

Distributed Message Service for RabbitMQ

User Guide

Issue	05
Date	2025-09-03



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

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

Trademarks and Permissions



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

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

Notice

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

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

Contents

- 1 Service Overview..... 1**
 - 1.1 What Is DMS for RabbitMQ?..... 1
 - 1.2 Product Advantages..... 1
 - 1.3 Application Scenarios..... 2
 - 1.4 Specifications..... 4
 - 1.5 Comparing RabbitMQ, Kafka, and RocketMQ..... 7
 - 1.6 Related Services..... 8
 - 1.7 Notes and Constraints..... 9
 - 1.8 Basic Concepts..... 13
 - 1.9 Exchanges..... 14
 - 1.10 Permissions Management..... 16
- 2 Getting Started..... 20**
 - 2.1 Getting Started with RabbitMQ for Message Production and Consumption..... 20
- 3 Process of Using RabbitMQ.....27**
- 4 Permissions Management..... 29**
 - 4.1 Creating an IAM User and Granting DMS for RabbitMQ Permissions..... 29
- 5 Buying a RabbitMQ Instance..... 33**
- 6 Configuring Virtual Hosts..... 44**
 - 6.1 Creating a RabbitMQ Virtual Host.....44
 - 6.2 Creating a RabbitMQ Exchange.....46
 - 6.3 Binding a RabbitMQ Exchange..... 48
 - 6.4 Creating a RabbitMQ Queue..... 50
 - 6.5 Binding a RabbitMQ Queue..... 52
 - 6.6 Managing RabbitMQ Virtual Hosts..... 53
 - 6.6.1 Viewing a RabbitMQ Virtual Host..... 53
 - 6.6.2 Deleting RabbitMQ Virtual Hosts..... 55
 - 6.7 Managing RabbitMQ Exchanges..... 57
 - 6.7.1 Unbinding a RabbitMQ Exchange..... 57
 - 6.7.2 Deleting RabbitMQ Exchanges..... 58
 - 6.8 Managing RabbitMQ Queues..... 59
 - 6.8.1 Viewing a RabbitMQ Queue..... 59

6.8.2 Clearing Messages in a RabbitMQ Queue.....	62
6.8.3 Unbinding a RabbitMQ Queue.....	63
6.8.4 Configuring Queue Mirroring.....	64
6.8.5 Configuring Lazy Queues.....	67
6.8.6 Configuring RabbitMQ Quorum Queues.....	70
6.8.7 Configuring a RabbitMQ Exclusive Queue.....	76
6.8.8 Configuring a Single Active Consumer.....	76
6.8.9 Deleting RabbitMQ Queues.....	79
7 Accessing a RabbitMQ Instance.....	83
7.1 Configuring RabbitMQ Network Connections.....	83
7.1.1 RabbitMQ Network Connection Requirements.....	83
7.1.2 Configuring RabbitMQ Public Access.....	84
7.2 Configuring Heartbeats on the RabbitMQ Client.....	87
7.3 Accessing RabbitMQ on a Client (SSL Disabled).....	89
7.4 Accessing RabbitMQ on a Client (SSL Enabled).....	92
8 Managing Messages.....	96
8.1 Configuring RabbitMQ Dead Letter Messages.....	96
8.2 Configuring RabbitMQ Message Acknowledgment.....	98
8.3 Configuring RabbitMQ Message Prefetch.....	99
9 Advanced Features.....	101
9.1 Configuring RabbitMQ Persistence.....	101
9.2 Configuring RabbitMQ TTL.....	107
10 Managing Instances.....	110
10.1 Viewing and Modifying Basic Information of a RabbitMQ Instance.....	110
10.2 Viewing RabbitMQ Client Connection Addresses.....	112
10.3 Viewing RabbitMQ Background Tasks.....	114
10.4 Managing RabbitMQ Instance Tags.....	115
10.5 Configuring RabbitMQ Recycling Policies.....	116
10.6 Resetting the RabbitMQ Instance Password.....	118
10.7 Enabling RabbitMQ Plug-ins.....	119
10.8 Using the rabbitmq_tracing Plug-in.....	121
10.9 Exporting the RabbitMQ Instance List.....	126
10.10 Deleting a RabbitMQ Instance.....	127
10.11 Logging In to RabbitMQ Management UI.....	128
11 Modifying Instance Specifications.....	130
11.1 Modifying RabbitMQ Instance Specifications.....	130
12 Migrating RabbitMQ Services.....	135
13 Applying for Increasing RabbitMQ Quotas.....	140
14 Viewing Metrics and Configuring Alarms.....	142

14.1 Viewing RabbitMQ Metrics.....	142
14.2 RabbitMQ Metrics.....	143
14.3 Configuring RabbitMQ Alarms.....	152
15 Viewing RabbitMQ Audit Logs.....	156
16 FAQs.....	159
16.1 Instances.....	159
16.1.1 What RabbitMQ Version Does DMS for RabbitMQ Use?.....	159
16.1.2 What SSL Version Does DMS for RabbitMQ Use?.....	159
16.1.3 Why Can't I View the Subnet and Security Group Information During Instance Creation?.....	159
16.1.4 How Are Requests Evenly Distributed to Each VM of a Cluster RabbitMQ Instance?.....	159
16.1.5 Do Queues Inside a Cluster RabbitMQ Instance Have Any Redundancy Backup?.....	159
16.1.6 Does DMS for RabbitMQ Support Data Persistence? How Do I Perform Scheduled Data Backups?.....	160
16.1.7 How Do I Obtain the Certificate After SSL Has Been Enabled?.....	160
16.1.8 Can I Change the SSL Setting of a RabbitMQ Instance?.....	160
16.1.9 Can RabbitMQ Instances Be Scaled Up?.....	160
16.1.10 Does RabbitMQ Support Two-Way Authentication?.....	160
16.1.11 Does DMS for RabbitMQ Support CPU and Memory Upgrades?.....	160
16.1.12 How Do I Disable the RabbitMQ Management UI?.....	160
16.1.13 Can I Change the AZ for an Instance?.....	160
16.1.14 How Do I Obtain the Region ID?.....	162
16.1.15 Why Can't I Select Two AZs?.....	162
16.1.16 How to Change Single-node RabbitMQ Instances to Cluster Ones?.....	162
16.1.17 Can I Change the VPC and Subnet After a RabbitMQ Instance Is Created?.....	162
16.1.18 Does Specification Modification Affect Services?.....	162
16.2 Connections.....	164
16.2.1 How Do I Configure a Security Group?.....	164
16.2.2 Why Does a Client Fail to Connect to a RabbitMQ Instance?.....	165
16.2.3 Does DMS for RabbitMQ Support Public Access?.....	166
16.2.4 Does DMS for RabbitMQ Support Cross-Region Deployment?.....	166
16.2.5 Do RabbitMQ Instances Support Cross-VPC Access?.....	167
16.2.6 Do RabbitMQ Instances Support Cross-Subnet Access?.....	167
16.2.7 What Should I Do If I Fail to Access a RabbitMQ Instance with SSL Encryption?.....	167
16.2.8 Can I Access a RabbitMQ Instance Using DNAT?.....	167
16.2.9 Why Can't I Open the Management Web UI?.....	167
16.2.10 Can a Client Connect to Multiple Virtual Hosts of a RabbitMQ Instance?.....	167
16.2.11 Why Does a RabbitMQ Cluster Have Only One Connection Address?.....	168
16.2.12 Do RabbitMQ Instances Support the Ping Command?.....	168
16.3 Messages.....	168
16.3.1 Does DMS for RabbitMQ Support Delayed Message Delivery?.....	168
16.3.2 How Does Message Accumulation Affect Services? What Can I Do?.....	169
16.3.3 How Long Are Messages Be Retained?.....	171

16.3.4 Where Is Message Creation Time Set?..... 171

16.3.5 What Is the Maximum Size of a Message that Can be Created?..... 171

16.4 Monitoring & Alarm..... 171

16.4.1 Why Can't I View the Monitoring Data of a RabbitMQ Instance?..... 171

16.4.2 What Should I Do If the Number of Channels Keeps Rising?..... 172

A Change History..... 173

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 alarm 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
Cluster RabbitMQ instances provide quorum queues, which can be used to replicate queue data between RabbitMQ nodes, ensuring that queues can still run when a node breaks down.
- 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 3.8.35.

- **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 are recorded and can be audited. Messages can be encrypted before transmission.
In addition to SSL, 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.

For synchronization links that require real-time results, Remote Procedure Call (RPC) is recommended.

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

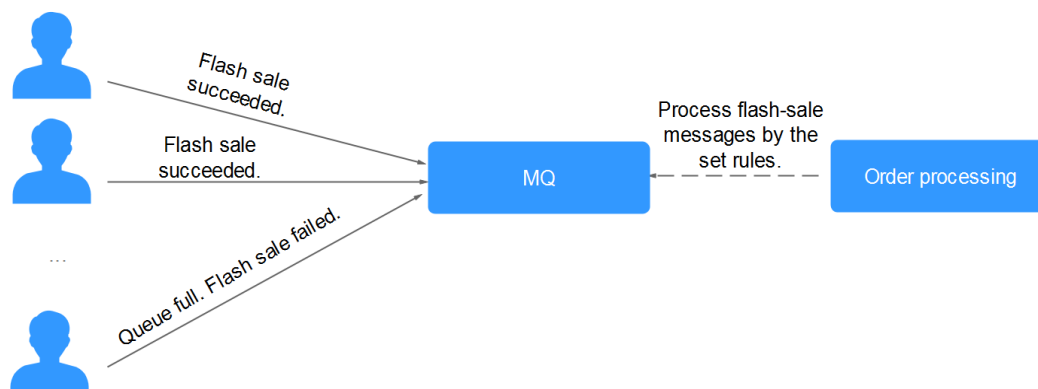


Traffic Control

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



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.

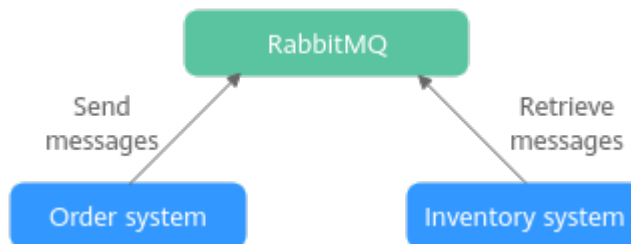
Figure 1-4 Closely coupled systems



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

Normally, there is only one broker. If the broker fails, queues on it will become unavailable.

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

Quorum queues are available since RabbitMQ 3.8. Quorum queues can be used to replicate queue data, ensuring that queues can still run if a broker breaks down.

1.4 Specifications

RabbitMQ Instance Specifications

RabbitMQ instances are compatible with RabbitMQ 3.8.35. [Table 1-1](#) lists the specifications of single-node and cluster RabbitMQ instances.

Table 1-1 Specifications of cluster RabbitMQ instances

Flavor	Brokers	Storage Space (GB)	Reference TPS	Maximum Consumers per Broker	Recommended Queues per Broker	Maximum Connections per Broker
rabbitmq.2u4g.cluster	3	300–90,000	3,000	4,000	100	1,000
	5	500–150,000	5,000	4,000	100	1,000
	7	700–210,000	7,000	4,000	100	1,000
rabbitmq.4u8g.cluster	3	300–90,000	6,000	8,000	200	2,000
	5	500–150,000	10,000	8,000	200	2,000
	7	700–210,000	14,000	8,000	200	2,000
rabbitmq.8u16g.cluster	3	300–90,000	12,000	16,000	400	4,000
	5	500–150,000	20,000	16,000	400	4,000
	7	700–210,000	28,000	16,000	400	4,000
rabbitmq.12u24g.cluster	3	300–90,000	24,000	24,000	600	6,000
	5	500–150,000	40,000	24,000	600	6,000
	7	700–210,000	56,000	24,000	600	6,000
rabbitmq.16u32g.cluster	3	300–90,000	48,000	32,000	800	8,000
	5	500–150,000	80,000	32,000	800	8,000
	7	700–210,000	112,000	32,000	800	8,000
rabbitmq.24u48g.cluster	3	300–90,000	60,000	40,000	1,000	10,000
	5	500–150,000	100,000	40,000	1,000	10,000
	7	700–210,000	140,000	40,000	1,000	10,000
rabbitmq.32u64g.cluster	3	300–90,000	72,000	40,000	1,000	10,000

Flavor	Brokers	Storage Space (GB)	Reference TPS	Maximum Consumers per Broker	Recommended Queues per Broker	Maximum Connections per Broker
	5	500–150,000	120,000	40,000	1,000	10,000
	7	700–210,000	168,000	40,000	1,000	10,000

 **NOTE**

- In the preceding tables, TPS (of production and consumption) 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.
- Single-node instances can be used for testing. Do not use them for message production. Single-node flavors are not yet available.
- The maximum number of connections per broker of RabbitMQ 3.x.x instances is listed in the table above. The maximum number of connections of the entire cluster is 30,000.

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 RabbitMQ, Kafka, and RocketMQ

Table 1-2 Functions

Feature	RocketMQ	Kafka	RabbitMQ
Priority queue	Not supported	Not supported	Supported. It is recommended that the priority be set to 0–10.
Delayed queue	Supported	Not supported	Supported
Dead letter queue	Supported	Not supported	Supported
Message retry	Supported	Not supported	Not supported
Retrieval mode	Pull-based and push-based	Pull-based	Pull-based and push-based
Message broadcasting	Supported	Supported	Supported
Message tracking	Supported	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	Supported	Supports higher accumulation performance than RabbitMQ thanks to high throughput.	Supported
Persistence	Supported	Supported	Supported
Message tracing	Supported	Not supported	Supported by the firehose feature or the rabbitmq_tracing plugin. However, rabbitmq_tracing reduces performance and should be used only for troubleshooting.

Feature	RocketMQ	Kafka	RabbitMQ
Message filtering	Supported	Supported	Not supported, but can be encapsulated.
Multi-tenancy	Supported	Supported	Supported
Multi-protocol	Compatible with RocketMQ.	Only supports Apache Kafka.	RabbitMQ is based on AMQP.
Multi-language	Supports clients in multiple programming languages.	Kafka is written in Scala and Java and supports clients in multiple programming languages.	Supports clients in multiple programming languages.
Throttling	RocketMQ 5.x supports traffic control based on instance specifications.	Supports throttling on producer or consumer clients, users, and topics.	Supports credit-based throttling on producers, a mechanism that triggers protection from within.
Ordered message delivery	Message order is maintained within a queue.	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 authentication.	Supports SSL and SASL authentication and read/write permissions control.	Supports SSL authentication.
Transactional messages	Supported	Supported	Supported

1.6 Related Services

- Elastic Cloud Server (ECS)
An ECS is a basic computing unit that consists of vCPUs, memory, OS, and EVS disks. RabbitMQ instances run on ECSs. A broker corresponds to an ECS.
- Elastic Volume Service (EVS)
EVS provides block storage services for ECSs. All RabbitMQ data, such as messages, metadata, and logs, is stored in EVS disks.

- 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 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 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 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.

- Elastic IP (EIP)
The EIP service provides independent public IP addresses and bandwidth for Internet access. RabbitMQ instances bound with EIPs can be accessed over public networks.
- Tag Management Service (TMS)
TMS is a visualized service for fast and unified cross-region tagging and categorization of cloud services.
Tags facilitate RabbitMQ instance identification and management.

1.7 Notes and Constraints

This section describes the notes and constraints on Distributed Message Service (DMS) for RabbitMQ. Use your RabbitMQ instances as prescribed to avoid program exceptions.

NOTICE

Any instability caused by ignorance of the notes and constraints is not covered by the SLA.

Instance

Table 1-3 Notes and constraints

Item	Constraint
Version	Server version: 3.8.35
Number of connections	The allowed number of connections differs by instance specifications and mode (single-node or cluster). For details, see Specifications .

Item	Constraint
Channels	Number of channels that can be created for a single connection: ≤ 2047
Memory high watermark	$\leq 40\%$ If the memory usage exceeds 40%, the high memory watermark may be 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.
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. 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.
RabbitMQ plug-ins	RabbitMQ plug-ins can be used for testing and service migration. Do not use them for production. Reliability issues caused from using plug-ins are not within commitments on SLAs.
VPC, subnet, and AZ	After an instance is created, its VPC, subnet, and AZ cannot be modified.
Storage space per broker	<ul style="list-style-type: none">• The storage space can be expanded but cannot be reduced.• You can expand the storage space 20 times.
Broker quantity	<ul style="list-style-type: none">• The broker quantity can be increased but cannot be decreased for a cluster instance.• Services may temporarily stutter during the broker increase. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.• This function is not available for single-node instances.

Item	Constraint
ping command	<ul style="list-style-type: none">Single-node instances support the ping command with both private and public connection addresses.Cluster instances only support the ping command with private connection addresses.

Virtual Host

Table 1-4 Constraint

Item	Constraint
Deleting a virtual host	The default virtual host created in instance creation cannot be deleted.
Vhost	<ul style="list-style-type: none">When a virtual host name of a RabbitMQ 3.x.x instance starts with a special character, such as a period (.), the monitoring data may not be displayed. When a virtual host name contains special characters such as percent (%), vertical bar (), or slash (/), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_) instead. For example, virtual host Vhost.1% 2_3/ is displayed as Vhost.1_1_2_3_ in monitoring.On Cloud Eye, a special character (.) is counted as five characters. If the name of a virtual host of a RabbitMQ 3.x.x instance contains periods (.) and the counted total length exceeds 256 characters, monitoring data may not be displayed.

Exchange

Table 1-5 Notes and exchanges

Item	Constraint
Default exchange	For RabbitMQ 3.x.x instances, seven exchanges are created by default after virtual host creation. These exchanges include (AMQP default), amq.direct, amq.fanout, amq.headers, amq.match, amq.rabbitmq.trace, and amq.topic.
Binding an exchange	<ul style="list-style-type: none">For RabbitMQ 3.x.x, the exchange (AMQP default) cannot be bound with any exchange.Internal exchanges can only be bound with exchanges and not queues.

Item	Constraint
Deleting an exchange	For RabbitMQ 3.x.x, the default exchange cannot be deleted.

Queue

Table 1-6 Notes and constraints

Item	Constraint
Binding a queue	<ul style="list-style-type: none">For RabbitMQ 3.x.x, the exchange (AMQP default) cannot be bound with any queue.Internal exchanges can only be bound with exchanges and not queues.
Lazy queues	Available for RabbitMQ 3.8.35 and later.
Quorum queues	Available for RabbitMQ 3.8.35 and later.
Single active consumer	Available for RabbitMQ 3.8.35 and later.
Queue	<ul style="list-style-type: none">When a queue name of a RabbitMQ 3.x.x instance starts with a special character, such as a period (.), the monitoring data may not be displayed. When a queue name contains special characters such as percent (%), vertical bar (), and slash (/), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_) instead. For example, queue Queue.1%1 2_3/ is displayed as Queue.1_1_2_3_ in monitoring.On Cloud Eye, a special character (.) is counted as five characters. If the name of a queue of a RabbitMQ 3.x.x instance contains periods (.) and the counted total length exceeds 256 characters, monitoring data may not be displayed.

Message

Table 1-7 Notes and constraints

Item	Constraint
Message size	<ul style="list-style-type: none">≤ 50 MB per messageDo not send a message larger than 50 MB. Otherwise, the message will fail to be created.

1.8 Basic Concepts

Cloud 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.

Virtual Host

Virtual hosts provide logical separation of exchanges, queues, and bindings. Different applications run on different virtual hosts without interfering with each other.

Exchange

Exchanges receive and assign messages. Producers send messages to exchanges first, rather than directly to queues. Exchanges route messages to one or more queues based on routing keys. If there are no matching queues, the messages are discarded. To learn about exchange types, see [Exchanges](#).

1.9 Exchanges

Direct, fanout, topic, and headers exchanges are available.

Direct Exchange

How It Works

1. A queue is bound to a direct exchange with a routing key.
2. The direct exchange routes a received message to the bound queue whose routing key is matched.

Routing

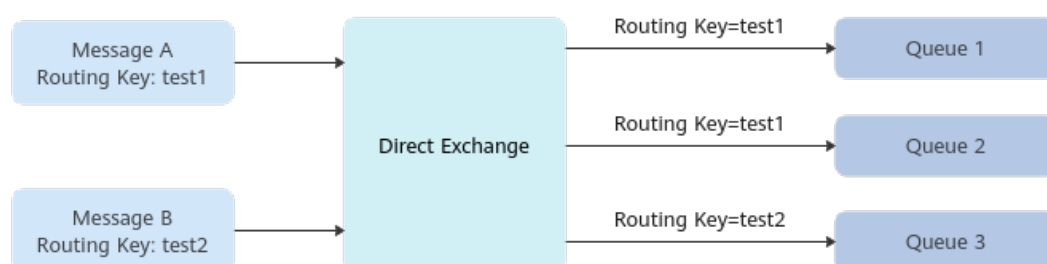
Based on a routing key matching

Scenario

Unicast routing

Example

Figure 1-6 Example direct exchange



As shown in this chart, Message A will be sent to Queue 1 and Queue 2. Message B will be sent to Queue 3.

Fanout Exchange

How It Works

A fanout exchange bound with multiple queues routes received messages to each queue. Fanout exchanges forward messages faster than other exchanges.

Routing

The fanout exchange delivers messages to all bound queues.

Scenario

Broadcast routing

Example

Figure 1-7 Example fanout exchange



As shown in this chart, Message A will be sent to Queue 1 and Queue 2.

Topic Exchange

How It Works

1. A queue is bound to a topic exchange with a routing key that includes a wildcard.
2. The topic exchange routes a received message to the queue if the message's routing key wildcard is matched.

Supported wildcards are stars (*) and hashes (#). Separate wildcards and words by periods (.), for example, **test.#**.

- * matches one word.
- # matches zero or more words.

Routing

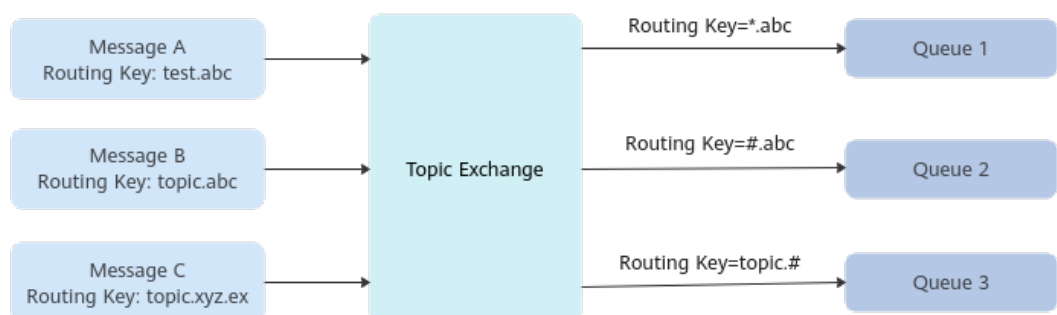
Based on a routing key wildcard matching

Scenario

Multicast routing

Example

Figure 1-8 Example topic exchange



As shown in this chart, Message A will be sent to Queue 1 and Queue 2, Message B to Queue 1, Queue 2, and Queue 3, and Message C to Queue 3.

