd      3.7.17
[  ] rabbitmq_peer_discovery_k8s       3.7.17
[  ] rabbitmq_random_exchange          3.7.17
[  ] rabbitmq_recent_history_exchange  3.7.17
[  ] rabbitmq_sharding                 3.7.17
[  ] rabbitmq_shovel                   3.7.17
[  ] rabbitmq_shovel_management        3.7.17
[  ] rabbitmq_stomp                    3.7.17
[E*] rabbitmq_top                      3.7.17
[  ] rabbitmq_tracing                  3.7.17
[  ] rabbitmq_trust_store              3.7.17
[e*] rabbitmq_web_dispatch             3.7.17
[  ] rabbitmq_web_mqtt                 3.7.17
[  ] rabbitmq_web_mqtt_examples        3.7.17
[  ] rabbitmq_web_stomp                3.7.17
[  ] rabbitmq_web_stomp_examples       3.7.17
```

The plug-ins that you can enable on the RabbitMQ console are rabbitmq_amqp1_0, rabbitmq_delayed_message_exchange, rabbitmq_federation, rabbitmq_sharding, rabbitmq_shovel, rabbitmq_tracing, rabbitmq_mqtt, rabbitmq_web_mqtt, rabbitmq_stomp, rabbitmq_web_stomp, and rabbitmq_consistent_hash_exchange. When an instance is created, these plug-ins are disabled by default. To enable them, go to the **Plug-ins** page of an instance on the RabbitMQ console.

To install other plug-ins, contact customer service. Services are not affected during the installation.

## 10.4 Messages

### 10.4.1 Does DMS for RabbitMQ Support Delayed Message Delivery?

Yes. To delay message delivery, you can set the time-to-live (TTL) and dead letter exchanges (DLX). You can also use plug-ins to delay message delivery. Currently, the following plug-ins are supported: rabbitmq_amqp1_0, rabbitmq_delayed_message_exchange, rabbitmq_federation, rabbitmq_sharding, rabbitmq_shovel, rabbitmq_tracing, rabbitmq_mqtt, rabbitmq_web_mqtt, rabbitmq_stomp, rabbitmq_web_stomp, and rabbitmq_consistent_hash_exchange.

### 10.4.2 How Does Message Accumulation Affect Services? What Can I Do?

#### How Does Message Accumulation Affect Services?

Excessive message accumulation in a queue may cause memory or disk alarms. As a result, all connections will be blocked, other queues cannot be used, and the overall service quality deteriorates.

#### Causes of Message Accumulation

1. Messages are published much faster than they are retrieved. For example, consumers process messages slowly in a certain period. It may take only 3

seconds to send a message, but 1 minute to retrieve the message. If 20 messages are sent per minute, and only one message is processed by consumers, a large number of messages will be stacked in the queue.

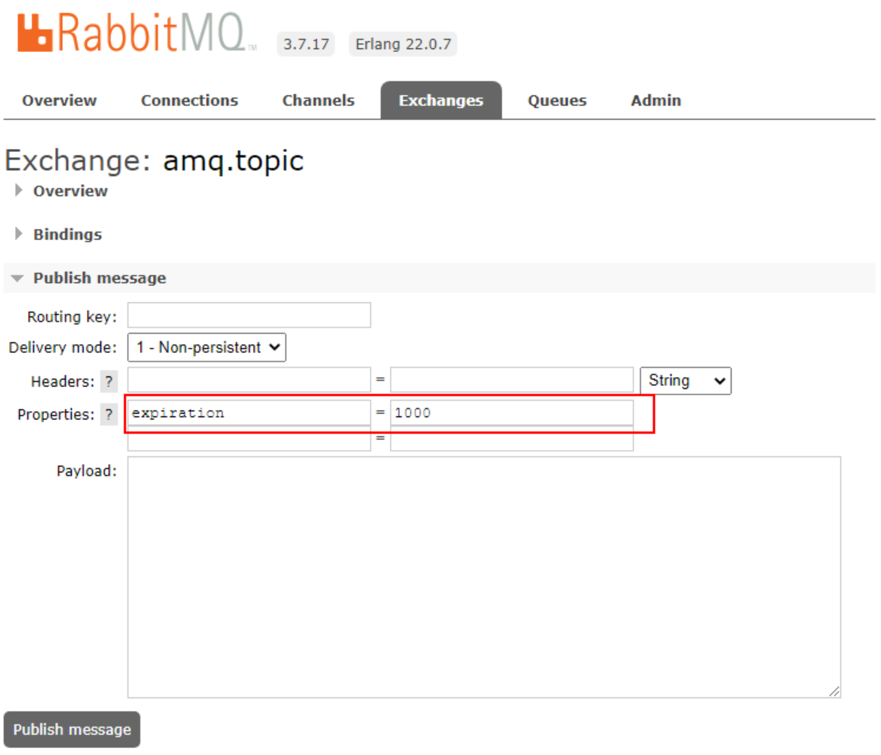
2. Consumers are abnormal and cannot retrieve messages, while publishers keep sending messages.
3. Consumers are normal, but their subscriptions to queues are abnormal.
4. Consumers and their subscriptions to queues are normal, but the code logic of consumers is time-consuming, which reduces the consumption capability and results in a situation similar to **1**.

## Solutions to Message Accumulation

1. If messages are published much faster than they are retrieved, solve the problem in the following ways:
   - Add consumers to 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.
2. If consumers are abnormal, check whether the consumer logic is correct and optimize the program.
3. Check whether consumers' subscriptions to queues are normal.
4. If the code logic of consumers is time-consuming, set expiration time on messages using either of the following methods:
   - When creating messages, set the message expiration time by using the **expiration** parameter.