Headers Exchange

How It Works

1. A queue is bound to a headers exchange with a binding expressed in a key-value pair.
2. The headers exchange routes a message to the queue if the binding matches the message's key-value header.

The matching algorithm uses a specific binding key-value pair, which is **x-match**. Values:

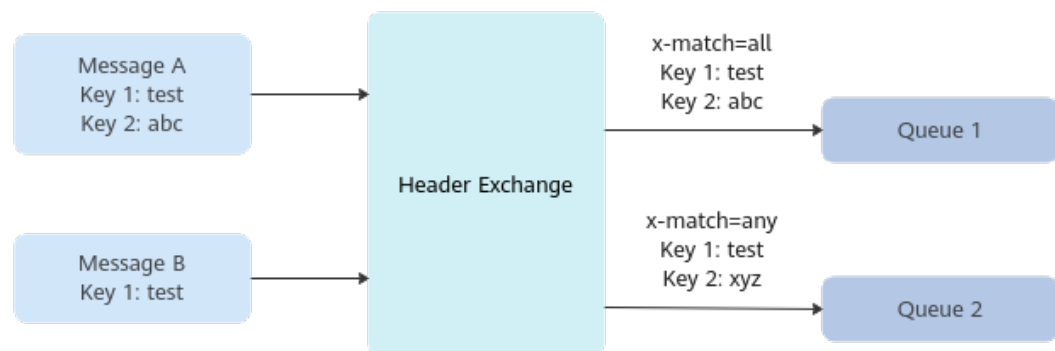
- **all**: Messages are routed only when all header pairs match.
- **any**: Messages are routed when any header pair matches.

Routing

Based on matching between key-value pairs in the message headers and the binding (a key-value pair)

Example

Figure 1-9 Example headers exchange



As shown in this chart, Message A will be sent to Queue 1 and Queue 2, and Message B to Queue 2.

1.10 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.

IAM is free of charge. You pay only for the resources you use. For more information, see [IAM Service Overview](#).

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 fine-grained authorization strategy that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization for more secure access control. For example, you can grant DMS for RabbitMQ users only the permissions for managing a certain type of DMS for RabbitMQ 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 1-8 lists all the system-defined roles and policies supported by DMS for RabbitMQ.

Table 1-8 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

Role/Policy Name	Description	Type	Dependency
DMS ReadOnlyAccess	Read-only permissions for DMS. Users granted these permissions can only view DMS data.	System-defined policy	None
DMS VPCAccess	VPC operation permissions to assign to DMS agencies.	System-defined policy	None
DMS KMSAccess	KMS operation permissions to assign to DMS agencies.	System-defined policy	None
DMS ELBAccess	ELB operation permissions to assign to DMS agencies.	System-defined policy	None
DMS VPCEndpointAccess	VPC endpoint operation permissions to assign to DMS agencies.	System-defined policy	None
DMSAgencyCheckAccessPolicy	IAM operation permissions to assign to DMS agencies.	System-defined policy	None
DMS BSSAccess	BSS operation permissions to assign to DMS agencies.	System-defined policy	None
DMS OBSAccess	OBS operation permissions to assign to DMS agencies.	System-defined policy	None

Table 1-9 lists the common operations supported by each DMS for RabbitMQ system policy or role. Select the policies or roles as required.

Table 1-9 Common operations supported by system-defined policies

Operation	DMS FullAccess	DMS UserAccess	DMS ReadOnlyAccess
Creating an instance	√	×	×
Modifying instances	√	×	×
Deleting instances	√	×	×

Operation	DMS FullAccess	DMS UserAccess	DMS ReadOnlyAccess
Modifying instance specifications	√	×	×
Querying instance information	√	√	√

Only system-defined policies listed in [Table 1-9](#) have permissions to perform common operations on DMS for RabbitMQ.

Helpful Links

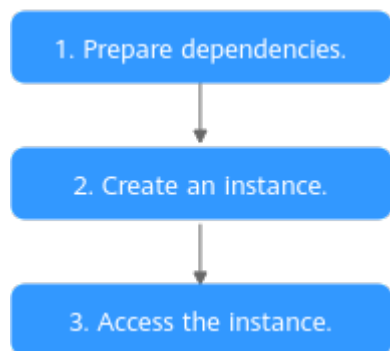
- [What Is IAM?](#)
- [Creating a User and Granting DMS for RabbitMQ Permissions](#)
- [Permissions Policies and Supported Actions](#)

2 Getting Started

2.1 Getting Started with RabbitMQ for Message Production and Consumption

This section takes an example to get you started with DMS for RabbitMQ. The example creates a RabbitMQ instance with SSL disabled, and accesses it over a private network on a client within a VPC. As a result, messages can be produced and consumed.

Figure 2-1 Procedure for using DMS for RabbitMQ



1. **Step 1: Preparations**

A RabbitMQ instance runs in a Virtual Private Cloud (VPC). Before creating a RabbitMQ instance, ensure that a VPC is available.

2. **Step 2: Creating a RabbitMQ Instance**

You can select the specifications and quantity when creating a RabbitMQ instance.

3. **Step 3: Accessing an Instance for Message Production and Consumption**

A client connects to the instance with SSL disabled using the demo provided by RabbitMQ.

Step 1: Preparations

Step 1 Grant RabbitMQ instance permissions.

To achieve fine-grained management of your cloud resources, create Identity and Access Management (IAM) user groups and users and grant specified permissions to the users. For more information, see [Creating an IAM User and Granting DMS for RabbitMQ Permissions](#).

Step 2 Create a VPC and subnet.

Before creating a RabbitMQ instance, ensure that a VPC and a subnet are available. For details about how to create a VPC and subnet, see [Creating a VPC with a Subnet](#).

The VPC **must** be created in the **same** region as the RabbitMQ instance.

Step 3 Create a security group and add security group rules.

Before creating a RabbitMQ instance, ensure that a security group is available. For details about how to create a security group, see [Creating a Security Group](#).

The security group **must** be created in the **same** region as the RabbitMQ instance.

To connect to RabbitMQ instances, add the security group rules described in [Table 2-1](#). Other rules can be added based on site requirements.

Table 2-1 Security group rules


Direction	Protocol	Port	Source address	Description
Inbound	TCP	5672	0.0.0.0/0	Accessing a RabbitMQ instance (SSL disabled)

NOTE

After a security group is created, it has a default inbound rule that allows communication among ECSs within the security group and a default outbound rule that allows all outbound traffic. If you access your RabbitMQ instance over a private network within a VPC, you do not need to add the rules described in [Table 2-1](#).

Step 4 Construct a client for message production and consumption.

This section uses a Linux Elastic Cloud Server (ECS) as the client. Before creating a RabbitMQ instance, create an ECS with elastic IPs (EIPs), install JDK, and configure the environment variables.

1. Log in to the console, click  in the upper left corner, click **Elastic Cloud Server** under **Computing**, and then create an ECS.
For details about how to create an ECS, see [Creating an ECS](#). If you already have an available ECS, skip this step.
2. Log in to the ECS as user **root**.
3. Install Java JDK and configure the environment variables **JAVA_HOME** and **PATH**.

- a. Download a JDK.

 **NOTE**

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

- b. Decompress the JDK.

```
tar -zxvf jdk-8u321-linux-x64.tar.gz
```

Change **jdk-8u321-linux-x64.tar.gz** to your JDK version.

- c. Open the **.bash_profile** file.

```
vim ~/.bash_profile
```

- d. Add the following content:

```
export JAVA_HOME=/opt/java/jdk1.8.0_321
export PATH=$JAVA_HOME/bin:$PATH
```

Change **/opt/java/jdk1.8.0_321** to the path where you install JDK.

- e. Press **Esc**. Enter the following line and press **Enter**. Save the **.bash_profile** file and exit.

```
:wq
```

- f. Run the following command to make the change take effect:

```
source ~/.bash_profile
```

- g. Check whether the JDK is installed.

```
java -version
```

If the following message is returned, the JDK is installed.

```
java version "1.8.0_321"
```

----End

Step 2: Creating a RabbitMQ Instance

Step 1 Log in to the RabbitMQ console, and click **Buy RabbitMQ Instance** in the upper right corner.

Step 2 On the **Quick Config** tab page, set basic instance configurations. [Table 2-2](#) lists the configuration details.

Table 2-2 Setting basic instance configurations

Parameter	Description
Billing Mode	Select Pay-per-use , which is a postpaid mode. You can pay after using the service, and will be billed for your usage duration. The fees are calculated in seconds and settled by hour.
Region	DMS for RabbitMQ in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access. Select MY-Kuala Lumpur .

Parameter	Description
AZ	An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. Select AZ1 .

Step 3 Set the instance specifications and storage space. For details, see [Table 2-3](#).

Table 2-3 Setting the instance specifications and storage space

Parameter	Description
Bundle	Select Test : Single-node instances with mini specs, saving resources and costs for testing or debugging. Note that high availability is unavailable.
Type	Select the disk type as required. The disk type is fixed once the instance is created. Select Ultra-high I/O .
Capacity	Specify the disk size as required. Total storage space of an instance = Storage space per broker × Number of brokers Select 100 GB.

Step 4 Configure the instance network. For details, see [Table 2-4](#).

Table 2-4 Configuring instance network

Parameter	Description
VPC	After the RabbitMQ instance is created, its VPC cannot be changed. Select the VPC prepared in Step 2 .
Subnet	After the RabbitMQ instance is created, its subnet cannot be changed. Select the subnet prepared in Step 2 .
Security Group	Select the security group prepared in Step 3 .

Step 5 Set the instance access mode. Retain the default setting.

Step 6 Set the instance authentication method. For details, see [Table 2-5](#).

Table 2-5 Configuring the instance authentication method

Parameter	Description
RabbitMQ Authentication Username	Enter the username used for accessing the instance. A username should contain 4 to 64 characters, start with a letter, and contain only letters, digits, hyphens (-), and underscores (_). Enter "test".
Password	Enter the password used for accessing the instance. A password must meet the following requirements: <ul style="list-style-type: none">Contains 8 to 32 characters.Contains at least three types of the following characters: uppercase letters, lowercase letters, digits, and special characters `~!@#\$%^&*()-_+=\ []{};:'",<.>?` and spaces, and cannot start with a hyphen (-).Cannot be the username spelled forwards or backwards.

Step 7 Configure **Advanced Settings**. For details, see [Table 2-6](#). Retain default settings for other parameters.

Table 2-6 Configuring instance advanced settings

Parameter	Description
Instance Name	You can customize a name that complies with the rules: 4–64 characters; starts with a letter; can contain only letters, digits, hyphens (-), and underscores (_). Enter "rabbitmq-test".
Enterprise Project	This parameter is for enterprise users. An enterprise project manages project resources in groups. Enterprise projects are logically isolated. Select default .

Step 8 Click **Confirm**.

Step 9 Confirm the instance information and then submit the request.

Step 10 Return to the instance list and check whether the instance has been created.

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

- If the instance is created successfully, its status changes to **Running**.
- If the instance is in the **Creation failed** state, delete it and try purchasing another one. If the instance purchase fails again, contact customer service.

Step 11 After the instance is created, click its name to go to the instance details page.

Step 12 In the **Connection** area, view and record the connection address.

----End

Step 3: Accessing an Instance for Message Production and Consumption

Step 1 Go to the **root** directory on the ECS and **download** the sample project code **RabbitMQ-Tutorial.zip**.

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

NOTE

/root is the path for storing the sample project code. Change it to the actual path if needed.

Step 2 Run the following command to decompress **RabbitMQ-Tutorial.zip**:

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

Step 3 Run the following command to navigate to the **RabbitMQ-Tutorial** directory, which contains the precompiled JAR file:

```
cd RabbitMQ-Tutorial
```

Step 4 Produce messages using the sample project.

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

Description:

- *host*: connection address obtained in the **instance creation**.
- *port*: port of the RabbitMQ instance. Enter **5672**.
- *user*: username set in **instance creation**.
- *password*: password set in **instance creation**.

Sample message production:

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

Step 5 Consume messages using the sample project.

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

Description:

- *host*: connection address obtained in the **instance creation**.
- *port*: port of the RabbitMQ instance. Enter **5672**.
- *user*: username set in **instance creation**.
- *password*: password set in **instance creation**.

Sample message consumption:

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

Press **Ctrl+C** to cancel.

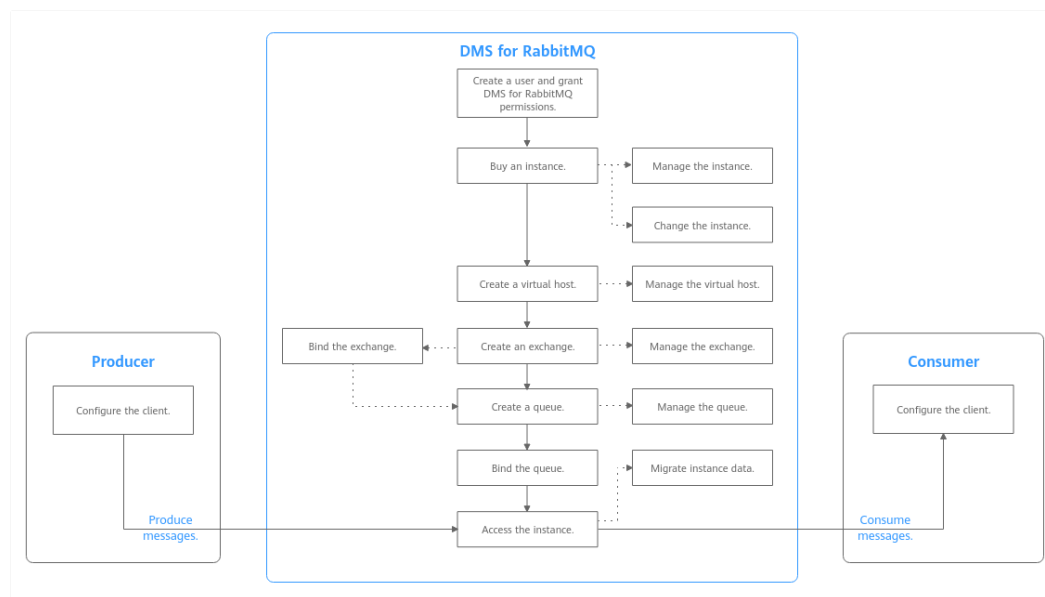
----**End**

3 Process of Using RabbitMQ

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

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

Figure 3-1 Process of using RabbitMQ



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

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

4. **Creating a RabbitMQ Exchange**

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

5. **Creating a RabbitMQ Queue**

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

6. **Binding a RabbitMQ Queue**

Exchanges route messages to queues based on routing keys.

7. **Accessing a RabbitMQ Instance**

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

4 Permissions Management

4.1 Creating an IAM User and Granting DMS for RabbitMQ Permissions

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

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

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

This section describes the procedure for granting user permissions. [Figure 4-1](#) shows the process flow.

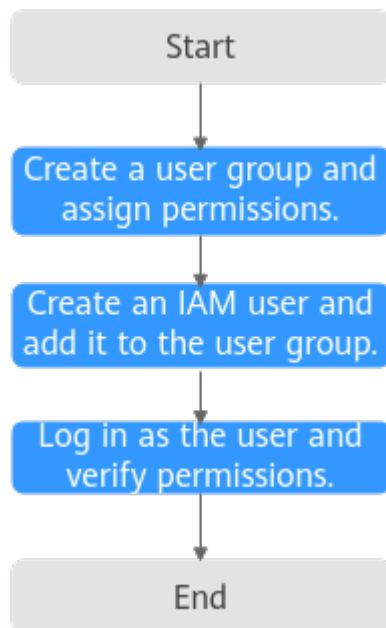
Prerequisites

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

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

Process Flow

Figure 4-1 Process of granting DMS for RabbitMQ permissions



1. For the following example, **create a user group on the IAM console**, and assign the **DMS ReadOnlyAccess** policy to the group.
2. **Create an IAM user and add it to the created user group.**
3. **Log in as the IAM user** and verify permissions.

In the authorized region, perform the following operations:

- Choose **Service List > Distributed Message Service for RabbitMQ**. Then click **Buy Instance** on the console of DMS for RabbitMQ. If a message appears indicating that you have insufficient permissions to perform the operation, the **DMS ReadOnlyAccess** policy is in effect.
- Choose **Service List > Elastic Volume Service**. If a message appears indicating that you have insufficient permissions to access the service, the **DMS ReadOnlyAccess** policy is in effect.
- Choose **Service List > Distributed Message Service for RabbitMQ**. The RabbitMQ console is displayed. If a list of RabbitMQ instances are displayed, the **DMS ReadOnlyAccess** policy is in effect.

Example Custom Policies

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

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

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

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

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

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

- Example 2: Grant permission to deny instance deletion.

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

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

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

DMS for RabbitMQ Resources

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

Table 4-1 DMS for RabbitMQ resources and their paths

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

DMS for RabbitMQ Request Conditions

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

DMS for RabbitMQ has a group of predefined condition keys that can be used in IAM. For example, to define an "Allow" permission, you can use the condition key **dms:ssl** to check whether SSL is enabled for a RabbitMQ instance. The following table lists the predefined condition keys of DMS for RabbitMQ.

Table 4-2 Predefined condition keys of DMS for RabbitMQ

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

5 Buying a RabbitMQ Instance

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

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

Preparing Required Resources

Dependency resources listed in [Table 5-1](#) have been prepared.

Table 5-1 RabbitMQ instance dependencies


Resource Name	Requirement	Reference
VPC and subnet	You need to configure a VPC and subnet for the RabbitMQ instance as required. You can use the current account's existing VPC and subnet, or create new ones. Note: VPCs must be created in the same region as the RabbitMQ instance.	For details about how to create a VPC and subnet, see the <i>Virtual Private Cloud User Guide</i> .
Security group	Different RabbitMQ instances can use the same or different security groups. The security group must be in the same region as the RabbitMQ instance. Before accessing a RabbitMQ instance, configure security groups based on the access mode. For details, see Table 7-2 .	For details about how to create a security group and configure security group rules, see the <i>Virtual Private Cloud User Guide</i> .

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

Quickly Configuring a RabbitMQ Instance

Step 1 Log in to the console.

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

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

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

Step 5 Set basic instance configurations on the **Quick Config** tab page.

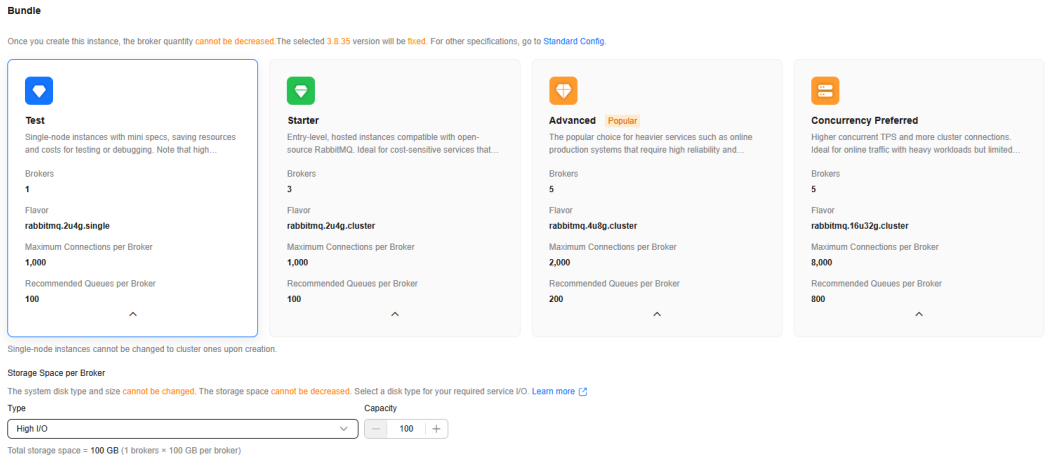
Table 5-2 Basic instance configuration parameters

Parameter	Description
Billing Mode	<ul style="list-style-type: none">• Pay-per-use is a postpaid mode. You can pay after using the service, and will be billed for your usage duration. The fees are calculated in seconds and settled by hour.
Region	DMS for RocketMQ instances in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access.
AZ	<p>An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network.</p> <p>Select one AZ or at least three AZs. The AZ setting is fixed once the instance is created.</p>

Step 6 Set the bundle.

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

Figure 5-1 Bundle




Step 7 Set the network.

Table 5-3 Instance network parameters

Parameter	Description
VPC	Select a created or shared VPC. A VPC provides an isolated virtual network for your RabbitMQ instances. You can configure and manage the network as required. You can click Manage VPC on the right to go to the VPC console and view or create a VPC. After the RabbitMQ instance is created, its VPC cannot be changed.
Subnet	Select a created subnet. After the RabbitMQ instance is created, its subnet cannot be changed.
Security Group	Select a created security group. A security group is a set of rules for accessing a RabbitMQ instance. You can click Create Security Group on the right to go to the security group page on the network console to view or create a security group. Before accessing a RabbitMQ instance on the client, configure security group rules based on the access mode. For details about security group rules, see Table 7-2 .

Step 8 Set the instance access mode.

Table 5-4 Instance access mode parameters

Parameter	Description
Public Network Access	<p>Clients can access RabbitMQ instances with public access enabled through elastic IP addresses (EIPs).</p> <p>Enabling public access requires EIPs. If EIPs are insufficient, click Create Elastic IP to create EIPs. Then, return to the RabbitMQ console and click  next to Public IP Addresses to refresh the elastic IP address list.</p> <p>NOTE</p> <ul style="list-style-type: none">• In comparison with intra-VPC access, enabling public access increases access latency and might lead to packet loss and jitter. Therefore, you are advised to enable public access only during the service development and testing phases.• If you manually unbind or delete an EIP on the VPC console, the public access function of the corresponding RabbitMQ instance is automatically disabled.
Encryption Mode	<p>Enabling SSL secures data transmission with encryption.</p> <p>Once the instance is created, SSL cannot be manually configured.</p>

Step 9 Set the instance authentication.**Table 5-5** Instance authentication parameters

Parameter	Description
RabbitMQ Authentication Username	<p>Enter the username used for accessing the instance.</p> <p>A username should contain 4 to 64 characters, start with a letter, and contain only letters, digits, hyphens (-), and underscores (_).</p>
Password	<p>Enter the password used for accessing the instance.</p> <p>A password must meet the following requirements:</p> <ul style="list-style-type: none">• Contains 8 to 32 characters.• Contains at least three types of the following characters: uppercase letters, lowercase letters, digits, and special characters `~! @#\$%^&*()-_+=\ [{ }];:","<.>?` and spaces, and cannot start with a hyphen (-).• Cannot be the username spelled forwards or backwards.

Step 10 Configure advanced settings.

Table 5-6 Advanced setting parameters

Parameter	Description
Instance Name	You can customize a name that complies with the rules: 4–64 characters; starts with a letter; can contain only letters, digits, hyphens (-), and underscores (_).
Enterprise Project	Available for enterprise users. Enterprise projects facilitate project-level management and grouping of cloud resources and users. The default project is default .
Tags	<p>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).</p> <p>If your organization has configured tag policies for DMS for RabbitMQ, add tags to RabbitMQ instances based on the tag policies. If a tag does not comply with the tag policies, RabbitMQ instance creation may fail. Contact your organization administrator to learn more about tag policies.</p> <ul style="list-style-type: none">• If you have created predefined tags, select a predefined pair of tag key and value. You can click Create predefined tags to go to the Tag Management Service (TMS) console and view or create tags.• You can also create new tags by entering Tag key and Tag value. <p>Up to 20 tags can be added to each RabbitMQ instance. For details about the requirements on tags, see Managing RabbitMQ Instance Tags.</p>
Description	Set the description of the instance for up to 1024 characters.

Step 11 Click **Confirm**.

Step 12 Confirm the instance information, and click **Submit**.

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


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


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

----End

Customizing a RabbitMQ Instance (3.x.x)

Step 1 Log in to the console.

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

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

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

Step 5 Set basic instance configurations on the **Standard Config** tab page.

Table 5-7 Basic instance configuration parameters

Parameter	Description
Billing Mode	<ul style="list-style-type: none">Pay-per-use is a postpaid mode. You can pay after using the service, and will be billed for your usage duration. The fees are calculated in seconds and settled by hour.
Region	DMS for RocketMQ instances in different regions cannot communicate with each other over an intranet. Select a nearest location for low latency and fast access.
AZ	An AZ is a physical region where resources use independent power supply and networks. AZs are physically isolated but interconnected through an internal network. Select one AZ or at least three AZs. The AZ setting is fixed once the instance is created.

Step 6 Configure the following instance parameters:

Table 5-8 Instance specifications parameters

Parameter	Description
Version	RabbitMQ version. Select 3.8.35 . The version is fixed once the instance is created.
Architecture	Single-node or Cluster are available. <ul style="list-style-type: none">Single-node: There is only one RabbitMQ broker.Cluster: There are multiple RabbitMQ brokers, achieving highly reliable message storage.
Broker Flavor	Select a broker flavor as required. Learn more about Specifications . To ensure service stability and reliability, DMS for RabbitMQ sets the default memory high watermark to 40%. Publishers will be blocked if the memory usage exceeds 40%. To avoid reaching the high watermark, retrieve messages stacked in the queue in time.

Parameter	Description
Brokers	Specify the broker quantity.
Type	Select the disk type for RabbitMQ data storage. The disk type is fixed once the instance is created.
Capacity	Specify the disk size for RabbitMQ data storage.

Figure 5-2 V3.8.35 instance specifications

Version

3.8.35

Software version of the instance. This setting is fixed once the instance is created. Ideally, use server and client with the same version.

Instance Type

Default

Architecture

Single-node

Cluster

Single-node instances cannot be changed to cluster ones upon creation.

Broker Flavor

Flavor Name	Maximum Connections per Broker	Recommended Queues per Broker
<input checked="" type="radio"/> rabbitmq.2u4g.single	1,000	100
<input type="radio"/> rabbitmq.4u8g.single	2,000	200
<input type="radio"/> rabbitmq.8u16g.single	4,000	400
<input type="radio"/> rabbitmq.16u32g.single	8,000	800
<input type="radio"/> rabbitmq.24u48g.single	10,000	1,000

Specifications Flavor Name: rabbitmq.2u4g.single | Maximum Connections per Broker: 1,000 | Recommended Queues per Broker: 100

Brokers

-

1

+

Storage Space per Broker

The system disk type and size cannot be changed. The storage space cannot be decreased. Select a disk type for your required service I/O. [Learn more](#)

Type

Capacity

Ultra-high I/O

-

100

+

Total storage space = 100 GB (1 brokers × 100 GB per broker)


Step 7 Set the network.

Table 5-9 Instance network parameters

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

Parameter	Description
Security Group	<p>Select a created security group.</p> <p>A security group is a set of rules for accessing a RabbitMQ instance. You can click Create Security Group on the right to go to the security group page on the network console to view or create a security group.</p> <p>Before accessing a RabbitMQ instance on the client, configure security group rules based on the access mode. For details about security group rules, see Table 7-2.</p>

Step 8 Set the instance access mode.**Table 5-10** Instance access mode parameters

Parameter	Description
Public Network Access	<p>Clients can access RabbitMQ instances with public access enabled through elastic IP addresses (EIPs).</p> <p>Enabling public access requires EIPs. If EIPs are insufficient, click Create Elastic IP to create EIPs. Then, return to the RabbitMQ console and click  next to Public IP Addresses to refresh the elastic IP address list.</p> <p>NOTE</p> <ul style="list-style-type: none">• In comparison with intra-VPC access, enabling public access increases access latency and might lead to packet loss and jitter. Therefore, you are advised to enable public access only during the service development and testing phases.• If you manually unbind or delete an EIP on the VPC console, the public access function of the corresponding RabbitMQ instance is automatically disabled.
Encryption Mode	<p>Enabling SSL secures data transmission with encryption.</p> <p>Once the instance is created, SSL cannot be manually configured.</p>

Step 9 Set the instance authentication.**Table 5-11** Instance authentication parameters

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

Parameter	Description
Password	<p>Enter the password used for accessing the instance.</p> <p>A password must meet the following requirements:</p> <ul style="list-style-type: none">• Contains 8 to 32 characters.• Contains at least three types of the following characters: uppercase letters, lowercase letters, digits, and special characters `~! @\$ %^&*()-_+=\ [{ }];:","<.>?` and spaces, and cannot start with a hyphen (-).• Cannot be the username spelled forwards or backwards.

Step 10 Configure advanced settings.**Table 5-12** Advanced setting parameters

Parameter	Description
Instance Name	You can customize a name that complies with the rules: 4–64 characters; starts with a letter; can contain only letters, digits, hyphens (-), and underscores (_).
Enterprise Project	<p>Available for enterprise users.</p> <p>Enterprise projects facilitate project-level management and grouping of cloud resources and users. The default project is default.</p>
Tags	<p>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).</p> <p>If your organization has configured tag policies for DMS for RabbitMQ, add tags to RabbitMQ instances based on the tag policies. If a tag does not comply with the tag policies, RabbitMQ instance creation may fail. Contact your organization administrator to learn more about tag policies.</p> <ul style="list-style-type: none">• If you have created predefined tags, select a predefined pair of tag key and value. You can click Create predefined tags to go to the Tag Management Service (TMS) console and view or create tags.• You can also create new tags by entering Tag key and Tag value. <p>Up to 20 tags can be added to each RabbitMQ instance. For details about the requirements on tags, see Managing RabbitMQ Instance Tags.</p>
Description	Set the description of the instance for up to 1024 characters.

Step 11 In **Summary** on the right, view the selected instance configuration.

Step 12 Click **Confirm**.

Step 13 Confirm the instance information, and click **Submit**.

Step 14 Return to the instance list and check whether the instance has been created.

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


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


----End

Purchasing a RabbitMQ Instance with Same Configurations

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

Step 1 Log in to the console.

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

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

Step 4 Select a target RabbitMQ instance and choose **More > Buy Another** in the **Operation** column.

Step 5 Adjust the automatically replicated parameter settings as required. For details, see [Customizing a RabbitMQ Instance \(3.x.x\)](#).

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

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

Step 6 In **Summary** on the right, view the selected instance configuration.

Step 7 Click **Confirm**.

Step 8 Confirm the instance information, and click **Submit**.

Step 9 Return to the instance list and check whether the instance has been created.

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

- If the instance is created successfully, its status changes to **Running**.

- If the instance is in the **Failed** state, delete it by referring to [Deleting a RabbitMQ Instance](#) and try purchasing another one. If the purchase fails a second time, contact customer service.

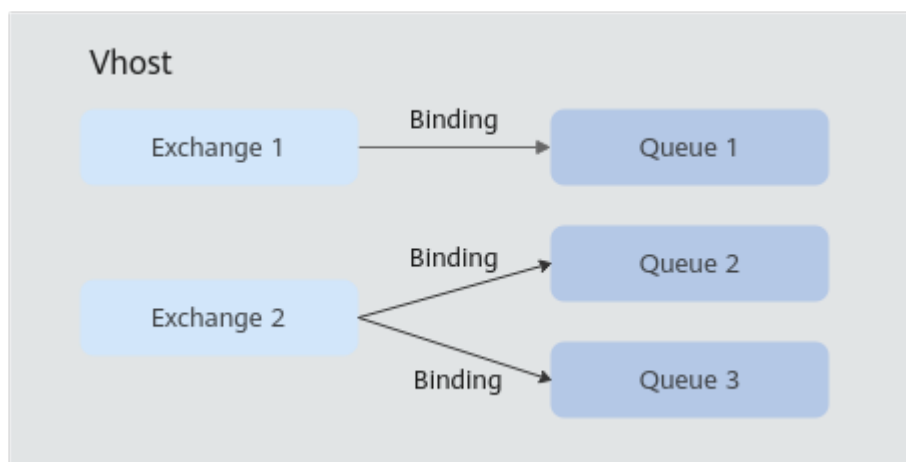
----End

6 Configuring Virtual Hosts

6.1 Creating a RabbitMQ Virtual Host

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

Figure 6-1 Virtual host architecture



Methods of creating a virtual host:

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


Notes and Constraints


- After an instance is created, a virtual host named `/` is automatically created.
- When a virtual host name of a RabbitMQ 3.x.x instance starts with a special character, such as a period (`.`), the monitoring data may not be displayed.

When a virtual host name contains special characters such as percent (%), vertical bar (|), or slash (/), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_) instead. For example, virtual host **Vhost.1%1|2_3/** is displayed as **Vhost.1_1_2_3_** in monitoring.

Creating a RabbitMQ Host (Console)

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click **Create Virtual Host**.

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

2 to 128 characters. Use only letters, digits, and special characters .%|_/_/

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

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

----End

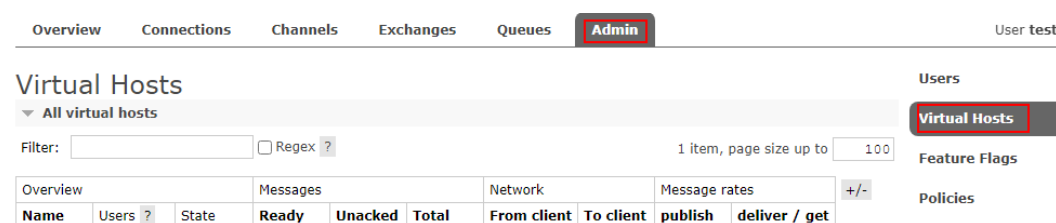
Creating a RabbitMQ Virtual Host (Management UI)

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

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

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

Figure 6-2 Virtual hosts



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

Figure 6-3 Creating a virtual host (management UI)

Virtual Hosts

▼ All virtual hosts

Filter: ☐ Regex ?

Overview			Messages		
Name	Users ?	State	Ready	Unacked	Total
/	guest, test	■ running	NaN	NaN	NaN

▼ Add a new virtual host

Name: *

Description:

Tags:

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

Figure 6-4 Virtual host list (management UI)

Virtual Hosts

▼ All virtual hosts

Filter: ☐ Regex ?

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

----End

6.2 Creating a RabbitMQ Exchange

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

This section describes how to create an exchange on the console.

Notes and Constraints


- For RabbitMQ 3.x.x instances, seven exchanges are created by default after virtual host creation. These exchanges include (AMQP default), amq.direct, amq.fanout, amq.headers, amq.match, amq.rabbitmq.trace, and amq.topic.


Prerequisite

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

Creating a RabbitMQ Exchange

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

Step 8 Configure the exchange name and other parameters by referring to [Table 6-1](#).

Table 6-1 Exchange parameters

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

Parameter	Description
Persistence	Indicates whether to enable exchange persistence. <ul style="list-style-type: none">• Enabled: The exchange survives server restart.• Disabled: The exchange will be deleted after server restarts and needs to be recreated.
Internal	Indicates whether exchanges are for internal use. <ul style="list-style-type: none">• Yes: An exchange can only bind another exchange instead of a queue.• No: Exchanges can bind exchanges and queues.

Step 9 Click **OK**.

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

----End

6.3 Binding a RabbitMQ Exchange

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

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

Notes and Constraints


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


Prerequisites

[An exchange](#) has been created.

Binding an Exchange to a Target Exchange

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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



- Step 5** In the navigation pane, choose **Instance > Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange. The **Bind Exchange** page is displayed.
- Step 8** Click **Add Binding**. The **Add Binding** dialog box is displayed.
- Step 9** Set the parameters by referring to [Table 6-2](#).

Table 6-2 Binding parameters

Parameter	Description
Type	Select the binding type: Select Exchange .
Target	Select a target exchange to be bound.
Routing Key	<p>Enter a key string to inform the exchange of which target exchanges to deliver messages to.</p> <ul style="list-style-type: none">This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to target exchanges with the routing keys matched. Messages cannot be routed to target exchanges without a routing key.For fanout exchanges and header exchanges, skip this parameter.

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

Binding Source Exchanges to a Queue

- Step 1** Log in to the console.
- Step 2** Click  in the upper left corner to select the region where your instance is located.
- Step 3** Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.
- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Instance > Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** On the **Queue** tab page, click **View Detail** in the row containing the desired queue.
- Step 8** On the **Bindings** tab, click **Add Binding**. The **Add Binding** dialog box is displayed.

Step 9 Set the parameters by referring to [Table 6-3](#).

Table 6-3 Binding parameters

Parameter	Description
Source	Select an exchange to be bound.
Routing Key	<p>Enter a key string to inform the exchange of which queues to deliver messages to.</p> <ul style="list-style-type: none">This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to queues with the routing keys matched. Messages cannot be routed to target queues without a routing key.For fanout exchanges and header exchanges, skip this parameter.

Step 10 Click **OK**.

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

----End

6.4 Creating a RabbitMQ Queue

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

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

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

Notes and Constraints


- When a queue name of a RabbitMQ 3.x.x instance starts with a special character, such as a period (.), the monitoring data may not be displayed. When a queue name contains special characters such as percent (%), vertical bar (|), and slash (/), this name is not consistently displayed on the monitoring page. The special characters are displayed as underscores (_) instead. For example, queue **Queue.1%1|2_3/** is displayed as **Queue.1_1_2_3_** in monitoring.
- On Cloud Eye, a special character (.) is counted as five characters. If the name of a queue of a RabbitMQ 3.x.x instance contains periods (.) and the counted total length exceeds 256 characters, monitoring data may not be displayed.


Prerequisite

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

Creating a RabbitMQ Queue

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

Step 8 Configure the queue name and other parameters by referring to [Table 6-4](#).

Table 6-4 Queue parameters

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

Parameter	Description
Lazy Queue	Enter lazy to make the queue lazy. Lazy queues store as many messages to the disk as possible. Messages are loaded to the memory only when they are being consumed. This reduces memory consumption.

Step 9 Click **OK**.

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

----End

6.5 Binding a RabbitMQ Queue

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

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

Notes and Constraints


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


Prerequisites

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

Binding a Queue to an Exchange

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

Step 8 Click **Add Binding**. The **Add Binding** dialog box is displayed.

Step 9 Set the parameters by referring to [Table 6-5](#).

Table 6-5 Binding parameters

Parameter	Description
Type	Select the binding type: To bind a queue, select Queue .
Target	Select a target queue to be bound.
Routing Key	Enter a key string to inform the exchange of which queues to deliver messages to. <ul style="list-style-type: none">• This parameter is required by direct exchanges and topic exchanges. Such exchanges route messages to queues with the routing keys matched. Messages cannot be routed to target queues without a routing key.• For fanout exchanges and header exchanges, skip this parameter.

Step 10 Click **OK**.

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

----End


6.6 Managing RabbitMQ Virtual Hosts


6.6.1 Viewing a RabbitMQ Virtual Host

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

Viewing a RabbitMQ Virtual Host

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

Table 6-6 Exchange details

Parameter	Description
Name	Name of an exchange.
Default	Whether this exchange is created by the system. <ul style="list-style-type: none">• Yes: This type of exchange cannot be deleted.• No: This type of exchange is manually created, and can be deleted.
Target Type	Type of an exchange. <ul style="list-style-type: none">• direct: Exchanges route messages to matching queues based on the routing keys.• fanout: Exchanges route messages to all bound queues.• topic: Exchanges route messages to queues based on routing key wildcard matching.• headers: Exchanges are related to the message headers. Routing keys are not used. Exchanges route messages based on matching between key-value pairs in the message headers and the binding (a key-value pair).
Persistence	Whether this exchange supports persistence. <ul style="list-style-type: none">• Yes: The exchange survives server restart.• No: The exchange will be deleted after server restarts and needs to be recreated.
Internal	Whether this exchange is for internal use. <ul style="list-style-type: none">• Yes: This exchange can only bind another exchange instead of a queue.• No: This exchange can bind exchanges and queues.
Automatically delete	Whether this exchange can be automatically deleted. <ul style="list-style-type: none">• Yes: This exchange will be automatically deleted when the last bound queue unbound from the exchange.• No: This exchange will not be deleted when the last bound queue unbound from the exchange.

Table 6-7 Queue details

Parameter	Description
Name	Name of a queue.
Accumulated Messages	Number of accumulated messages in this queue.

Parameter	Description
Persistence	Whether this queue supports persistence. <ul style="list-style-type: none">• Yes: This queue survives after server restart.• No: This queue will be deleted after server restart and needs to be recreated.
Auto-Delete	Whether this queue can be automatically deleted. <ul style="list-style-type: none">• Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue.• No: This queue will not be deleted when the last consumer unsubscribes from the queue.
Policy	Name of the policy set for this queue.

----End

6.6.2 Deleting RabbitMQ Virtual Hosts

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


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


Notes and Constraints

- The default virtual host created in instance creation cannot be deleted.
- Deleting a virtual host removes all its resources including exchanges and queues permanently.

Deleting Virtual Hosts (Console)

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Delete virtual hosts in any of the following ways:

- Select one or more virtual hosts and click **Delete Virtual Host** in the upper left corner.
- In the row containing the desired virtual host, click **Delete**.
- Click a virtual host name. The virtual host details page is displayed. Click **Delete** in the upper right corner.

WARNING

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

Step 7 In the displayed dialog box, click **OK**.

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

----End

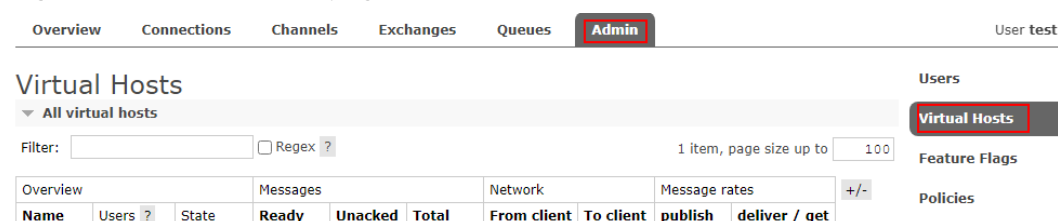
Deleting Virtual Hosts (RabbitMQ Management UI)

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

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

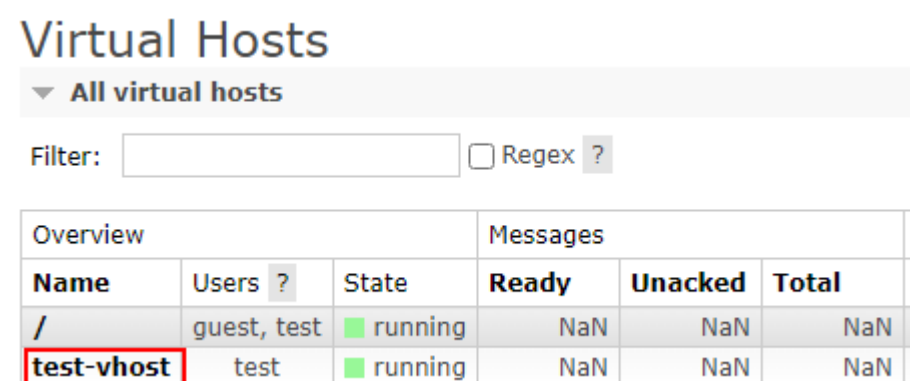
Step 3 In the navigation tree on the right, choose **Virtual Hosts**.

Figure 6-5 Virtual Hosts page



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

Figure 6-6 Virtual host to be deleted



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

Figure 6-7 Deleting a virtual host

Virtual Host: test-vhost

► Overview

► Permissions

► Topic permissions

▼ Delete this vhost

Delete this virtual host



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

Step 6 Click **OK**.

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

----End

6.7 Managing RabbitMQ Exchanges

6.7.1 Unbinding a RabbitMQ Exchange

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

Prerequisite


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


Notes and Constraints

Unbinding an exchange makes it unavailable permanently. Exercise caution.

Unbinding an Exchange from a Target Exchange

Step 1 Log in to the console.

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

- Step 3** Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.
- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Instance > Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** On the **Exchange** tab page, click **Bind Exchange** in the row containing the desired exchange.
- Step 8** In the row containing the desired exchange, click **Remove Binding**.





WARNING

Removing a binding makes it unavailable permanently. Exercise caution.

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

----End

Unbinding a Queue from a Source Exchange

- Step 1** Log in to the console.
- Step 2** Click  in the upper left corner to select the region where your instance is located.
- Step 3** Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.
- Step 4** Click an instance name to go to the instance details page.
- Step 5** In the navigation pane, choose **Instance > Virtual Hosts**.
- Step 6** Click a virtual host name.
- Step 7** On the **Queue** tab page, click **View Detail** in the row containing the desired queue.
- Step 8** In the row containing the desired exchange, click **Remove Binding**.



WARNING

Removing a binding makes it unavailable permanently. Exercise caution.

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

----End

6.7.2 Deleting RabbitMQ Exchanges

This section describes how to delete exchanges on the console.

Notes and Constraints


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


Prerequisite

An [exchange](#) has been created.

Deleting RabbitMQ Exchanges

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

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



WARNING

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

Step 8 Click **OK**.

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

----End

6.8 Managing RabbitMQ Queues

6.8.1 Viewing a RabbitMQ Queue


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


Prerequisite

A [queue](#) has been created.

Viewing a RabbitMQ Queue

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

Step 7 On the **Queue** tab page, click **View Detail** in the row containing the desired queue. The [basic information](#), [bindings](#), and [consumers](#) of the queue are displayed.

Table 6-8 Queue basic information parameters

Parameter	Description
Name	Name of a queue.
Vhost	Name of the virtual host to which the queue belongs.
Accumulated Messages	Number of accumulated messages in this queue.
Consumers	Number of consumers that consume messages in this queue.
Automatically delete	Whether this queue can be automatically deleted. <ul style="list-style-type: none">• Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue.• No: This queue will not be deleted when the last consumer unsubscribes from the queue.
Dead Letter Exchange	When this queue is bound to a dead letter exchange, the name of that exchange is displayed. Otherwise, -- is displayed.
Dead Letter Routing Key	When this queue is bound to a dead letter exchange and a dead letter routing key is set, the key is displayed. Otherwise, -- is displayed.