     - Set the value of **expiration** in **properties**. The unit is ms.
       ```
       AMQP.BasicProperties properties = new AMQP.BasicProperties().builder()
           .deliveryMode(2)
           .contentEncoding("UTF-8")
           .expiration("10000")
           .build();

       String message = "hello rabbitmq";
       channel.basicPublish(exchange, routingKey, properties,
       message.getBytes(StandardCharsets.UTF_8));
       ```
     - Set the value of **expiration** on the management UI. The unit is ms.

       **Log in to the management UI**. On the **Exchanges** tab page, click an exchange name to view its details. In the **Publish message** area, set **expiration**, as shown in the following figure.



Exchange: amq.topic

| | | |
|---|---|---|
| Routing key: | | |
| Delivery mode: | 1 - Non-persistent | |
| Headers: ? | = | String |
| Properties: ? | expiration | = 1000 |
| Payload: | | |

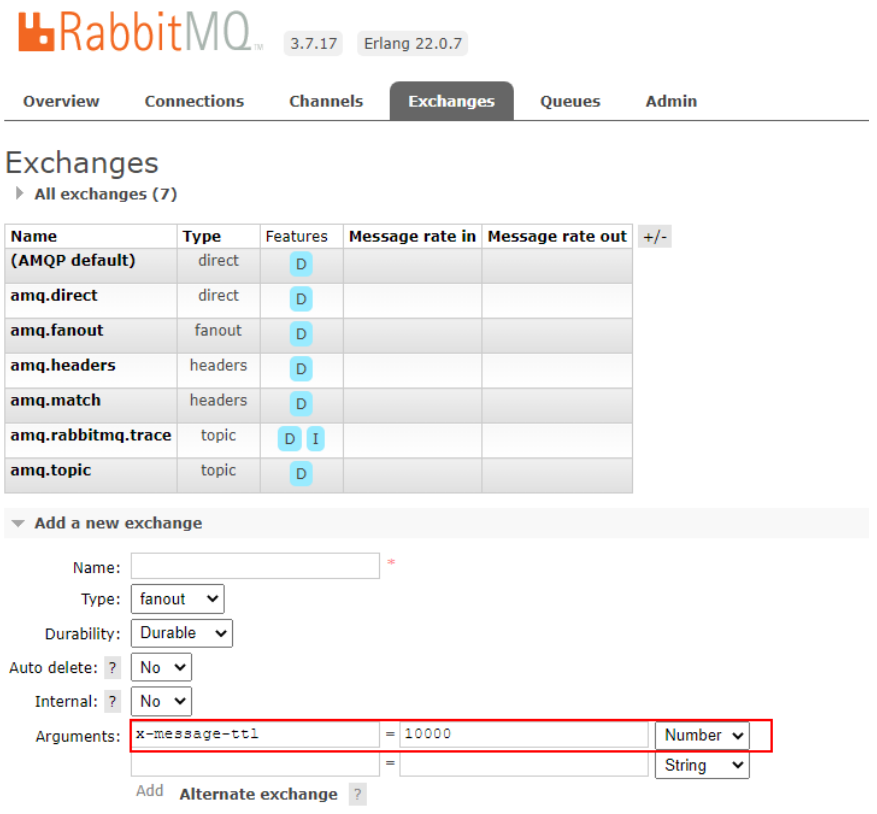
– Set the queue expiration time by using the **x-message-ttl** parameter. The expiration time starts when messages enter the queue. When the expiration time elapses, the messages are automatically deleted.

- Set the value of **x-message-ttl** in the client code. The unit is ms.

```
Map<String, Object> arguments = new HashMap<String, Object>();
arguments.put("x-message-ttl", 10000);
channel.queueDeclare(queueName, true, false, false, arguments);
```

- Set the value of **x-message-ttl** when creating a queue on the management UI. The unit is ms.

  **Log in to the management UI**. On the **Exchanges** tab page, create a queue and set the value of **x-message-ttl**, as shown in the following figure.



Exchanges

All exchanges (7)

| Name | Type | Features | Message rate in | Message rate out |
|---|---|---|---|---|
| (AMQP default) | direct | D | | |
| amq.direct | direct | D | | |
| amq.fanout | fanout | D | | |
| amq.headers | headers | D | | |
| amq.match | headers | D | | |
| amq.rabbitmq.trace | topic | D I | | |
| amq.topic | topic | D | | |

Add a new exchange

| | |
|---|---|
| Name: | * |
| Type: | fanout |
| Durability: | Durable |
| Auto delete: ? | No |
| Internal: ? | No |
| Arguments: | x-message-ttl = 10000 Number |
| | = String |

Add Alternate exchange ?

## 10.4.3 How Long Are Messages Be Retained?

Normally, messages are retained until they are retrieved. However, if a message has a time to live (TTL), it will be retained until expiry.

## 10.5 Monitoring & Alarm

### 10.5.1 Why Can't I View the Monitoring Data of a RabbitMQ Instance?

The monitoring data cannot be displayed if the queue name starts with a special character, such as a period (.) or underscore (_). You are advised to delete the queue whose name starts with a special character.

### 10.5.2 What Should I Do If the Number of Channels Keeps Rising?

A maximum of 2047 channels are allowed in each connection. If this limit is exceeded, new channels cannot be created. Check if unused resources are not released.

# A Change History

| Released On | Description |
|---|---|
| 2022-08-12 | This issue is the first official release. |