Parameter	Description
Time to Live(ms)	When this queue is configured with message retention, this time is displayed. Otherwise, -- is displayed. If the time to live passed and messages are still not consumed, they become dead letter messages and are sent to the dead letter exchange.
Lazy Queue	If this queue is a lazy queue, lazy is displayed. Otherwise, -- is displayed. For more information, see Configuring Lazy Queues .

Table 6-9 Bindings parameters

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

Table 6-10 Consumers parameters

Parameter	Description
Consumer Tag	Unique identifier of a consumer client.
Channel	Channel through which a consumer client connects to the RabbitMQ instance.
Ack Required	Whether messages are automatically acknowledged. <ul style="list-style-type: none">• true: Messages are marked as acknowledged and deleted from the queue immediately after being sent to a consumer.• false: Messages are marked as unacknowledged and retained in the queue until a consumer sends an acknowledgment (ack) to RabbitMQ. RabbitMQ then deletes the messages.
Prefetch Count	Prefetch value of messages. For more information, see Configuring RabbitMQ Message Prefetch .
User	Username used by a consumer client to connect to the RabbitMQ instance.

----End

6.8.2 Clearing Messages in a RabbitMQ Queue

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

Methods of deleting messages:

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

Notes and Constraints


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


Prerequisite

[A queue](#) has been created.

Clearing Messages in a Queue (Console)

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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



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

Step 8 Click **OK**.

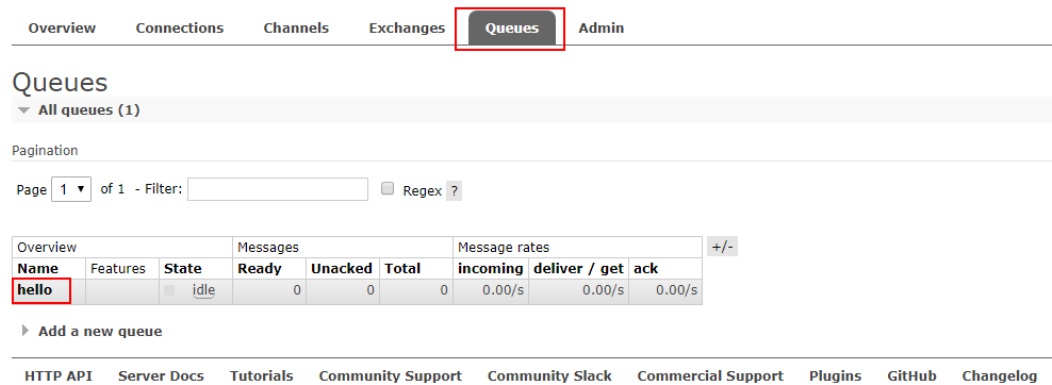
----End

Clearing Messages in a Queue (RabbitMQ Management UI)

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

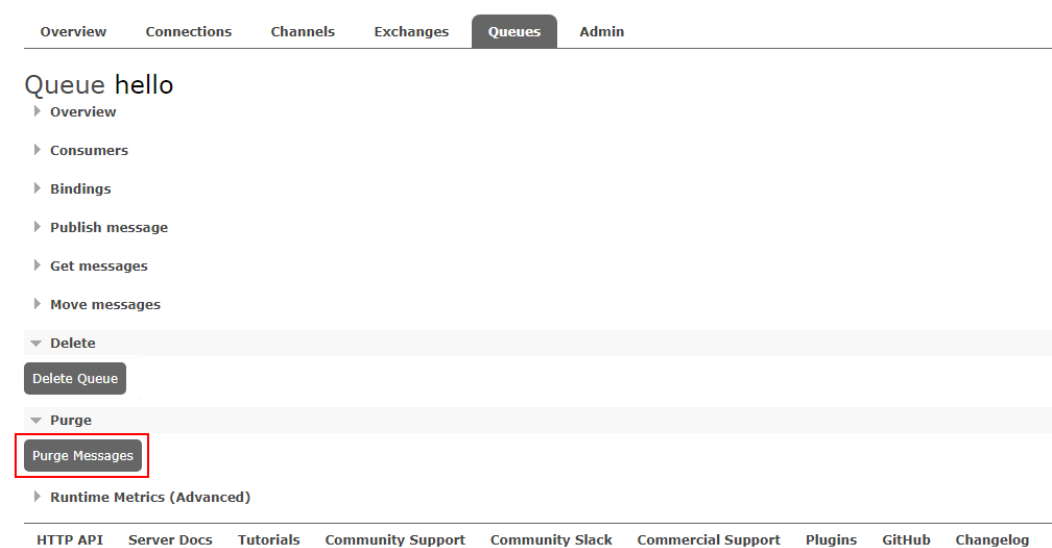
Step 2 On the **Queues** tab page, click the name of a queue.

Figure 6-8 Queues



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

Figure 6-9 Clearing messages in a queue



CAUTION

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

----End

6.8.3 Unbinding a RabbitMQ Queue

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

Notes and Constraints


Unbinding a queue makes it unavailable permanently. Exercise caution.


Prerequisites

- [An exchange](#) has been created.
- [A queue](#) has been created.
- The exchange has been [bound with the queue](#).

Unbinding an Exchange from a Queue

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

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



Removing a binding makes it unavailable permanently. Exercise caution.

Step 9 Click **Yes**.

----End

6.8.4 Configuring Queue Mirroring

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

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

Prerequisite

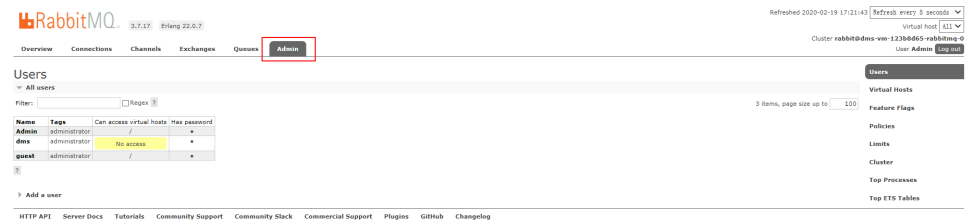
A cluster RabbitMQ instance has been created.

Configuring RabbitMQ Queue Mirroring

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

Step 2 Click the **Admin** tab.

Figure 6-10 Admin tab page



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

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

Figure 6-11 Creating a virtual host



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

Figure 6-12 Setting virtual host policies

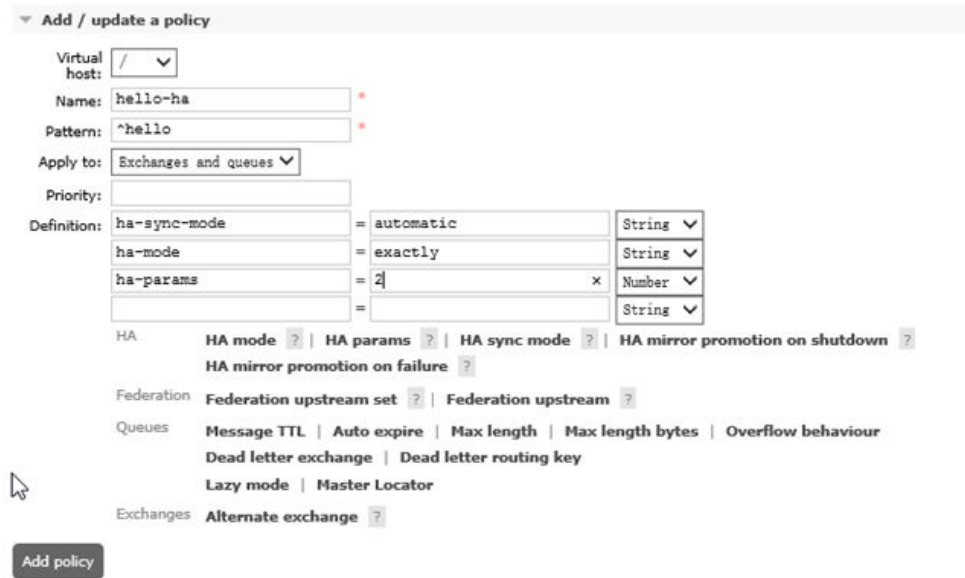


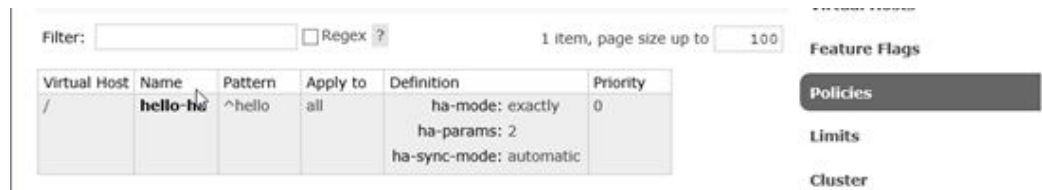
Table 6-11 Policy elements

Parameter	Description
Virtual Host	Select the virtual host to which the policy applies. The slash (/) indicates the default virtual host.

Parameter	Description
Name	The policy name, which can be customized.
Pattern	Regular expression that defines the pattern for matching queues.
Apply to	Object to which the policy applies.
Priority	A larger value indicates a higher priority.
Definition	<p>Definition of the mirror, which consists of ha-sync-mode, ha-mode, and ha-params.</p> <ul style="list-style-type: none">• ha-sync-mode: queue synchronization mode. Options: automatic and manual.<ul style="list-style-type: none">– automatic: Data is automatically synchronized from the master.– manual: Data is manually synchronized from the master.• ha-mode: queue mirroring mode. Options: all, exactly, and nodes.<ul style="list-style-type: none">– all: Mirror the queue across all nodes in the cluster.– exactly: Mirror the queue to a specific number (determined through ha-params) of nodes in the cluster.– nodes: Mirror the queue to specific nodes (determined through ha-params).• ha-params: This parameter is used in the ha-mode mode. <p>NOTE Mirroring queues to all nodes in a cluster may waste network and disk I/O resources. You are advised to set parameters as follows:</p> <ul style="list-style-type: none">• ha-sync-mode: automatic• ha-mode: exactly• ha-params: $n/2 + 1$. n is the total number of nodes in a cluster. The value of $n/2$ is rounded down. For example, if the total number of nodes in a cluster is 3, set ha-params to 2 ($3/2 = 1.5$, 1.5 is rounded down to 1, $1 + 1 = 2$). Queues will be mirrored to a master and a standby node. This configuration ensures high data availability, and avoids unnecessary resource overhead.

Step 5 Click **Add policy**.

The following figure shows a successfully added policy.

Figure 6-13 Virtual host policy

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

----End

6.8.5 Configuring Lazy Queues

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

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

Lazy queues store as many messages to the disk as possible. Messages are loaded to the memory only when they are being consumed. This reduces memory consumption, but increases I/O and affects single-queue throughput. An important goal of lazy queues is to support long queues, that is, queues with many messages stored or stacked.

Lazy queues are recommended in the following scenarios:

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

Lazy queues are not suitable in the following scenarios:

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

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

Notes and Constraints

Available for RabbitMQ 3.8.35 and later.

Configuring a Lazy Queue

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

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

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

Configuring a Lazy Queue on a Java Client

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

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

Configuring a Lazy Queue on the RabbitMQ Management UI

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

Step 2 Click the **Admin** tab.

Figure 6-14 Admin tab page



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

Step 4 Add a policy.

Figure 6-15 Adding a policy

▼ Add / update a policy

Virtual host:

Name:

Pattern:

Apply to:

Priority:

Definition: =

Queues [All types] | Max length | Max length bytes | Overflow behaviour ? | Auto expire

Dead letter exchange | Dead letter routing key

Message TTL ?

Table 6-12 Policy elements

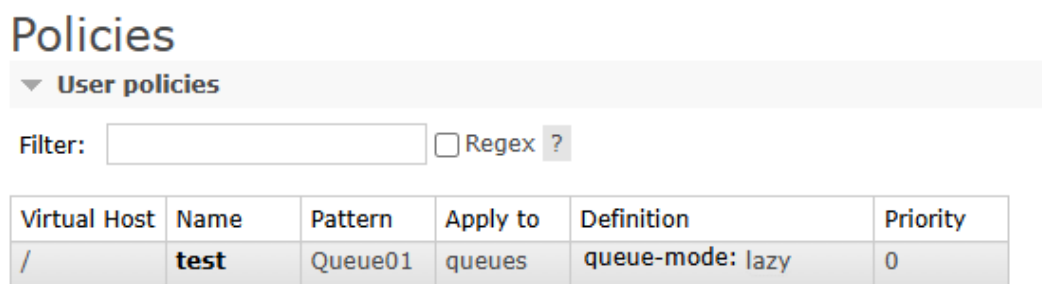
Parameter	Description
Virtual Host	Specify the virtual host. The slash (/) indicates the default virtual host.

Parameter	Description
Name	The policy name, which can be customized.
Pattern	Regular expression that defines the pattern for matching queues.
Apply to	Object to which the policy applies. Select Queues .
Priority	A larger value indicates a higher priority.
Definition	Configure lazy queues by setting queue-mode=lazy .

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

The following figure shows a successfully added policy.

Figure 6-16 Viewing a lazy queue policy



-----End

Configuring a Lazy Queue on the RabbitMQ Console

Step 1 Create a queue by referring to [Creating a RabbitMQ Queue](#) and set **Lazy Queue** to **lazy**.

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

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

Figure 6-17 Viewing queue details**View Detail**

Name	Queue03	Vhost	/
Accumulated Messages	0	Consumers	0
Automatically delete	No	Dead Letter Exchange	--
Dead Letter Routing Key	--	Time to Live(ms)	--
Lazy Queue	lazy		

----End

6.8.6 Configuring RabbitMQ Quorum Queues

Quorum queues provide the queue replication capability to ensure high data availability and security. Quorum queues can be used to replicate queue data between RabbitMQ nodes, ensuring that queues can still run when a node breaks down.

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

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

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

Notes and Constraints

Available for RabbitMQ 3.8.35 and later.

Comparing Quorum Queues and Mirrored Queues

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

Mirrored queues are slow at replicating messages.

- A mirrored queue has a queue leader and many mirrors. When a producer sends a message to the queue leader, the leader replicates the message to the mirrors, and sends back confirmation to the producer only after all mirrors have saved the message.

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

Quorum queues can solve these problems.

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

Compared with mirrored queues, quorum queues have fewer features, as shown in [Table 6-13](#). Quorum queues consume more memory and disk space.

Table 6-13 Comparing quorum queues and mirrored queues

Feature	Mirrored Queues	Quorum Queues
Non-durable queues	Supported	Not supported
Exclusive queues	Supported	Not supported
Message persistence	Per message	Always
Queue rebalancing	Automatic	Manual
Message TTL	Supported	Not supported
Queue TTL	Supported	Supported
Queue length limit	Supported	Supported (except x-overflow: reject-publish-dlx)
Lazy queues	Supported	Supported after the queue length is limited
Message priority	Supported	Not supported
Consumption priority	Supported	Supported

Feature	Mirrored Queues	Quorum Queues
Dead letter exchanges	Supported	Supported
Dynamic policy	Supported	Supported
Poison message (let consumers consume infinitely) handling	Not supported	Supported
Global QoS prefetch	Supported	Not supported

Configuring a Quorum Queue on a Java Client

When declaring a queue, set the **x-queue-type** queue argument to **quorum**. The default replication factor of a quorum queue is five.

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

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

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

Configuring a Quorum Queue on the RabbitMQ Management UI

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

- Step 1** Log in to the [RabbitMQ management UI](#).
- Step 2** Click the **Queues** tab.
- Step 3** Create a quorum queue.

Figure 6-18 Creating a quorum queue

▼ Add a new queue

Virtual host:

Type:

Name:

Node:

Arguments: =

Add [Auto expire](#) | [Message TTL](#) | [Overflow behaviour](#) | [Single active consumer](#) | [Dead letter exchange](#) | [Dead letter routing key](#) | [Max length](#) | [Max length bytes](#) | [Delivery limit](#) | [Initial cluster size](#) | [Dead letter strategy](#) | [Leader locator](#)

Table 6-14 Queue parameters

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

Step 4 Click **Add queue**.

Step 5 On the **Queues** page, check that **Type** is **quorum**, as shown in [Figure 6-19](#). The quorum queue is created.

Figure 6-19 Checking the queue type

Queues

▼ All queues (4)

Pagination

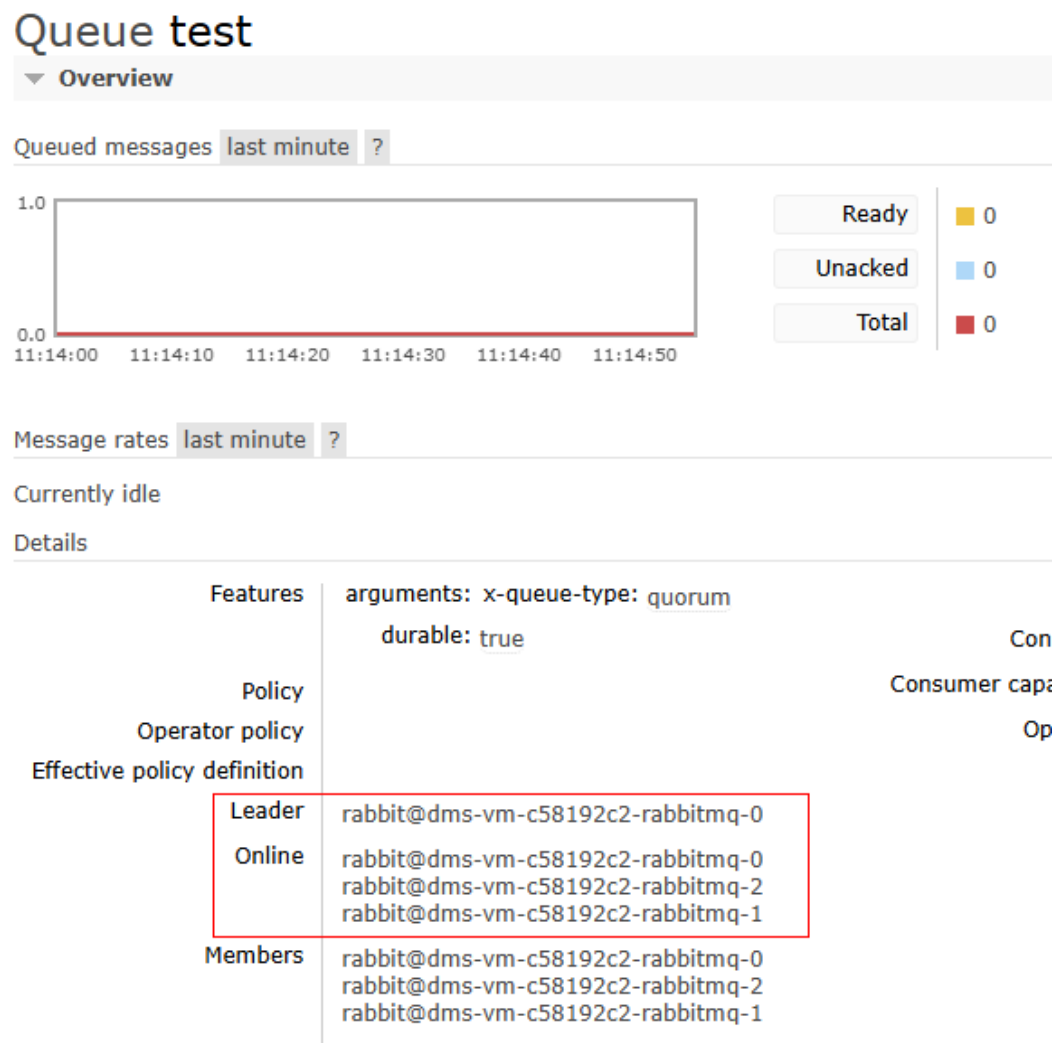
Page 1 of 1 - Filter: ☐ Regex ?

Overview						Messages		
Virtual host	Name	Node	Type	Features	State	Ready	Unacked	To
/	Queue01	rabbit@dms-vm-c58192c2-rabbitmq-0	classic	D test	idle	20,806	0	
/	Queue03	rabbit@dms-vm-c58192c2-rabbitmq-2	classic	D Args	idle	0	0	
/	sac-queue	rabbit@dms-vm-c58192c2-rabbitmq-1	classic	D SAC	idle	0	0	
/	test	rabbit@dms-vm-c58192c2-rabbitmq-0 +2	quorum	D Args	running	0	0	

In the **Node** column, **+2** indicates that the queue has two replicas. Blue indicates that the two replicas have been synchronized. Red indicates that some messages have not been replicated.

Step 6 (Optional) Click the name of the desired queue. Check the node where the leader of this quorum queue resides and the node where active followers reside.

Figure 6-20 Queue details page



----End

Configuring the Quorum Queue Length

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

Table 6-15 Configuring the Quorum Queue Length

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

Parameter	Description
x-max-in-memory-length	Maximum number of messages of the quorum queue that can be stored in memory.
x-max-in-memory-bytes	Maximum size (in bytes) of messages of the quorum queue that can be stored in memory.

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

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

Figure 6-21 Using a policy to set x-max-in-memory-bytes

Policies

▼ User policies

Filter: ☐ Regex ?

... no policies ...

▼ Add / update a policy

Name: *

Pattern: *

Apply to:

Priority:

Definition: = *

=

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

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

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

Exchanges [Alternate exchange](#) ?

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

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

Figure 6-22 Setting x-max-in-memory-length in the queue argument

▼ Add a new queue

Type:

Name:

Node:

Arguments: =

Add | | | | | | | |

Max in memory bytes

Add queue

6.8.7 Configuring a RabbitMQ Exclusive Queue

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

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

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

Notes and Constraints

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

Example Configuration

On a **Java client**:

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

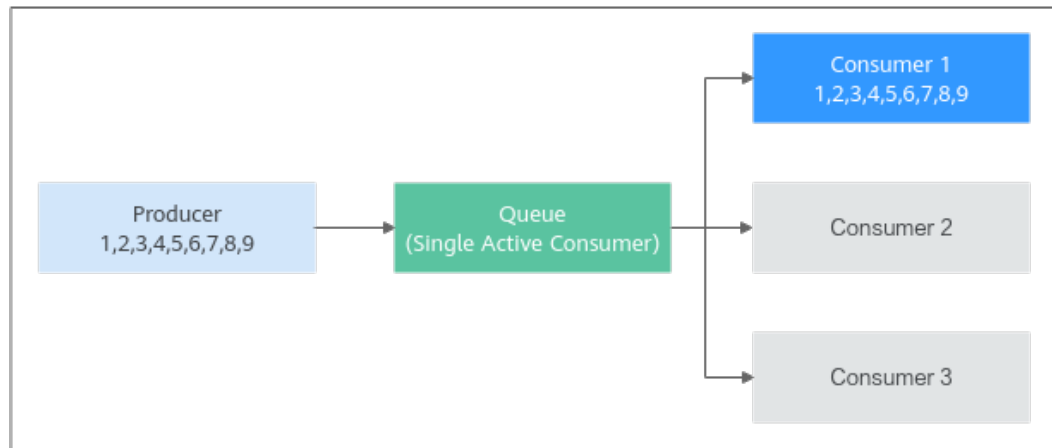
6.8.8 Configuring a Single Active Consumer

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

In [Figure 6-23](#), Producer produces nine messages. Due to the setting of single active consumer, only Consumer 1 can consume messages.

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

Figure 6-23 Single active consumer



Notes and Constraints

Available for RabbitMQ 3.8.35 and later.

Configuring a Single Active Consumer on a Java Client

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

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

```
Channel ch = ...;
Map<String, Object> arguments = newHashMap<String, Object>();
arguments.put("x-single-active-consumer", true);
ch.queueDeclare("my-queue", false, false, false, arguments);
```

Configuring a Single Active Consumer on the RabbitMQ Management UI

- Step 1** Log in to the [RabbitMQ management UI](#).
- Step 2** Click the **Queues** tab.
- Step 3** Create a single active consumer queue.

Figure 6-24 Creating a single active consumer queue

Add a new queue

Virtual host:

Type:

Name:

Durability:

Node:

Auto delete:

Arguments: =

=

Add | |

| |

|

| |

Table 6-16 Queue parameters

Parameter	Description
Virtual Host	Virtual host to which a single active consumer queue belongs. The slash (/) indicates the default virtual host.
Type	Queue type.
Name	Name of the single active consumer queue, which is user-defined.
Durability	Whether to enable persistence. <ul style="list-style-type: none">• Durable: This queue survives after server restart.• Transient: This queue will be deleted after server restart and needs to be recreated.
Node	Node where a single active consumer queue is deployed.
Auto delete	Whether to enable automatic deletion. <ul style="list-style-type: none">• Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue.• No: This queue will not be deleted when the last consumer unsubscribes from the queue.
Arguments	Set the single active consumer attribute through parameter x-single-active-consumer=true .

Step 4 Click **Add queue**.

Step 5 Check whether the queue features contain single active consumer on the **Queues** page. As shown in [Figure 6-25](#), **SAC** indicates that a single active consumer has been set in the queue.

Figure 6-25 Viewing queue features

Queues

▼ All queues (1)

Pagination

Page 1 of 1 - Filter: ☐ Regex ?

Overview				Messages			Message rates			+/-
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
sac-queue	classic	D SAC Args	idle	0	0	0				

▼ Add a new queue x-single-active-consumer: true

----End

6.8.9 Deleting RabbitMQ Queues

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

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

Notes and Constraints


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


Prerequisite

[A queue](#) has been created.

Deleting a Queue (Console)

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 6 Click a virtual host name.

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

- Select one or more queues and click **Delete Queue** in the upper left corner.
- In the row containing the desired queue, click **Delete**.

 **WARNING**

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

Step 8 Click **OK**.

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

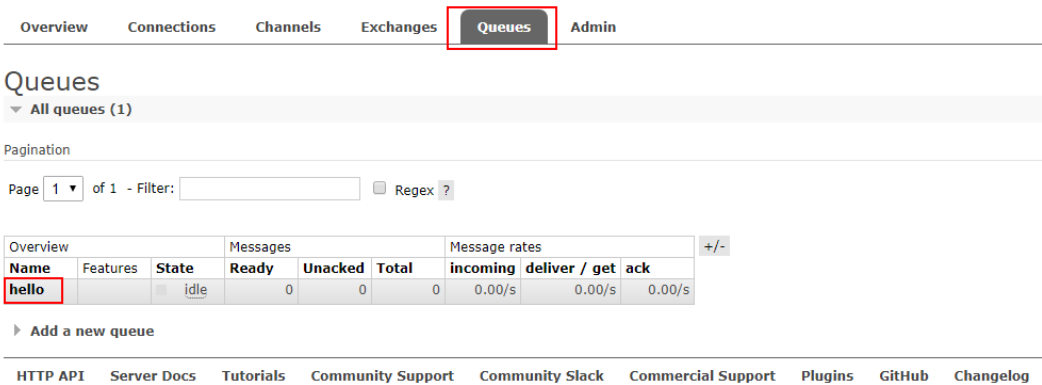
-----End

Deleting a Queue (RabbitMQ Management UI)

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

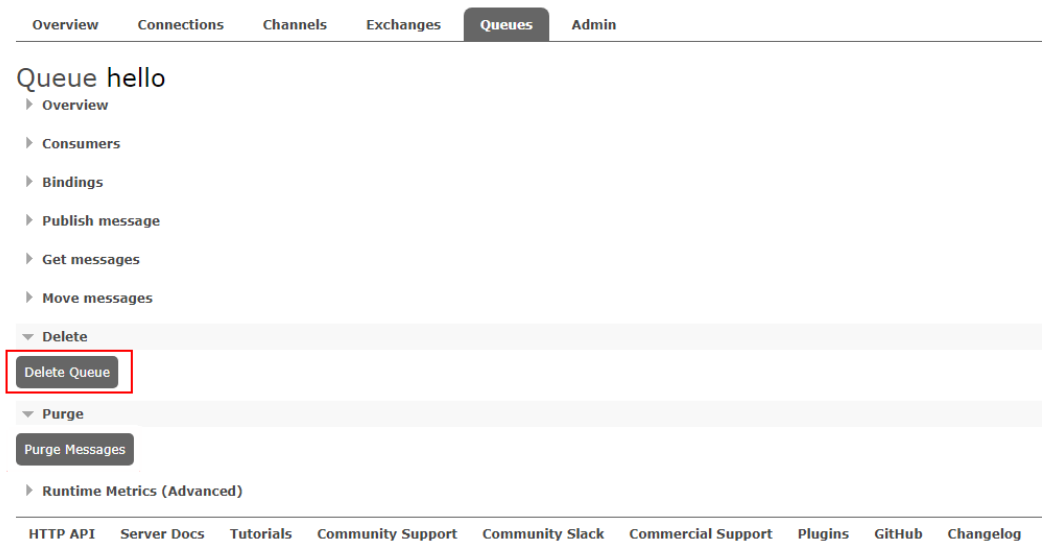
Step 2 On the **Queues** tab page, click the name of the desired queue.

Figure 6-26 Queues



Step 3 Click **Delete Queue**.

Figure 6-27 Deleting a queue



WARNING

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

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

----End

Deleting Queues in Batches (RabbitMQ Management UI)

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

Step 1 Log in to the RabbitMQ management UI.

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

Figure 6-28 Adding a policy to delete queues in batches

The screenshot shows the RabbitMQ Management UI with the 'Admin' tab selected. The 'Policies' section is active, and the 'Add / update a policy' form is displayed. The form includes the following fields and options:

- Name:** Delete queues
- Pattern:** .*
- Apply to:** Queues
- Priority:** (empty)
- Definition:** expires = 1 (Number)
- HA:** HA mode, HA params, HA sync mode, HA mirror promotion on shutdown
- Federation:** Federation upstream set, Federation upstream
- Queues:** Message TTL, Auto expire, Max length, Max length bytes, Overflow behaviour, Dead letter exchange, Dead letter routing key, Lazy mode, Master Locator
- Exchanges:** Alternate exchange

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

Table 6-17 Policy elements

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

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

**WARNING**

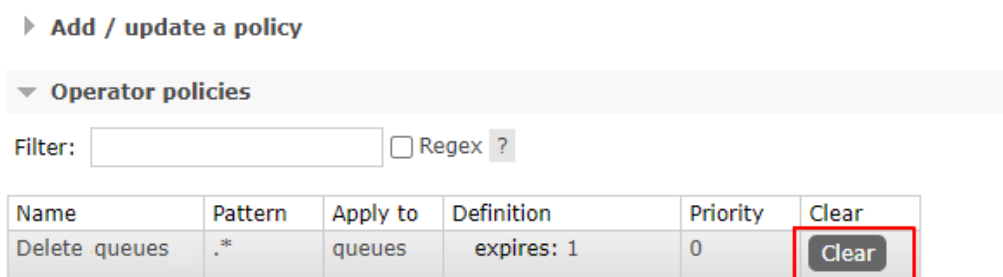
Deleting a queue removes all its configurations including exchange-queue bindings permanently.

Step 3 Click **Add policy**.

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

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

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

Figure 6-29 Deleting the policy

----End

7

Accessing a RabbitMQ Instance

7.1 Configuring RabbitMQ Network Connections

7.1.1 RabbitMQ Network Connection Requirements

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

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

Table 7-1 Connection modes

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

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

 **NOTE**

After a security group is created, its default inbound rule allows communication among ECSs within the security group and its default outbound rule allows all outbound traffic. In this case, you can access a RabbitMQ instance within a VPC, and do not need to add rules according to [Table 7-2](#).

Table 7-2 Security group rules

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

7.1.2 Configuring RabbitMQ Public Access

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

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

Notes and Constraints


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


Prerequisite

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

Enabling Public IPv4 Access

Step 1 Log in to the console.


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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Click the desired instance to view its details.

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

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

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

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

Step 7 Set the security group rules listed in [Table 7-3](#) for the RabbitMQ instance so that you can access RabbitMQ using the IPv4 EIP.

Table 7-3 Security group rules


Direction	Type	Protocol	Port	Source	Description
Inbound	IPv4	TCP	5672	IP address or IP address group of the RabbitMQ client	Accessing a RabbitMQ instance at an IPv4 address on a client (without SSL)


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

----End

Disabling Public IPv4 Access

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Click the desired instance to go to the instance details page.

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

Step 6 Click .

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

----End

7.2 Configuring Heartbeats on the RabbitMQ Client

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

What Is a Heartbeat?

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

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

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

After the heartbeat timeout is configured, the RabbitMQ server and client send AMQP heartbeat frames to each other at an interval of half the heartbeat timeout. After a client misses two heartbeats, it is considered unreachable and the TCP connection is closed. If the client detects that the server cannot be accessed due to heartbeats, the client needs to reconnect to the server. For more information about heartbeats, see [Detecting Dead TCP Connections with Heartbeats and TCP Keepalives](#).

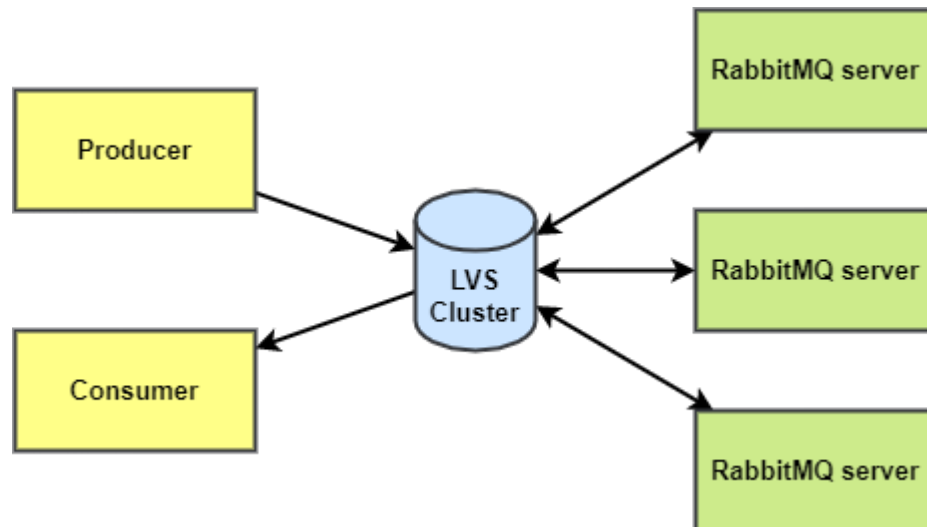


CAUTION

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

LVS Heartbeat Timeout

Cluster RabbitMQ instances use Linux Virtual Servers (LVSs) for load balancing, as shown in [Figure 7-1](#). Single-node instances do not have LVSs.

Figure 7-1 Load balancing of a cluster instance

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

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

Configuring Heartbeats on a Client

- **Java client**

Before creating a connection, configure the

ConnectionFactory#setRequestedHeartbeat parameter. Example:

```
ConnectionFactory cf = new ConnectionFactory();  
// The heartbeat timeout is 10s.  
cf.setRequestedHeartbeat(10);
```

- **.NET client**

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

- **Python Pika**

```
// The heartbeat is 10s.  
params = pika.ConnectionParameters(host='host', heartbeat=10,  
credentials=pika.PlainCredentials('username', 'passwd'))  
connection = pika.BlockingConnection(params)  
  
while True:  
    channel.basic_publish(exchange="", routing_key='hello', body='Hello World!')  
    print(" [x] Sent 'Hello World!'")  
    # The producer needs to use connection.sleep() to trigger a heartbeat. time.sleep() cannot trigger  
    heartbeats.  
    connection.sleep(200)
```

- **PHP client**

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

- **Spring-amqp**

```
// The heartbeat is 10s.  
@Bean
```

```
CachingConnectionFactory connectionFactory(ConnectionFactory rabbitConnectionFactory) {  
    CachingConnectionFactory ccf = new CachingConnectionFactory(rabbitConnectionFactory);  
    ccf.setHost("");  
    ccf.setRequestedHeartBeat(10);  
    // ...  
    return ccf;  
}
```

7.3 Accessing RabbitMQ on a Client (SSL Disabled)

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

Prerequisites

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

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

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

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

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

Accessing the Instance in CLI Mode

The following uses Linux as an example.

Step 1 Log in to the client server.

Step 2 Download [RabbitMQ-Tutorial.zip](#) sample project code.

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

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

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

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

```
cd RabbitMQ-Tutorial
```

Step 5 Create messages using the sample project.

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

Table 7-4 Message production parameters

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

Sample message production:

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

Step 6 Retrieve messages using the sample project.

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

Table 7-5 Message consumption parameters

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

Sample message consumption:

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

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

----End

Java Sample Code

- Accessing an instance and producing messages are as follows. [Table 7-6](#) describes the parameters to be modified.

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

```
factory.setPort(port);
factory.setVirtualHost(" VHOST_NAME");

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

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

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

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

Table 7-6 Parameters

Parameter	Description
VHOST_NAME	Name of the virtual host that contains the queue for messages to be sent to
QUEUE_NAME	Name of the queue for messages to be sent to
Hello World!	The message to be sent in this sample

- Accessing an instance and consuming messages are as follows. [Table 7-7](#) describes the parameters to be modified.

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

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

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

Table 7-7 Parameters

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

Parameter	Description
QUEUE_NAME	Name of the queue to consume messages

7.4 Accessing RabbitMQ on a Client (SSL Enabled)

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

Prerequisites

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

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

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

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

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

Accessing the Instance in CLI Mode

The following uses Linux as an example.

Step 1 Log in to the client server.

Step 2 Download [RabbitMQ-Tutorial.zip](#) sample project code.

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

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

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

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

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

Step 5 Produce messages using the sample project.

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

Table 7-8 Message production parameters

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

Figure 7-2 Sample project for message creation

```
root@ecs-3b6f RabbitMQ-Tutorial-SSL# java -cp ./rabbitmq-tutorial-sll.jar Send 192.168.1.35 5671 root admin13
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 admin13
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[x] Sent 'Hello World!'
```

Step 6 Consume messages using the sample project.

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

Table 7-9 Message consumption parameters

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

Figure 7-3 Sample project for message retrieval

```
root@ecs-3b6f RabbitMQ-Tutorial-SSL# java -cp ./rabbitmq-tutorial-sll.jar Recv 192.168.1.35 5671 root admin13
LF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
LF4J: Defaulting to no-operation (NOP) logger implementation
LF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
[*] Waiting for messages. To exit press CTRL+C
[x] Received 'Hello World!'
[x] Received 'Hello World!'
c[root@ecs-3b6f RabbitMQ-Tutorial-SSL#
```

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

----End

Java Sample Code

- Accessing an instance and producing messages are as follows. [Table 7-10](#) describes the parameters to be modified.

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

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

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

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

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

Table 7-10 Parameters

Parameter	Description
VHOST_NAME	Name of the virtual host that contains the queue for messages to be sent to
QUEUE_NAME	Name of the queue for messages to be sent to
Hello World!	The message to be sent in this sample

- Accessing an instance and consuming messages are as follows. [Table 7-11](#) describes the parameters to be modified.

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

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

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

Table 7-11 Parameters

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

8 Managing Messages

8.1 Configuring RabbitMQ Dead Letter Messages

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

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

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

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

RabbitMQ dead letter messages may compromise performance. Exercise caution.

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

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

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

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

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

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

Figure 8-1 Configuring a dead letter exchange and routing key

Create Queue

Basic Settings

Name
Fixed once created.

Persistence ☒

Auto-Delete ☐

More Settings ^

Dead Letter Exchange
Rejected and expired messages are resent to this exchange.

Dead Letter Routing Key
The dead letter exchange sends dead letter messages to the queue with a binding key that corresponds to this routing key.

Time to Live ms
How long messages published to queues can remain before being discarded.

Lazy Queue
Make the queue lazy to store more messages on disk and save memory. Otherwise, messages are stored in memory and delivered quickly.

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

If the dead letter exchange and routing key are the same as those set in [Step 1](#), the dead letter exchange and routing key are configured successfully.

Figure 8-2 Viewing the dead letter exchange and routing key**View Detail**

Name	Queue05	Vhost	/
Accumulated Messages	0	Consumers	0
Automatically delete	No	Dead Letter Exchange	DLQ01
Dead Letter Routing Key	test	Time to Live(ms)	--
Lazy Queue	--		

----End

8.2 Configuring RabbitMQ Message Acknowledgment

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

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

Notes and Constraints

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

Producer Confirms

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

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

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

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

Consumer Acknowledgment

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

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

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

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

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

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

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

8.3 Configuring RabbitMQ Message Prefetch

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

Consider the following factors when setting prefetch:

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

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

Prefetch Suggestions

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

Notes and Constraints

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

Setting the Prefetch Value

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

```
ConnectionFactory factory = new ConnectionFactory();

Connection connection = factory.newConnection();
Channel channel = connection.createChannel();

// Set the prefetch to 10.
channel.basicQos(10, false);

QueueingConsumer consumer = new QueueingConsumer(channel);
channel.basicConsume("my_queue", false, consumer);
```

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

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

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

9 Advanced Features

9.1 Configuring RabbitMQ Persistence

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

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

Notes and Constraints

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

Setting Exchange Persistence on the RabbitMQ Management UI

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

Step 2 Click the **Exchanges** tab.

Step 3 Create an exchange and set **durable** to **true**, as shown in [Figure 9-1](#).

Figure 9-1 Configuring exchange persistence (management UI)

▼ Add a new exchange

Name: *

Type: ▼

Durability: ▼

Auto delete: ? ▼

Internal: ? ▼

Arguments: = ▼

= ▼

Add **Alternate exchange** ?

Add exchange

Table 9-1 Exchange creation parameters

Parameter	Description
Name	Exchange name, which is user-defined.
Type	<p>Exchange type.</p> <p>Select a value from the drop-down list.</p> <ul style="list-style-type: none">• direct: Exchanges route messages to matching queues based on the routing keys.• fanout: Exchanges route messages to all bound queues.• topic: Exchanges route messages to queues based on routing key wildcard matching.• headers: Exchanges are related to the message headers. Routing keys are not used. Exchanges route messages based on matching between key-value pairs in the message headers and the binding (a key-value pair).
Durability	<p>Whether to enable persistence.</p> <ul style="list-style-type: none">• Durable: The exchange survives server restart.• Transient: The exchange will be deleted after server restarts and needs to be recreated.
Auto delete	<p>Whether to enable automatic deletion.</p> <ul style="list-style-type: none">• Yes: This exchange will be automatically deleted when the last bound queue unbound from the exchange.• No: This exchange will not be deleted when the last bound queue unbound from the exchange.

Parameter	Description
Internal	Indicates whether exchanges are for internal use. <ul style="list-style-type: none">• Yes: This exchange can only bind another exchange instead of a queue.• No: This exchange can bind other exchanges and queues.
Arguments	Set durable=true to configure exchange persistence.

Step 4 In the exchange list, move the cursor to **Args** of the new exchange, as shown in [Figure 9-2](#). If **durable: true** is displayed in the floating window, the exchange persistence is configured successfully.

Figure 9-2 Exchange persistence configured (management UI)

Exchanges

▼ All exchanges (9)

Pagination

Page 1 ▼ of 1 - Filter: ☐ Regex ?

Name	Type	Features	Message rate in	Message rate out	+/-
(AMQP default)	direct	D			
Exchange01	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			
durable-exchange	direct	D Args			

----End

Setting Exchange Persistence on the RabbitMQ Console

Step 1 Create an exchange by referring to [Creating a RabbitMQ Exchange](#) and configure exchange persistence, as shown in [Figure 9-3](#).

Figure 9-3 Configuring exchange persistence (console)

Create Exchange

1.Exchanges route RabbitMQ messages.

2.Exchanges route messages based on the binding keys, routing keys, and headers.

3.Producers send messages to exchanges first, rather than directly to queues. Exchange route messages to one or more queues.

Name

Exchange-36202605

Fixed once created.

Target Type

direct

fanout

topic

headers

Automatically delete

Persistence

Internal

Step 2 In the exchange list, check whether persistence is enabled for the new exchange. If **Yes** is displayed in the **Persistence** column, persistence is enabled, as shown in [Figure 9-4](#).

Figure 9-4 Exchange persistence configured (console)

<input type="checkbox"/> Name	Default	Target Type	Persistence	Internal	Automatica...	Operation
<input type="checkbox"/> (AMQP default)	Yes	direct	Yes	No	No	Bind Exchange Delete
<input type="checkbox"/> Exchange-36202605	No	direct	Yes	No	No	Bind Exchange Delete

----End

Setting Queue Persistence on the RabbitMQ Management UI

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

Step 2 Click the **Queues** tab.

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

Issue 05 (2025-09-03)

Copyright © Huawei Cloud Computing Technologies Co., Ltd.

104

Figure 9-5 Creating queues

▼ Add a new queue

Type:

Name:

Durability:

Node:

Auto delete:

Arguments: =

Add | | | |
 | | |
 |

Add queue

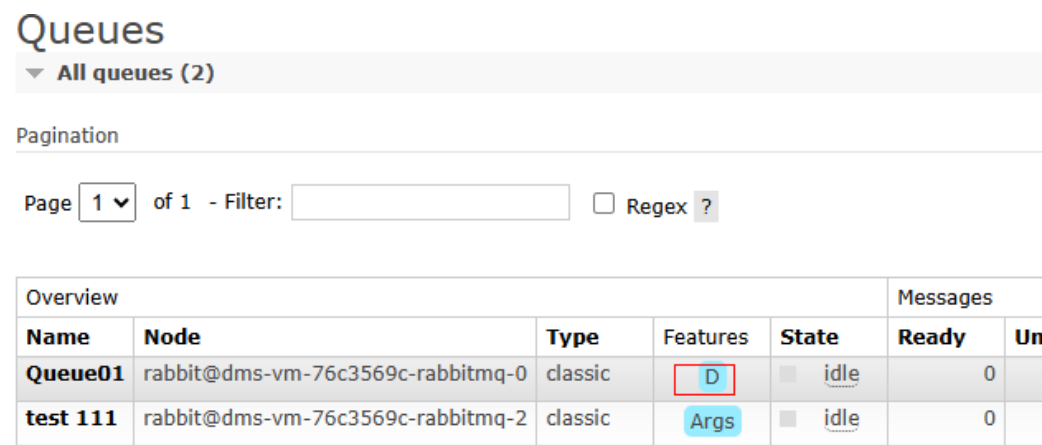
Table 9-2 Queue parameters

Parameter	Description
Type	Queue type.
Name	Name of the single active consumer queue, which is user-defined.
Durability	Whether to enable persistence. Set Durability to Durable . <ul style="list-style-type: none">• Durable: This queue survives after server restart.• Transient: This queue will be deleted after server restart and needs to be recreated.
Node	Node where a single active consumer queue is deployed.
Auto delete	Whether to enable automatic deletion. <ul style="list-style-type: none">• Yes: This queue will be automatically deleted when the last consumer unsubscribes from the queue.• No: This queue will not be deleted when the last consumer unsubscribes from the queue.
(Optional) Arguments	Queue attribute. You do not need to set it.

Step 4 Click **Add queue**.

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

Figure 9-6 Viewing queue attributes

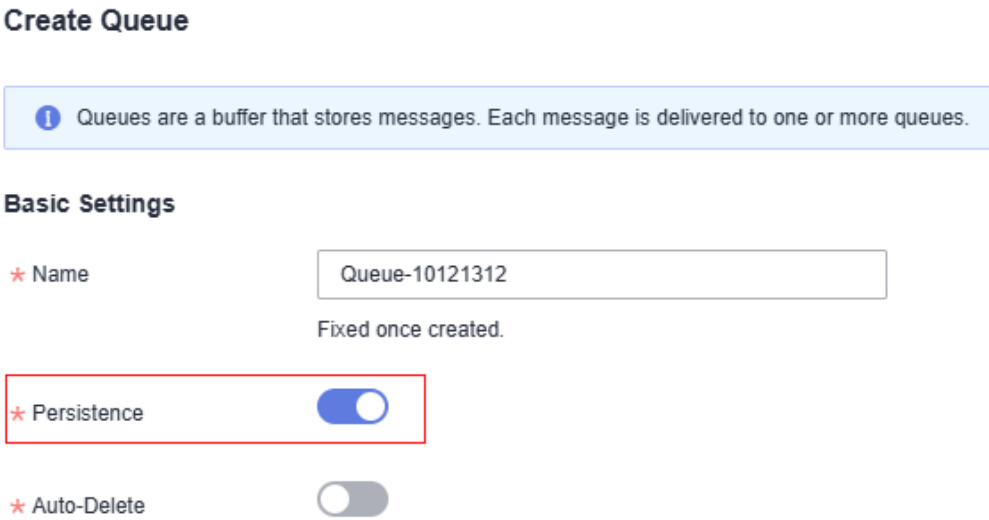


----End

Setting Queue Persistence on the RabbitMQ Console

Step 1 Create a queue by referring to [Creating a RabbitMQ Queue](#) and set the queue persistence, as shown in [Figure 9-7](#).

Figure 9-7 Configuring queue persistence (console)



Step 2 In the queue list, check whether persistence is enabled for the new queue. If **Yes** is displayed in the **Persistence** column, as shown in [Figure 9-8](#), persistence is enabled.

Figure 9-8 Queue persistence configured (console)

<input type="checkbox"/>	Name	Accumulated Messages	Persistence	Auto-Delete	Policy	Operation
<input type="checkbox"/>	Queue-25718751	0	Yes	No		Clear Message Delete

----End

Configuring Message Persistence

After configuring queue persistence, set **MessageProperties** to **PERSISTENT_TEXT_PLAIN** on the client to send persistent messages to the queue.

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

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

9.2 Configuring RabbitMQ TTL

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

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

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

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

Notes and Constraints

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

Setting Queue TTL on the RabbitMQ Console

- Step 1** Create a queue by referring to [Creating a RabbitMQ Queue](#) and set **Time to Live**.

Figure 9-9 Setting the time to live

Create Queue

Queues are a buffer that stores messages. Each message is delivered to one or more queues.

Basic Settings

Name

Queue06

Fixed once created.

Persistence

☒

Auto-Delete

☐

More Settings ^

Dead Letter Exchange

--Select--

Rejected and expired messages are resent to this exchange.

Dead Letter Routing Key

Enter a dead letter routing key.

The dead letter exchange sends dead letter messages to the queue with a binding key that corresponds to this routing key.

Time to Live

300

ms

How long messages published to queues can remain before being discarded.

Lazy Queue

Enter lazy to make the queue lazy.

Make the queue lazy to store more messages on disk and save memory. Otherwise, messages are stored in memory and delivered quickly.

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

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

Figure 9-10 Viewing the TTL

View Detail

Name	Queue06	Vhost	/
Accumulated Messages	0	Consumers	0
Automatically delete	No	Dead Letter Exchange	--
Dead Letter Routing Key	--	Time to Live(ms)	300
Lazy Queue	--		

----End

Setting Queue TTL on a Java Client

The **x-expires** parameter in the **channel.queueDeclare** argument is used to control how long a queue will remain active after being unused before it is

automatically deleted. "Unused" indicates that the queue has no consumer and is not re-declared, and the **Basic.Get** command is not called before expiration. The value of **x-expires** must be an integer other than 0, in milliseconds.

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

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

Configuring Message TTL

- In the queue configuration

Add the **x-message-ttl** parameter to the **channel.queueDeclare** argument. The value must be an integer other than 0, in milliseconds.

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

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

- For a dedicated message

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

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

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

10 Managing Instances

10.1 Viewing and Modifying Basic Information of a RabbitMQ Instance

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

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


Prerequisite

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

Viewing RabbitMQ Instance Details

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Specific RabbitMQ instances can be queried using filters. Current filters include the name, status, version, instance type, specification, used/available storage space, billing mode, AZ, instance ID, private connection address, public connection address, enterprise project, and resource tag.

- Only enterprise users can use enterprise projects to filter resources.
- [Table 10-1](#) describes the various possible statuses of a RabbitMQ instance.

Table 10-1 RabbitMQ instance status description

Status	Description
Creating	The instance is being created.
Creation failed	The instance failed to be created.
Running	The instance is running properly. Only instances in the Running state can provide services.
Faulty	The instance is not running properly.
Changing	The instance specifications are being changed.
Change failed	The instance specifications failed to be changed.
Binning	The instance is being moved to Recycle Bin.
Binned	The instance is in Recycle Bin.
Recovering	The instance is being recovered from Recycle Bin.

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

Table 10-2 describes the parameters for connecting to an instance. For details about other parameters, see the **Basic Information** tab page of the RabbitMQ instance on the console.


Table 10-2 Connection parameters


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

----End

Modifying Basic Information of a RabbitMQ Instance

Step 1 Log in to the console.









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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

Step 5 Modify the following parameters if needed:

Table 10-3 Modifiable RabbitMQ parameters

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

----End

10.2 Viewing RabbitMQ Client Connection Addresses

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

Notes and Constraints

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 10-1](#).

Figure 10-1 Client connection addresses

Connections

▼ All connections (4)

Pagination

Page 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	

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

A client can function as a producer to create messages and as a consumer to retrieve messages. The producer and consumer IP addresses are the same, as shown in [Figure 10-1](#), and are difficult to distinguish. To differentiate between producer and consumer IP addresses, you can set the **clientProperties** parameter on the client. The following is an example:

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

After the **clientProperties** parameter is set, the connection addresses are displayed as shown in [Figure 10-2](#).

Figure 10-2 Client connection addresses (with producer/consumer differentiated)

Connections

▼ All connections (2)

Pagination

Page 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	

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

----End

10.3 Viewing RabbitMQ Background Tasks


Performing an instance operation listed in [Table 10-4](#) starts a background task. The tasks are displayed on the **Background Tasks** page. Tasks can be cleared there.


Table 10-4 Background tasks

Task Name	Description
Creating an Instance	Creates a RabbitMQ instance.
Plug-in change	<ul style="list-style-type: none">• Enables a plug-in.• Disables a plug-in.
Enable public access	Enables public access.
Disable public access	Disables public access.
Modify Specifications	<ul style="list-style-type: none">• Increases the storage space.• Increases brokers.• Increases a broker flavor.• Decreases a broker flavor.

Procedure

Step 1 Log in to the console.

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


Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

Step 5 In the navigation pane on the left, choose **Instance > Background Tasks**.

Step 6 On the **Current Tasks** tab page, select a time period from the drop-down box, enter a keyword in the search box, and press **Enter**. Tasks started in the specified time period are displayed.

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

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

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

----End

10.4 Managing RabbitMQ Instance Tags

Tags facilitate RabbitMQ instance identification and management.

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

A tag consists of a tag key and a tag value. [Table 10-5](#) lists the tag key and value requirements.

Table 10-5 Tag key and value requirements


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


Notes and Constraints

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

Configuring Tags for a RabbitMQ Instance

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Click the desired instance to view its details.

Step 5 In the navigation pane, choose **Instance > Tags**.

Step 6 Click **Edit Tag**.

Step 7 Add, edit, or delete tags as required.

Table 10-6 Tag operations

Operation	Procedure
Adding a tag	<ol style="list-style-type: none">1. Click Add Tag to set tags with Tag key and Tag value. If you have predefined tags, select a predefined pair of tag key and value, and click Add. A maximum of 20 tags can be added.2. Click OK. View the new tag on the tag list page.
Editing a tag	Modify the tag key and value, and click OK . On the tag list page, view the new tag key and value.
Deleting a tag	In the row containing the tag to be deleted, click Delete . Then, click OK to delete the tag. The tags are deleted when they are no longer displayed in the tag list.

----End

10.5 Configuring RabbitMQ Recycling Policies

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


Recycling is disabled by default.


Constraints

- Pay-per-use instance in Recycle Bin will not generate fees, but their storage will.

Enabling Recycling

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 In the navigation pane, choose **Recycle Bin**.


Step 5 Click **Modify Recycling Policy** and the **Modify Recycling Policy** dialog box is displayed.


Step 6 Enable **Recycle Bin**, specify **Retention Days** (1–7), and click **OK**.

----End

Recovering RabbitMQ Instances

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 In the navigation pane, choose **Recycle Bin**.

Step 5 Recover RabbitMQ instances using either of the following methods:

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


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


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

----End

Modifying Retention Days

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 In the navigation pane, choose **Recycle Bin**.

Step 5 Click **Modify Recycling Policy** and the **Modify Recycling Policy** dialog box is displayed.

Step 6 Modify the retention days (1–7) and click **OK**.


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

----End

Exporting Instances in the Recycle Bin

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.


Step 4 In the navigation pane, choose **Recycle Bin**.


Step 5 Choose **Export > Export all data to an XLSX file** or **Export > Export selected data to an XLSX file**.

----End

Deleting Instances Permanently

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 In the navigation pane, choose **Recycle Bin**.

Step 5 Delete instances using either of the following methods:

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


Step 6 In the displayed **Delete Instance** dialog box, enter **DELETE** and click **OK**.


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

----End

Disabling Recycling

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 In the navigation pane, choose **Recycle Bin**.

Step 5 Click **Modify Recycling Policy** and the **Modify Recycling Policy** dialog box is displayed.

Step 6 Disable **Recycle Bin** and click **OK**.

----End

10.6 Resetting the RabbitMQ Instance Password


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


Prerequisite

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

Procedure

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

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.

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

----End

10.7 Enabling RabbitMQ Plug-ins

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

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

Table 10-7 lists plug-ins supported by RabbitMQ. **The ports of the plug-ins cannot be changed.**

Table 10-7 Plug-ins

Name	Function	Port
rabbitmq_amqp1_0	Support for AMQP 1.0	-
rabbitmq_delayed_message_exchange	Delayed messages 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	-

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

Notes and Constraints

- When plug-ins are enabled, the instance will not be restarted. However, enabling plug-ins rabbitmq_mqtt, rabbitmq_web_mqtt, rabbitmq_stomp, and rabbitmq_web_stomp will restart Keepalived and disconnect the instance. After the instance is disconnected, it may be automatically reconnected depending on the service logic.
- The rabbitmq_shovel, rabbitmq_federation, and rabbitmq_tracing plug-ins can be enabled only for specific instances. For details, see [Table 10-8](#).


Table 10-8 Instances for which plug-ins can be enabled


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

Instance	rabbitmq_shovel	rabbitmq_federation	rabbitmq_tracing
Cluster instances with SSL enabled	Not supported	Not supported	Not supported

Enabling RabbitMQ Plug-ins

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Click the desired instance to view its details.

Step 5 In the navigation pane, choose **Instance > Plug-ins**. Then, click **Enable** in the row containing the desired plug-in.

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

----End

10.8 Using the rabbitmq_tracing Plug-in

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

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

Operation Impact

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

Prerequisites

You have purchased an instance.

Using the rabbitmq_tracing Plug-in

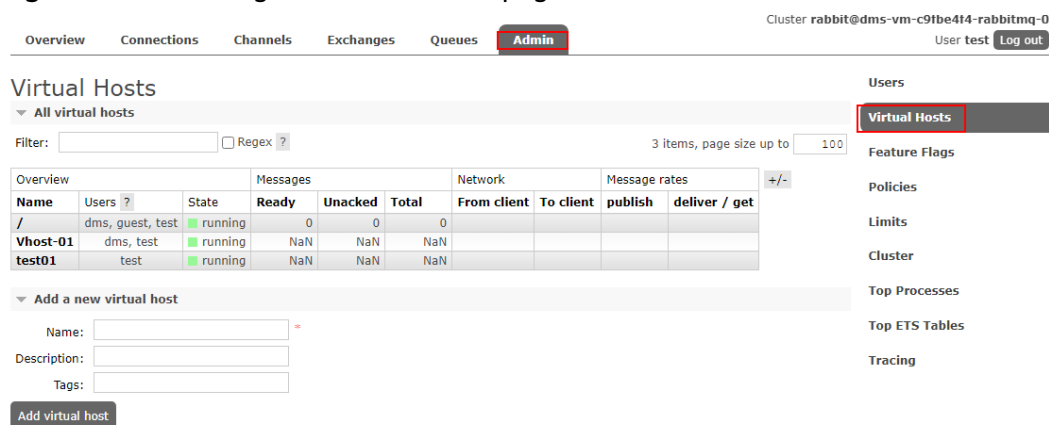
Step 1 Enable the rabbitmq_tracing plug-in by referring to [Enabling RabbitMQ Plug-ins](#).

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

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

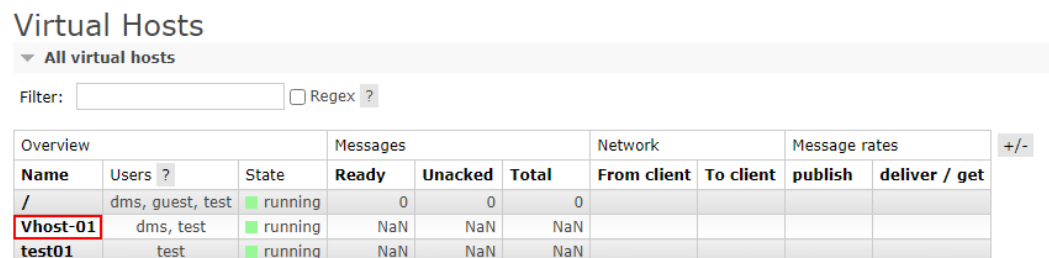
Step 4 In the navigation tree on the right, choose **Virtual Hosts**.

Figure 10-3 Entering the Virtual Hosts page



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

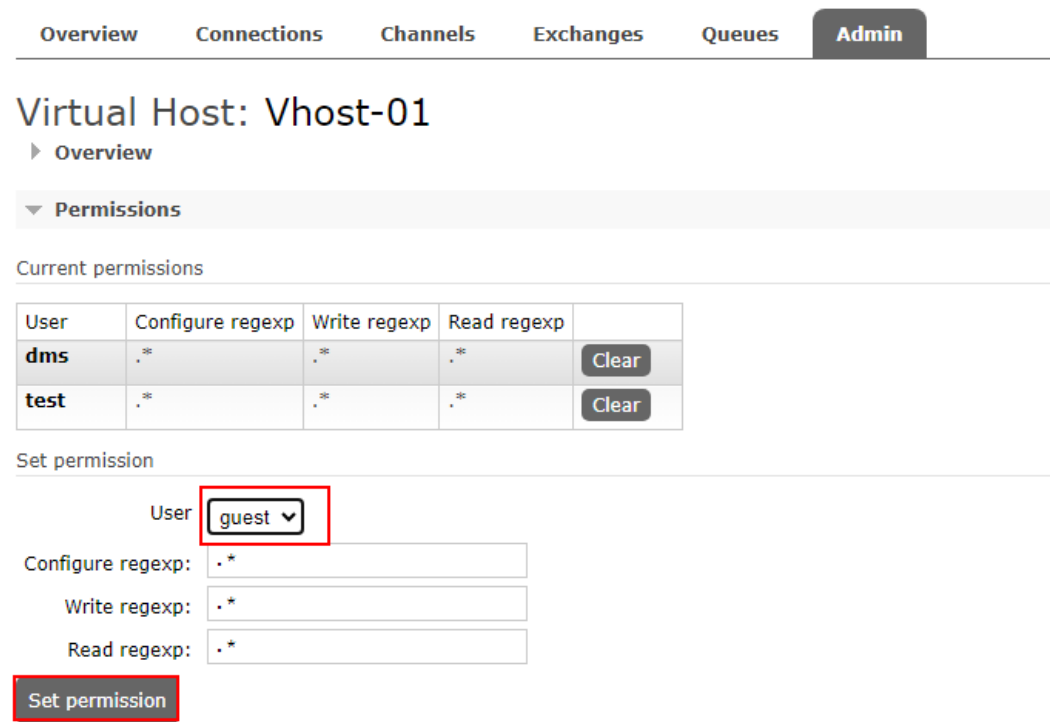
Figure 10-4 Virtual host to create a trace for



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

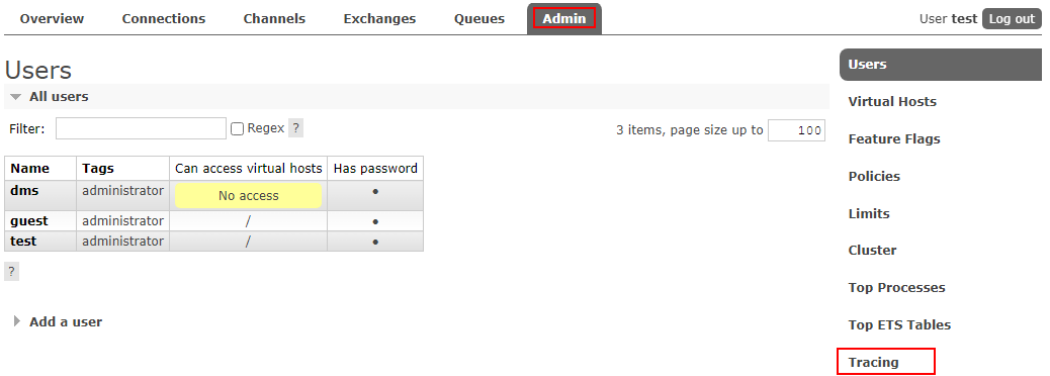
Virtual hosts must be configured the **guest** permission to enable tracing. Otherwise, an error is reported when a trace is created.

Figure 10-5 Granting the guest permission for a virtual host



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

Figure 10-6 Admin page



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

Table 10-9 Trace parameters

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

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

Figure 10-7 Adding a trace

▼ Add a new trace

Virtual host:

Name:

Format:

Tracer connection username:

Tracer connection password:

Max payload bytes: ?

Pattern:

Examples: #, publish.#, deliver.# #.amq.direct, #.myqueue

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

Figure 10-8 Trace list

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

Node:

▼ All traces

Currently running traces

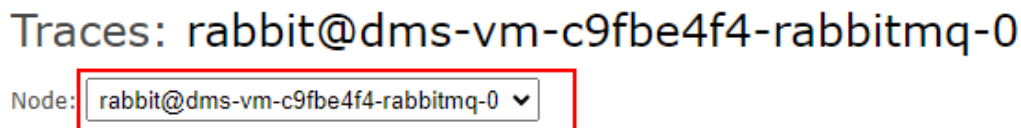
Virtual host	Name	Pattern	Format	Payload limit	Rate	Queued	Tracer connection username	
Vhost-01	delay_queue_trace	deliver.delay_queue	text	Unlimited		0 (queue)		<input type="button" value="Stop"/>
Vhost-01	delay_exchange_trace	publish.delay_exchange	text	Unlimited		0 (queue)		<input type="button" value="Stop"/>

Trace log files

Name	Size	
delay_queue_trace.log	0 B	<input type="button" value="Delete"/>
delay_exchange_trace.log	0 B	<input type="button" value="Delete"/>

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

Figure 10-9 Switching nodes



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

Figure 10-10 Trace log files

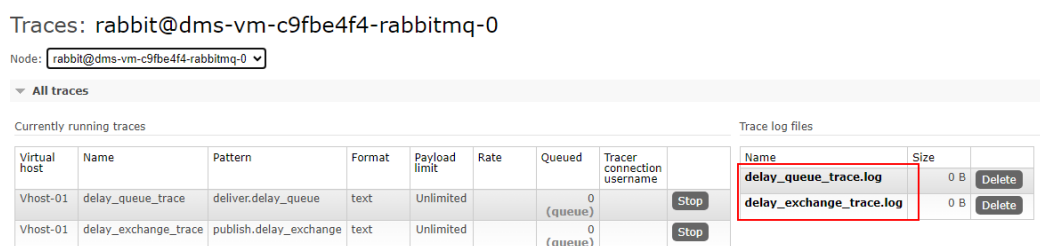


Figure 10-11 shows the content of **delay_exchange_trace.log**.

Figure 10-11 delay_exchange_trace.log

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

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

Figure 10-12 shows the content of **delay_queue_trace.log**.

Figure 10-12 delay_queue_trace.log

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

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

-----End
```

Stopping the Tracing Task and Disabling the Plug-in

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

Figure 10-13 Stopping a tracing task

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

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

▼ All traces

Currently running traces

Virtual host	Name	Pattern	Format	Payload limit	Rate	Queued	Tracer connection username	
Vhost-01	delay_queue_trace	deliver.delay_queue	text	Unlimited		0 (queue)		Stop
Vhost-01	delay_exchange_trace	publish.delay_exchange	text	Unlimited		0 (queue)		Stop

Trace log files

Name	Size	
delay_queue_trace.log	0 B	Delete
delay_exchange_trace.log	0 B	Delete

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

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


-----End


10.9 Exporting the RabbitMQ Instance List

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

Procedure

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Export the instance list in either of the following ways:

- Select the desired instances and choose **Export > Export selected data to an XLSX file** to export specified instances.
- Choose **Export > Export all data to an XLSX file** to export all instances.

----End

10.10 Deleting a RabbitMQ Instance

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


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


Prerequisite

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

Procedure

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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



WARNING

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

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

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

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

----End

10.11 Logging In to RabbitMQ Management UI



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

Prerequisite

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

Procedure

Step 1 Obtain the management address of an instance.

1. Log in to the console.
2. In the upper left corner, click  and select a region.
3. Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.
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**.

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

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

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

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

Figure 10-14 Logging in to the management UI



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

----End

11

Modifying Instance Specifications

11.1 Modifying RabbitMQ Instance Specifications

After creating a RabbitMQ instance, you can increase or decrease its specifications. For details, see [Table 11-1](#).

Table 11-1 Supported specification changes (for RabbitMQ 3.x.x)

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

Impact

Table 11-2 Impact of specification modification

Modified Object	Impact
Brokers	<ul style="list-style-type: none">During the broker increase, the RabbitMQ workload balance process is restarted, triggering a master/standby switchover and causing services to temporarily stutter. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.The change duration depends on the number of added brokers, and is 5–10 minutes per broker. The total change duration = Single change duration × Number of added brokers.
Storage space	<ul style="list-style-type: none">Storage space expansion does not affect services.You can expand the storage space 20 times for each instance.Available storage space = Actual storage space – Storage space for storing logs – Disk formatting loss For example, if the storage space is expanded to 700 GB, the storage space for storing logs is 100 GB, and the disk formatting loss is 7 GB, then the available storage space after capacity expansion will be 593 GB.The total change duration is within 5 minutes.

Modified Object	Impact
Broker flavor	<ul style="list-style-type: none">• For single-node RabbitMQ 3.x.x instances, nodes are restarted and services may temporarily stutter in minutes during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.• For cluster RabbitMQ 3.x.x instances not configured with mirrored/quorum queues, nodes are restarted in sequence and services may stutter in minutes during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.• For cluster RabbitMQ 3.x.x instances not configured with mirrored/quorum queues, nodes are restarted in sequence and services may stutter for several seconds during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.• During the modification of cluster RabbitMQ 3.x.x instances, connections to the changed nodes will be switched to other nodes, posing overload risks (for example, connections excess or memory high watermark) on them. You are advised to use them within the instance specifications. For more information, see Specifications.• Persistent exchanges, queues, and messages are required. Otherwise, messages may be lost after node restarts. For details, see Configuring RabbitMQ Persistence.• Brokers will be restarted in sequence during an instance change, which depends on the number of brokers. The change duration per single broker is 5–10 minutes. The total change duration = Single change duration × Number of brokers.

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

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

Notes and Constraints


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


Prerequisites

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

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

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 5 Specify the required storage space.


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


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

----End

Increasing the Broker Quantity (V3.x.x)

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

Step 5 Specify the required number of brokers.

1. For **Modify By**, select **Brokers**. Then, enter the number of brokers. The broker quantity range varies by instance specifications. For details, see

Specifications. If public access has been enabled, configure EIPs for the new brokers. Then click **Next**.


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


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

----End

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

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

Step 4 Modify the instance specifications in either of the following ways:

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

Step 5 Specify the required broker flavor.

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

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

----End

12 Migrating RabbitMQ Services

There are two scenarios for migrating RabbitMQ services:

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

Migration Principle

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

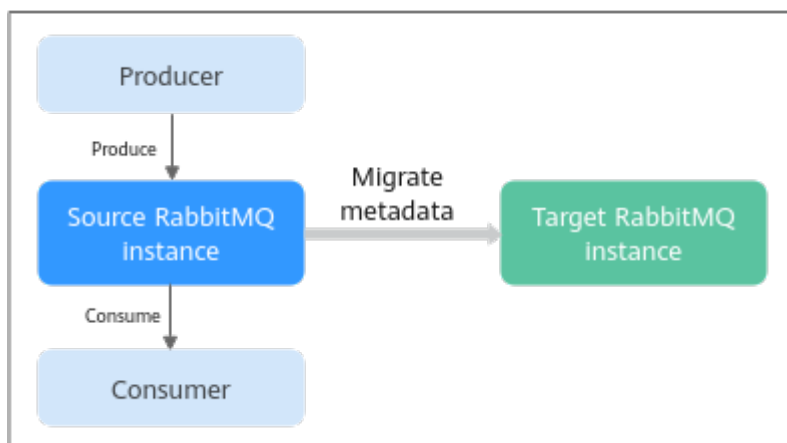
Prerequisite

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

Implementation (Dual-Read)

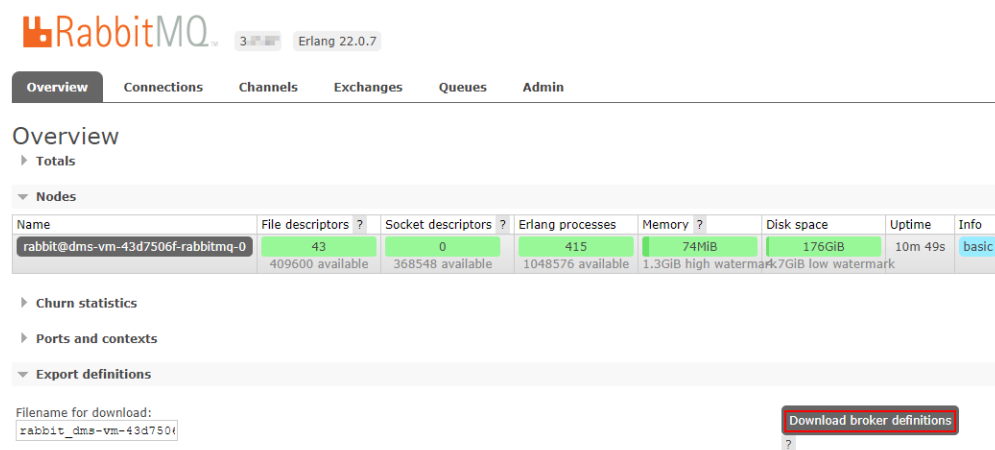
Step 1 Migrate source RabbitMQ instance metadata to a target RabbitMQ instance.

Figure 12-1 Migrating metadata

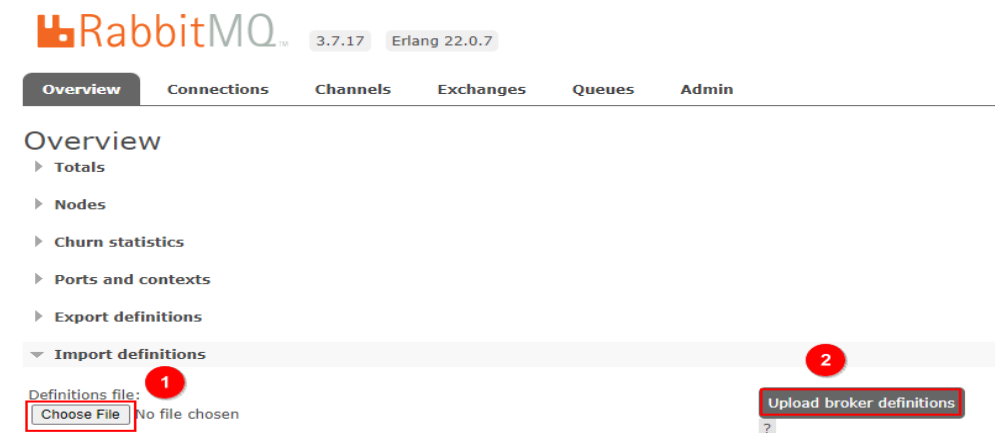


Do as follows:

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

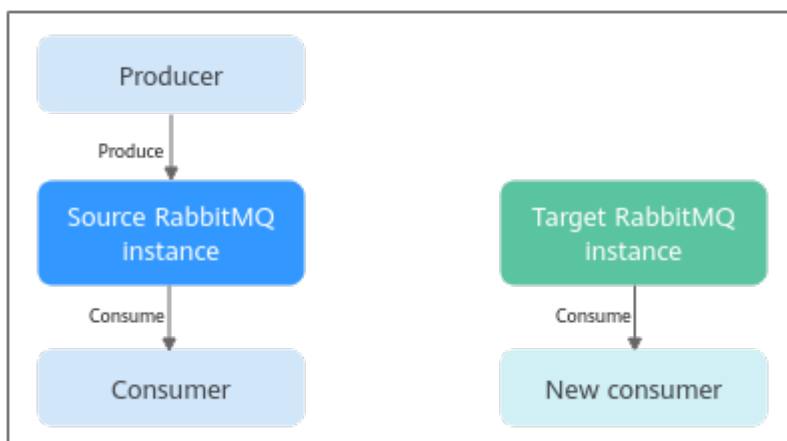
Figure 12-2 Exporting metadata

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

Figure 12-3 Importing metadata

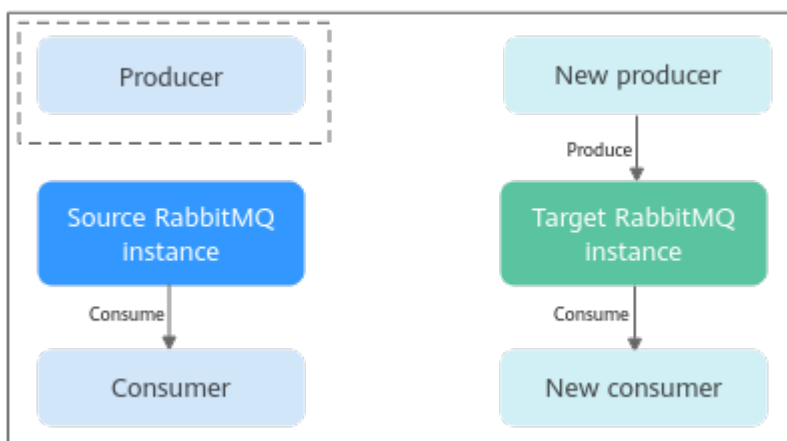
Step 2 Add new consumers for the target RabbitMQ instance.

Figure 12-4 Adding new consumers



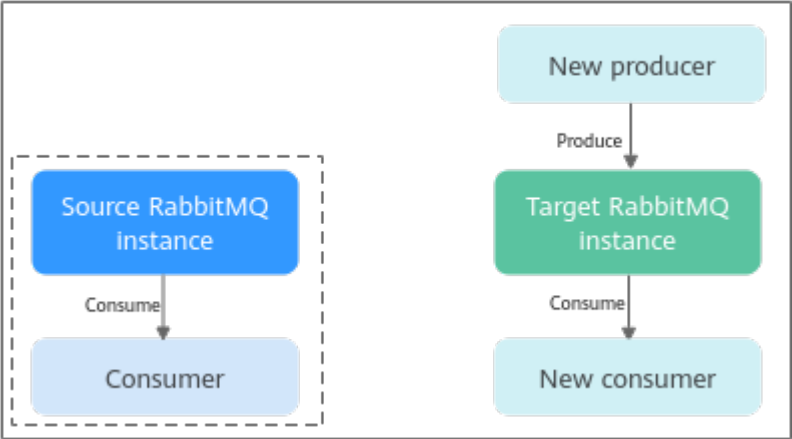
Step 3 Add new producers for the target RabbitMQ instance and remove the producers of the source RabbitMQ instance. The old consumers continue consuming messages from the source RabbitMQ instance.

Figure 12-5 Migrating producers



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

Figure 12-6 Removing an old consumer and a source RabbitMQ instance



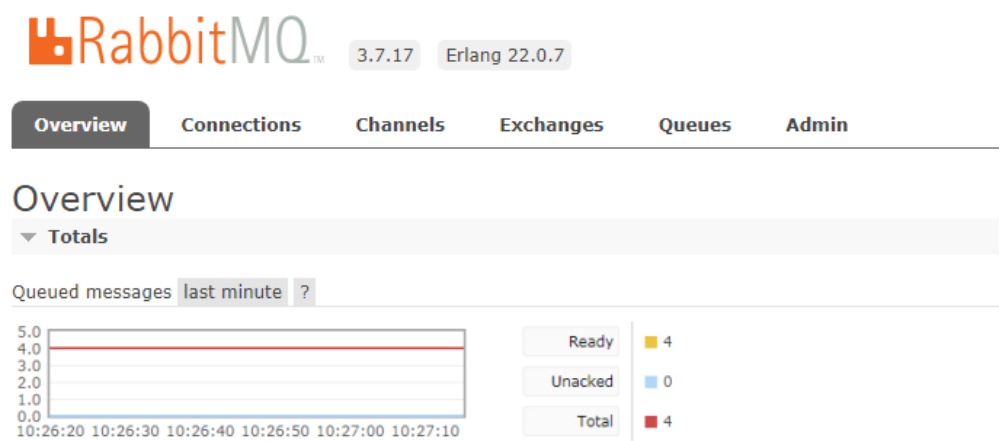
----End

Check After Migration

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

- Using the RabbitMQ management UI, as shown in Figure 12-7.
On the **Overview** tab page, if the number of messages that can be consumed (**Ready**) and the number of messages that are not acknowledged (**Unacked**) are both 0, the consumption is complete.

Figure 12-7 RabbitMQ management UI



- Calling an API
`curl -s -u username:password -XGET http://ip.port/api/overview`

Table 12-1 Parameters

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

Parameter	Description
password	Password of the source instance to log in to the RabbitMQ management UI
ip	IP address of the source instance to log in to the RabbitMQ management UI
port	Port of the source instance to log in to the RabbitMQ management UI

The consumption is complete when **messages_ready** and **messages_unacknowledged** values in the command output are both 0.

Figure 12-8 Command output

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



13 Applying for Increasing RabbitMQ Quotas

What Is Quota?

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

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

How Do I View My Quotas?

1. Log in to the console.
2. Click  in the upper left corner of the console and select a region and a project.
3. Click  (the **My Quota** icon) in the upper right corner.
The **Quotas** page is displayed.
4. On the **Quotas** page, view the used and total quotas of each type of resources.

If a quota cannot meet your service requirements, apply for more quotas.

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.

14 Viewing Metrics and Configuring Alarms

14.1 Viewing RabbitMQ Metrics


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


Prerequisites

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

Viewing RabbitMQ Metrics

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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

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

name". For example, if the **test01** queue is on **Vhost-13142708**, the queue name displayed on the monitoring page is **Vhost-13142708_test01**.

----End

14.2 RabbitMQ Metrics

Introduction

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

Namespace

SYS.DMS

Instance Metrics

Table 14-1 Instance metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
connections	Connections	Number of connections in the RabbitMQ instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels	Channels	Number of channels in the RabbitMQ instance	0-2047	Count	N/A	RabbitMQ instance	1 minute
queues	Queues	Number of queues in the RabbitMQ instance	0-7,000	Count	N/A	RabbitMQ instance	1 minute
consumers	Consumers	Number of consumers in the RabbitMQ instance	0-280,000	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
messages_ready	RabbitMQ instance Available Messages	Number of messages that can be consumed in the RabbitMQ instance	0–10,000,000	Count	N/A	RabbitMQ instance	1 minute
messages_unacknowledged	Unacknowledged Messages	Total number of messages that have been consumed but not acknowledged in a RabbitMQ instance	0–10,000,000	Count	N/A	RabbitMQ instance	1 minute
publish	Production Rate	Rate at which messages are produced in the RabbitMQ instance	0–25,000	Count/s	N/A	RabbitMQ instance	1 minute
deliver	Retrieval Rate (Manual Ack)	Rate at which messages are consumed (in the manual acknowledgment scenario) in a RabbitMQ instance	0–25,000	Count/s	N/A	RabbitMQ instance	1 minute
deliver_no_ack	Retrieval Rate (Auto Ack)	Rate at which messages are consumed (in the automatic acknowledgment scenario) in a RabbitMQ instance	0–50,000	Count/s	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
connections_state_running	Normal Connections	Number of starting , tuning , opening , and running connections in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
connections_state_flow	Flow Connections	Number of flow connections in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
connections_state_blocking	Blocked/Blocking Connections	Number of blocking and blocked connections in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
connections_state_closing	Closed/Closing Connections	Number of closing and closed connections in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels_states_running	Normal Channels	Number of starting , tuning , opening , and running channels in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels_states_flow	Flow Channels	Number of flow channels in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
channels_states_blocking	Blocked/Blocking Channels	Number of blocking and blocked channels in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
channels_states_close	Closed/Closing Channels	Number of closing and closed channels in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
queues_states_running	Normal Queues	Number of running queues in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute
queues_states_flow	Flow Queues	Number of flow queues in the instance	≥ 0	Count	N/A	RabbitMQ instance	1 minute

Broker Metrics

Table 14-2 Broker metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
fd_used	File Handles	Number of file handles used by RabbitMQ in the node	0–65,535	Count	N/A	RabbitMQ instance node	1 minute
socket_used	Socket Connections	Number of socket connections used by RabbitMQ in the node	0–50,000	Count	N/A	RabbitMQ instance node	1 minute
proc_used	Erlang Processes	Number of Erlang processes used by RabbitMQ in the node	0–1,048,576	Count	N/A	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
mem_used	Memory Usage	Memory usage of RabbitMQ in the node	0–32,000,000,000	Byte	1024(IEC)	RabbitMQ instance node	1 minute
disk_free	Available Memory	Available memory of RabbitMQ in the node	0–500,000,000,000	Byte	1024(IEC)	RabbitMQ instance node	1 minute
rabbitmq_alive	Node Alive	Whether the RabbitMQ node is alive	1: alive 0: not alive	N/A	N/A	RabbitMQ instance node	1 minute
rabbitmq_disk_usage	Disk Capacity Usage	Disk usage of the RabbitMQ VM	0~100	%	N/A	RabbitMQ instance node	1 minute
rabbitmq_cpu_usage	CPU Usage	CPU usage of the RabbitMQ VM	0~100	%	N/A	RabbitMQ instance node	1 minute
rabbitmq_cpu_core_load	Average Load per CPU Core	Average load of each CPU core of the RabbitMQ VM	> 0	N/A	N/A	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
rabbitmq_memory_usage	Memory usage of the RabbitMQ VM	Memory usage of the RabbitMQ VM <ul style="list-style-type: none">Before September 2025, Memory usage = Used memory of the VM/ Total memory of the VMSeptember 2025 and later, Memory usage = Memory used by the RabbitMQ process/ Total memory of the VM	0~100	%	N/A	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	> 0	ms	N/A	RabbitMQ instance node	1 minute
rabbitmq_disk_write_await	Average Disk Write Time	Average time for each disk I/O write in the monitoring period	> 0	ms	N/A	RabbitMQ instance node	1 minute
rabbitmq_node_bytes_in_rate	Inbound Traffic	Inbound traffic per second	> 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
rabbitmq_node_bytes_out_rate	Outbound Traffic	Outbound traffic per second	> 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute
rabbitmq_node_queues	Queues	Number of queues in the node	> 0	Count	N/A	RabbitMQ instance node	1 minute
rabbitmq_memory_high_watermark	Memory High Watermark	Whether the node has reached the memory high watermark, blocking all producers in the cluster	1: yes 0: no	N/A	N/A	RabbitMQ instance node	1 minute
rabbitmq_disk_insufficient	Disk High Watermark	Whether the node has reached the disk high watermark, blocking all producers in the cluster	1: yes 0: no	N/A	N/A	RabbitMQ instance node	1 minute
rabbitmq_disk_read_rate	Disk Read Speed	Number of bytes read from the disk of the node each second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute
rabbitmq_disk_write_rate	Disk Write Speed	Number of bytes written to the disk of the node each second	≥ 0	Byte/s	1024(IEC)	RabbitMQ instance node	1 minute
connections_usage	Connection Usage	Percentage of current connections to the maximum number of connections	≥ 0	%	N/A	RabbitMQ instance node	1 minute

Queue Metrics

Table 14-3 Queue metrics

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_messages_unacknowledged	Unacknowledged Messages	Number of messages that have been consumed but not acknowledged in the RabbitMQ queue	0–10,000,000	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_ready	Queue Available Messages	Number of messages that can be retrieved in a RabbitMQ queue	0–10,000,000	Count	N/A	RabbitMQ instance queue	1 minute
queue_consumers	Consumers	Number of consumers that are subscribed to the queue	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_publish_rate	Production Rate	Number of messages sent to the queue each second	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute
queue_messages_ack_rate	Consumption Rate (Manual)	Number of acknowledged messages sent from the queue to clients each second	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute

Metric ID	Metric Name	Description	Value Range	Unit	Conversion Rule	Monitored Object (Dimension)	Monitoring Period (Raw Data)
queue_messages_deliver_get_rate	Consumption Rate	Number of messages sent from the queue each second	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute
queue_messages_redeliver_rate	Redelivery Rate	Number of messages in the queue redelivered each second	≥ 0	Count/s	N/A	RabbitMQ instance queue	1 minute
queue_messages_persistent	Total Persisted Messages	Total number of persisted messages in the queue. This is always 0 for transient queues.	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_messages_ram	Total Messages in RAM	Total number of messages in the queue that are kept in RAM	≥ 0	Count	N/A	RabbitMQ instance queue	1 minute
queue_memory	Bytes Consumed by Erlang	Bytes of memory consumed by the Erlang process associated with the queue, including stack, heap, and internal structures	≥ 0	Byte	1024(IEC)	RabbitMQ instance queue	1 minute
queue_message_bytes	Total Message Size	Total number of bytes of all messages in the queue	≥ 0	Byte	1024(IEC)	RabbitMQ instance queue	1 minute

Dimensions

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

14.3 Configuring RabbitMQ Alarms

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

Table 14-4 RabbitMQ instance metrics and alarm policies

Metric	Alarm Policy	Description	Solution
Memory High Watermark	Alarm threshold: Raw data ≥ 1 Number of consecutive periods: 1 Alarm severity: Critical	A threshold of 1 indicates that the memory high watermark is reached, blocking message publishing.	<ul style="list-style-type: none">Accelerate message retrieval.Use publisher confirms and monitor the publishing rate and duration on the publishing end. When the duration increases significantly, apply flow control.
Disk High Watermark	Alarm threshold: Raw data ≥ 1 Number of consecutive periods: 1 Alarm severity: Critical	A threshold of 1 indicates that the disk high watermark is reached, blocking message publishing.	<ul style="list-style-type: none">Reduce the number of messages accumulated in lazy queues.Reduce the number of messages accumulated in durable queues.Delete queues.

Metric	Alarm Policy	Description	Solution
Memory Usage	Alarm threshold: Raw data > Expected usage (30% is recommended) Number of consecutive periods: 3-5 Alarm severity: Major	To prevent high memory watermarks from blocking publishing, configure an alarm for this metric on each node.	<ul style="list-style-type: none"> Accelerate message retrieval. Use publisher confirms and monitor the publishing rate and duration on the publishing end. When the duration increases significantly, apply flow control.
CPU Usage	Alarm threshold: Raw data > Expected usage (70% is recommended) Number of consecutive periods: 3-5 Alarm severity: Major	A high CPU usage may slow down publishing rate. Configure an alarm for this metric on each node.	<ul style="list-style-type: none"> Reduce the number of mirrored queues. For a cluster instance, add nodes and rebalance queues between all nodes.
Available Messages	Alarm threshold: Raw data > Expected number of available messages Number of consecutive periods: 1 Alarm severity: Major	If the number of available messages is too large, messages are accumulated.	See the solution to preventing message accumulation .
Unacked Messages	Alarm threshold: Raw data > Expected number of unacknowledged messages Number of consecutive periods: 1 Alarm severity: Major	If the number of unacknowledged messages is too large, messages may be accumulated.	<ul style="list-style-type: none"> Check whether the consumer is abnormal. Check whether the consumer logic is time-consuming.

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


 NOTE

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

Configuring RabbitMQ Alarms

Step 1 Log in to the console.

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

Step 3 Click  in the upper left corner and choose **Application > Distributed Message Service for RabbitMQ** to open the RabbitMQ instance list.

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

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



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

Step 6 Specify the alarm rule details.

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

1. Enter the alarm name and description.
2. Specify the alarm policy and alarm severity.
For example, an alarm can be triggered and notifications can be sent once every day if the raw value of connections exceeds the preset value for three consecutive periods and no actions are taken to handle the exception.
3. Set **Alarm Notification** configurations. If you enable **Alarm Notification**, set the validity period, notification object, and trigger condition.
4. Click **Create**.

----End

15 Viewing RabbitMQ Audit Logs

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

Prerequisite

CTS has been enabled.

DMS for RabbitMQ Operations Supported by CTS

Table 15-1 DMS for RabbitMQ operations supported by CTS

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

Operation	Resource Type	Trace Name
Successfully deleting multiple instances at a time	rabbitmq	batchDeleteDMSInstanceSuccess
Failing to delete multiple instances at a time	rabbitmq	batchDeleteDMSInstanceFailure
Successfully modifying instance information	rabbitmq	modifyDMSInstanceInfoSuccess
Failing to modify instance information	rabbitmq	modifyDMSInstanceInfoFailure
Successfully deleting multiple instance tasks at a time	rabbitmq	batchDeleteDMSInstanceTask
Successfully deleting an instance task	rabbitmq	deleteDMSInstanceTaskSuccess
Failing to delete an instance task	rabbitmq	deleteDMSInstanceTaskFailure
Successfully creating an instance task	rabbitmq	createDMSInstanceTaskSuccess
Failing to create an instance task	rabbitmq	createDMSInstanceTaskFailure
Successfully submitting a request for scaling up an instance	rabbitmq	extendDMSInstanceTaskSuccess
Failing to submit a request for scaling up an instance	rabbitmq	extendDMSInstanceTaskFailure
Successfully submitting a request for modifying instance information	rabbitmq	modifyDMSInstanceInfoTaskSuccess
Failing to submit a request for modifying instance information	rabbitmq	modifyDMSInstanceInfoTaskFailure
Successfully restoring the instance from Recycle Bin	rabbitmq	out_recycleTaskSuccess
Failing to restore the instance from Recycle Bin	rabbitmq	out_recycleTaskFailure

Viewing Audit Logs

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

16 FAQs

16.1 Instances

16.1.1 What RabbitMQ Version Does DMS for RabbitMQ Use?

On the RabbitMQ server: 3.8.35

16.1.2 What SSL Version Does DMS for RabbitMQ Use?

TLS 1.2.

16.1.3 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 about how to add user permissions, see [Modifying User Group Permissions](#).

16.1.4 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.

16.1.5 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.

16.1.6 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.

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

When SSL is enabled, DMS for RabbitMQ 3.x.x uses one-way authentication, and does not involve any certificates.

16.1.8 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.

16.1.9 Can RabbitMQ Instances Be Scaled Up?

Single-node RabbitMQ instances: The storage space can be increased.

Cluster RabbitMQ instances: The storage space and the broker quantity can be increased.

16.1.10 Does RabbitMQ Support Two-Way Authentication?

No.

16.1.11 Does DMS for RabbitMQ Support CPU and Memory Upgrades?

No.

16.1.12 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.

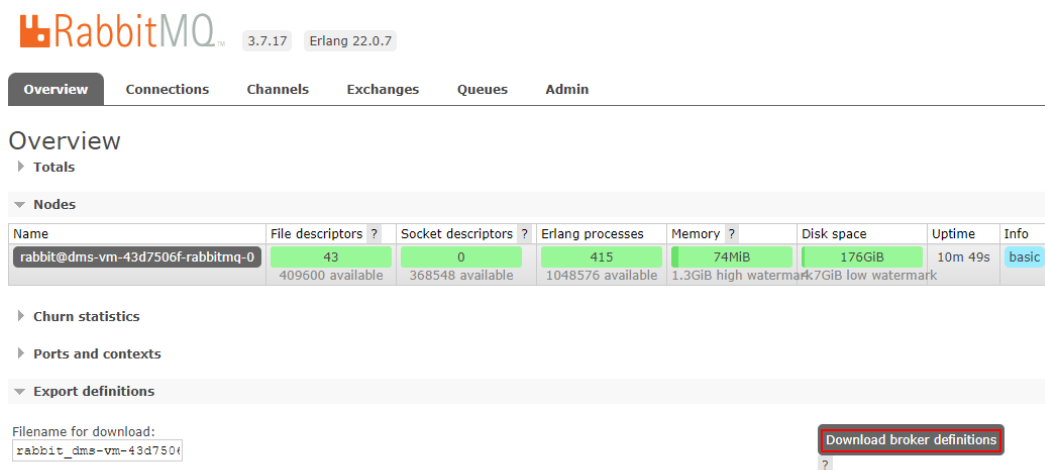
16.1.13 Can I Change the AZ for an Instance?

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

To migrate RabbitMQ 3.x.x instance metadata, perform the following steps:

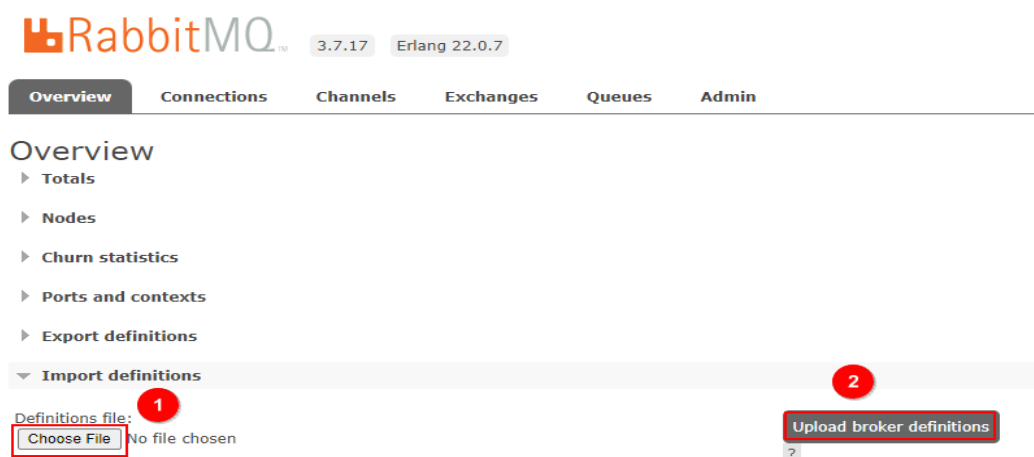
Step 1 [Log in to the management UI of the purchased RabbitMQ instance.](#)

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

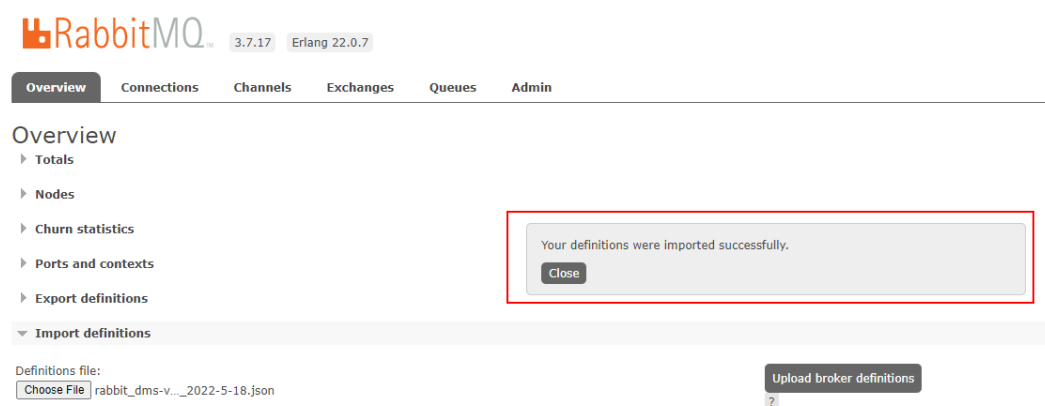


Step 3 Log in to the management UI of the new instance. On the **Overview** tab page, click **Choose File** and select the metadata exported from [Step 2](#).

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



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



----End

16.1.14 How Do I Obtain the Region ID?

To obtain the region ID, do as follows:

Step 1 Go to the [Regions and Endpoints](#) page.

Step 2 Obtain region IDs from the **Region** column.

----End

16.1.15 Why Can't I Select Two AZs?

Because there are split-brain risks. For higher reliability, you are advised to select three or more AZs when creating a cluster RabbitMQ instance.

16.1.16 How to Change Single-node RabbitMQ Instances to Cluster Ones?

You cannot perform such a change. To use Cluster RabbitMQ instances, create ones and migrate your services to them.

16.1.17 Can I Change the VPC and Subnet After a RabbitMQ Instance Is Created?

No.

16.1.18 Does Specification Modification Affect Services?

Yes. See [Table 16-1](#).

Table 16-1 Impact of specification modification

Modified Object	Impact
Brokers	<ul style="list-style-type: none">During the broker increase, the RabbitMQ workload balance process is restarted, triggering a master/standby switchover and causing services to temporarily stutter. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.The change duration depends on the number of added brokers, and is 5–10 minutes per broker. The total change duration = Single change duration × Number of added brokers.

Modified Object	Impact
Storage space	<ul style="list-style-type: none">Storage space expansion does not affect services.You can expand the storage space 20 times for each instance.Available storage space = Actual storage space – Storage space for storing logs – Disk formatting loss For example, if the storage space is expanded to 700 GB, the storage space for storing logs is 100 GB, and the disk formatting loss is 7 GB, then the available storage space after capacity expansion will be 593 GB.The total change duration is within 5 minutes.
Broker flavor	<ul style="list-style-type: none">For single-node RabbitMQ 3.x.x instances, nodes are restarted and services may temporarily stutter in minutes during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.For cluster RabbitMQ 3.x.x instances not configured with mirrored/quorum queues, nodes are restarted in sequence and services may stutter in minutes during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.For cluster RabbitMQ 3.x.x instances not configured with mirrored/quorum queues, nodes are restarted in sequence and services may stutter for several seconds during the modification. Ensure that your client can auto-reconnect. Modify specifications during off-peak hours.During the modification of cluster RabbitMQ 3.x.x instances, connections to the changed nodes will be switched to other nodes, posing overload risks (for example, connections excess or memory high watermark) on them. You are advised to use them within the instance specifications. For more information, see Specifications.Persistent exchanges, queues, and messages are required. Otherwise, messages may be lost after node restarts. For details, see Configuring RabbitMQ Persistence.Brokers will be restarted in sequence during an instance change, which depends on the number of brokers. The change duration per single broker is 5–10 minutes. The total change duration = Single change duration × Number of brokers.

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

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

16.2 Connections

16.2.1 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 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** 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 16-2 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 16-3 Security group rule

Direction	Protocol & Port	Source
Inbound	All	sg-53d4

- Public access:

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

Table 16-4 Security group rule

Direction	Protocol & Port	Source
Inbound	TCP:5672	IP address or IP address group of the RabbitMQ client

16.2.2 Why Does a Client Fail to Connect to a RabbitMQ Instance?

This problem occurs when the connection address, port number, username, password, or virtual host name is incorrect, or when there is no virtual host or the maximum allowed number of connections is exceeded.

Possible Cause 1: Incorrect Connection Address

Error message displayed when an incorrect connection address is used during intra-VPC access:

```
[root@ecs-test 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 when an incorrect connection address is used during public access:

```
[root@ecs-test 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)
```

Solution: On the **Overview** page of the RabbitMQ console, obtain the private or public network connection address and modify the connection address in the code.

Possible Cause 2: Incorrect Port

Error message displayed when an incorrect port is used during intra-VPC access:

```
[root@ecs-test 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 when an incorrect port is used during public access:

```
[root@ecs-test 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)
```

Solution: Change the port number.

Possible Cause 3: Incorrect Username or Password

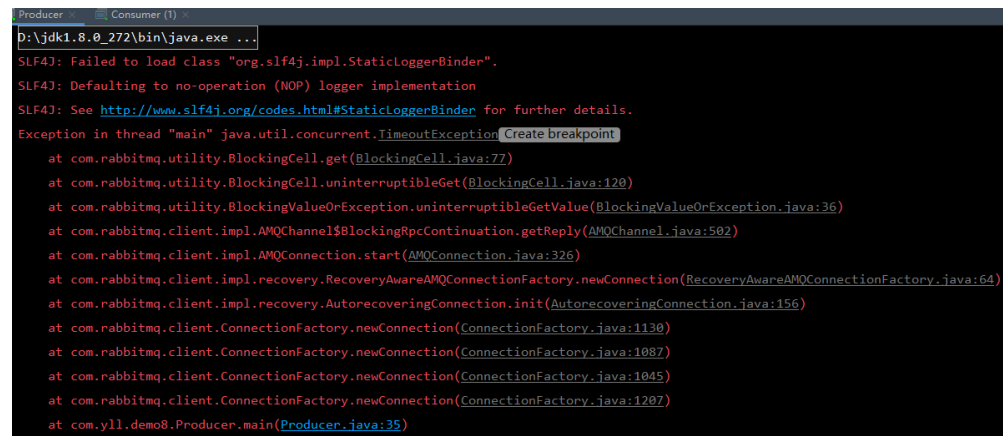
The error information is as follows:

```
[root@ecs-test 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(RecoveryAwareAMQ
ConnectionFactory.java:64)
```

Solution: Change the username or password. If you forget your password, [reset it](#).

Possible Cause 4: Maximum Number of Connections Exceeded

The error information is as follows:



```
Producer - Consumer (1)
D:\jdk1.8.0_272\bin\java.exe ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Exception in thread "main" java.util.concurrent.TimeoutException Create breakpoint
at com.rabbitmq.utility.BlockingCell.get(BlockingCell.java:77)
at com.rabbitmq.utility.BlockingCell.uninterruptibleGet(BlockingCell.java:120)
at com.rabbitmq.utility.BlockingValueOrException.uninterruptibleGetValue(BlockingValueOrException.java:36)
at com.rabbitmq.client.impl.AMQChannel$BlockingRpcContinuation.getReply(AMQChannel.java:502)
at com.rabbitmq.client.impl.AMQConnection.start(AMQConnection.java:326)
at com.rabbitmq.client.impl.recovery.RecoveryAwareAMQConnectionFactory.newConnection(RecoveryAwareAMQConnectionFactory.java:64)
at com.rabbitmq.client.impl.recovery.AutorecoveringConnection.init(AutorecoveringConnection.java:156)
at com.rabbitmq.client.ConnectionFactory.newConnection(ConnectionFactory.java:1130)
at com.rabbitmq.client.ConnectionFactory.newConnection(ConnectionFactory.java:1087)
at com.rabbitmq.client.ConnectionFactory.newConnection(ConnectionFactory.java:1045)
at com.rabbitmq.client.ConnectionFactory.newConnection(ConnectionFactory.java:1207)
at com.yll.demo8.Producer.main(Producer.java:35)
```

Solution: Close unused connections.

Possible Cause 5: No Virtual Host Available or Incorrect Virtual Host Name

The error information is as follows:

```
Couldn't log in: server connection error 530, message: NOT_ALLOWED - vhost /localdev/ not found
```

Solutions:

- If no virtual host is available, go to the **Instance > Virtual Hosts** page of the RabbitMQ console and create one.
- If the virtual host name is incorrect, modify the connection URL and configuration file according to the virtual host name displayed on the **Instance > Virtual Hosts** page of the RabbitMQ console.

16.2.3 Does DMS for RabbitMQ Support Public Access?

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

16.2.4 Does DMS for RabbitMQ Support Cross-Region Deployment?

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

16.2.5 Do RabbitMQ Instances Support Cross-VPC Access?

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

For details about VPC peering connections, see [VPC Peering Connection](#).

16.2.6 Do RabbitMQ Instances Support Cross-Subnet Access?

Yes.

If the client and the instance are in the same VPC, cross-subnet access is supported. By default, subnets in the same VPC can communicate with each other.

16.2.7 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();
```

16.2.8 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.

16.2.9 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 **Overview** page, click the name of the security group.
2. Click the **Inbound Rules** tab to view the inbound rules of the security group.
 - Allow access to port 15672 if SSL is disabled.
 - Allow access to port 15671 if SSL is enabled.

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

Yes.

Virtual hosting is a basic feature of RabbitMQ. Each virtual host (vhost) serves as an independent RabbitMQ server. Different vhosts have different data directories

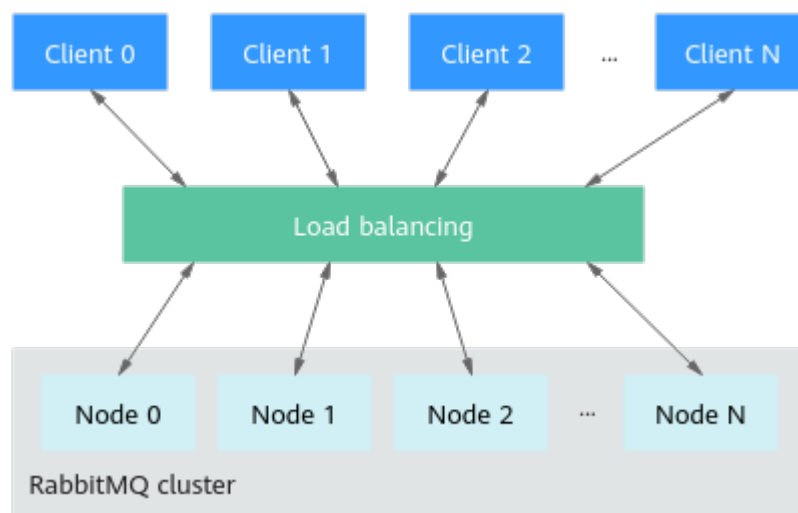
but share the same process. Connecting to multiple vhosts does not differ much in performance from connecting to one vhost. The only difference is that the RabbitMQ process has more objects. You are advised to test the performance by using the service model.

For details, see [Virtual Hosts](#) on the official RabbitMQ website.

16.2.11 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.

Figure 16-1 Connections



16.2.12 Do RabbitMQ Instances Support the Ping Command?

Certain RabbitMQ instances do. Specifically:

- Single-node instances support the **ping** command with both private and public connection addresses.
- Cluster instances only support the **ping** command with private connection addresses.

16.3 Messages

16.3.1 Does DMS for RabbitMQ Support Delayed Message Delivery?

You are suggested to use the scheduled/delayed messages of DMS for RocketMQ.

16.3.2 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 confirmation 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.

The screenshot shows the RabbitMQ Management UI for the 'amq.topic' exchange. The 'Publish message' section is expanded, showing fields for 'Routing key', 'Delivery mode' (set to '1 - Non-persistent'), 'Headers', and 'Properties'. The 'Properties' section is highlighted with a red box, showing the 'expiration' property set to '1000'. Below the properties is a 'Payload' text area and a 'Publish message' button.

- 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.

RabbitMQ 3.7.17 Erlang 22.0.7

Overview Connections Channels **Exchanges** Queues Admin

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:

Durability:

Auto delete: ?

Internal: ?

Arguments: =

=

Add Alternate exchange ?

16.3.3 How Long Are Messages Be Retained?

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

16.3.4 Where Is Message Creation Time Set?

The message creation time is set by the producer during message production.

16.3.5 What Is the Maximum Size of a Message that Can be Created?

50 MB. Larger messages will fail to be produced.

16.4 Monitoring & Alarm

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

RabbitMQ 3.x.x Instance

Monitoring data may fail to be displayed if the queue name or virtual host name of a RabbitMQ 3.x.x instance meets any of the following conditions:

- The queue name starts with a special character, for example, a period (.). You are advised to delete queues whose names contain special characters.
- The virtual host name starts with a special character, for example, a period (.). You are advised to delete virtual hosts whose names contain special characters. The queue name of a RabbitMQ 3.x.x instance is displayed in two ways on the monitoring page. The name of a queue is displayed if the queue is on the default virtual host. If a queue is not on the default virtual host, the queue name is displayed in the format "*Name of the virtual host where the queue is*__*Queue name*". For example, if the **test01** queue is on **Vhost-13142708**, the queue name displayed on the monitoring page is **Vhost-13142708__test01**.
- On Cloud Eye, a special character (.) is counted as five characters. If the name of a queue or virtual host contains periods (.) and the counted total length exceeds 256 characters, the monitoring data cannot be displayed. You are advised to delete queues or virtual hosts whose names contain periods (.).

When the queue or virtual host name of a RabbitMQ 3.x.x instance contains special characters, such as %, |, and /, the special characters are displayed as underscores (_) on the monitoring page. For example, the name of queue Queue.1%1|2_3/ in the default virtual host is displayed as **Queue.1_1_2_3_** on the monitoring page.

16.4.2 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
2025-08-29	This issue incorporates the following changes: <ul style="list-style-type: none">Added the one-click purchase of an instance with the same configuration in section Buying a RabbitMQ Instance.Optimized the navigation pane on the console. See Creating a RabbitMQ Virtual Host.
2025-04-30	This issue incorporates the following changes: <ul style="list-style-type: none">Added the recycle bin function. See Configuring RabbitMQ Recycling Policies.Redesigned the purchase page. See Buying a RabbitMQ Instance.Discontinued the RabbitMQ restart function.
2024-06-26	This issue incorporates the following change: <ul style="list-style-type: none">Added tag policies in Buying a RabbitMQ Instance and Managing RabbitMQ Instance Tags.
2023-05-09	This issue incorporates the following changes: <ul style="list-style-type: none">Added Getting Started.Added description about new instance flavors and version 3.8.35 in Buying a RabbitMQ Instance and Specifications.Redesigned the instance details page. See "Restarting a RabbitMQ Instance", Deleting a RabbitMQ Instance, and Resetting the RabbitMQ Instance Password.Added Configuring Virtual Hosts, and Advanced Features.
2022-08-12	This issue is the first official release.