



APUSIC
固若长城
睿比世界

Installation Manual

金蝶Apusic分布式消息队列V2.0.6_for_rabbitmq

Copyright Notice

All intellectual property rights including software copyrights involved in this document have been legally registered and duly owned by Kingdee Apusic Cloud Computing Co., Ltd. Such rights are protected by Copyright Law of the People's Republic of China, Regulations on Protection of Computer Software, Regulations on Intellectual Property Protection, relevant international copyright conventions, laws, regulations and other intellectual property legislations. Unauthorized use is strictly prohibited.

Disclaimer

All copyright information contained herein is legally owned by Kingdee Apusic Cloud Computing Co., Ltd. and protected by applicable laws. The Company shall bear no liability for any information unrelated to Kingdee Apusic contained in this document. You may view, copy and print this document only within the scope permitted by Copyright Law of the People's Republic of China. No entity or individual may use, modify or republish any part or content of this document without written authorization from Kingdee Apusic Cloud Computing Co., Ltd. Any violation shall constitute infringement, and the Company reserves the right to pursue legal liabilities accordingly.

Updates to this document shall not be notified separately. You may consult the Company regarding issues related to this document. All rights not expressly granted by the Company are reserved.

Trademark Notice

 is a registered trademark filed with the National Intellectual Property Administration of PRC by Kingdee Apusic Cloud Computing Co., Ltd. Shenzhen. Exclusive rights to the registered trademark are legally owned by Kingdee Apusic and protected by law. Without written consent of Kingdee Apusic, no entity or individual shall use, copy, modify, disseminate, transcribe, or bundle and sell any part of this trademark with other products in any manner or for any reason. Kingdee Apusic shall pursue legal liabilities against any infringement of its trademark rights. All other trademarks or registered trademarks mentioned in this document belong to their respective owners.

Contents

- 1 Preface
 - 1.1 Target Audience
 - 1.2 Related Documentation
 - 1.3 Technical Support
- 2 Deployment Architecture
 - 2.1 Overall Architecture
 - 2.2 Component Description
 - 2.3 Deployment Mode
 - 2.4 Architecture Features
- 3 Environment Preparation
 - 3.1 System Requirements
 - 3.2 Network Requirements
- 4 Standalone Deployment
 - 4.1 Create Deployment Directory
 - 4.2 Download and Extract Package
 - 4.3 Directory Structure Description
 - 4.4 Configuration Changes
 - 4.4.1 Modify Hostname
 - 4.4.2 Modify Listening Port etc. (Optional)
 - 4.5 License
 - 4.6 Start Service
 - 4.7 Service Verification
 - 4.7.1 Check Process Status
 - 4.7.2 Check Port Listening
 - 4.7.3 View Runtime Logs
 - 4.7.4 Create User
 - 4.7.5 Create Virtual Host
- 5 Cluster Deployment
 - 5.1 Deployment Planning
 - 5.2 All Nodes Execute: Basic Installation
 - 5.2.1 Create Directory and Extract
 - 5.2.2 Place License File

- 5.2.3 Modify Hostname
- 5.2.4 Modify Server hosts File
- 5.3 Configure Cluster (Optional)
 - 5.3.1 Node 1 Configuration (172.20.140.171)
 - 5.3.2 Node 2 Configuration (172.20.140.172)
 - 5.3.3 Node 3 Configuration (172.20.140.173)
- 5.4 Start Each Node Service
- 5.5 Create Cluster
- 5.6 Mirror Mode (Optional)
- 5.7 Cluster Verification and Testing
 - 5.7.1 Verify Port Status of Each Node
 - 5.7.2 Check Cluster Status
 - 5.7.3 Functional Test
- 5.8 Cluster Failure Simulation and Maintenance
 - 5.8.1 Stop Service
 - 5.8.2 Common Error Troubleshooting
- 6 Common Issues and Troubleshooting
 - 6.1 Service Startup Failure
 - 6.2 Cluster Join Failure
 - 6.3 Console Inaccessible
 - 6.4 Datasource 403 Error

1 Preface

This document is the installation guide for Kingdee Apusic Distributed Message Queue for RabbitMQ (Apusic Distributed Message Queue, referred to as ADMQ for RabbitMQ). It provides guidance on the installation and deployment process for various deployment modes (including standalone and cluster) across different operating systems and container environments, along with relevant configuration instructions.

1.1 Target Audience

This document is intended for IT business leaders, R&D managers, software project managers, software architects, and operations engineers.

1.2 Related Documentation

To learn more about ADMQ for RabbitMQ, please refer to the following ADMQ for RabbitMQ product documentation set:

No.	Document	Description
1	Kingdee Apusic Distributed Message Queue for RabbitMQ Quick Start Guide	A brief introduction on how to quickly get started with ADMQ for RabbitMQ.
2	Kingdee Apusic Distributed Message Queue for RabbitMQ Installation Guide	Detailed instructions on installing ADMQ for RabbitMQ on various operating systems, as well as operations such as starting and stopping ADMQ for RabbitMQ services.
3	Kingdee Apusic Distributed Message Queue for RabbitMQ Message Engine User Guide	Detailed instructions on using, configuring, and managing ADMQ for RabbitMQ message engine features, along with supporting tools.
4	Kingdee Apusic Distributed Message Queue for RabbitMQ Console User Guide	Detailed instructions on using and operating the ADMQ for RabbitMQ console features.
5	Kingdee Apusic Distributed Message Queue for RabbitMQ Developer Guide	Detailed instructions on developing ADMQ for RabbitMQ client applications in various programming languages.
6	Kingdee Apusic Distributed Message Queue for RabbitMQ Migration Guide	Detailed instructions on migrating from RabbitMQ to ADMQ for RabbitMQ.

7	Kingdee Apusic Distributed Message Queue for RabbitMQ Operations Guide	Detailed instructions on monitoring, operations, and security hardening of ADMQ for RabbitMQ.
8	Kingdee Apusic Distributed Message Queue for RabbitMQ Performance Optimization Guide	Detailed instructions on performance tuning for ADMQ for RabbitMQ.

1.3 Technical Support

ADMQ for RabbitMQ provides comprehensive technical support services. You can obtain technical support through the following channels:

- Website: www.apusic.com
- Phone: 400-855-5800
- Email: support@apusic.com
- Kingdee Cloud Community: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

When contacting technical support, please provide the following information:

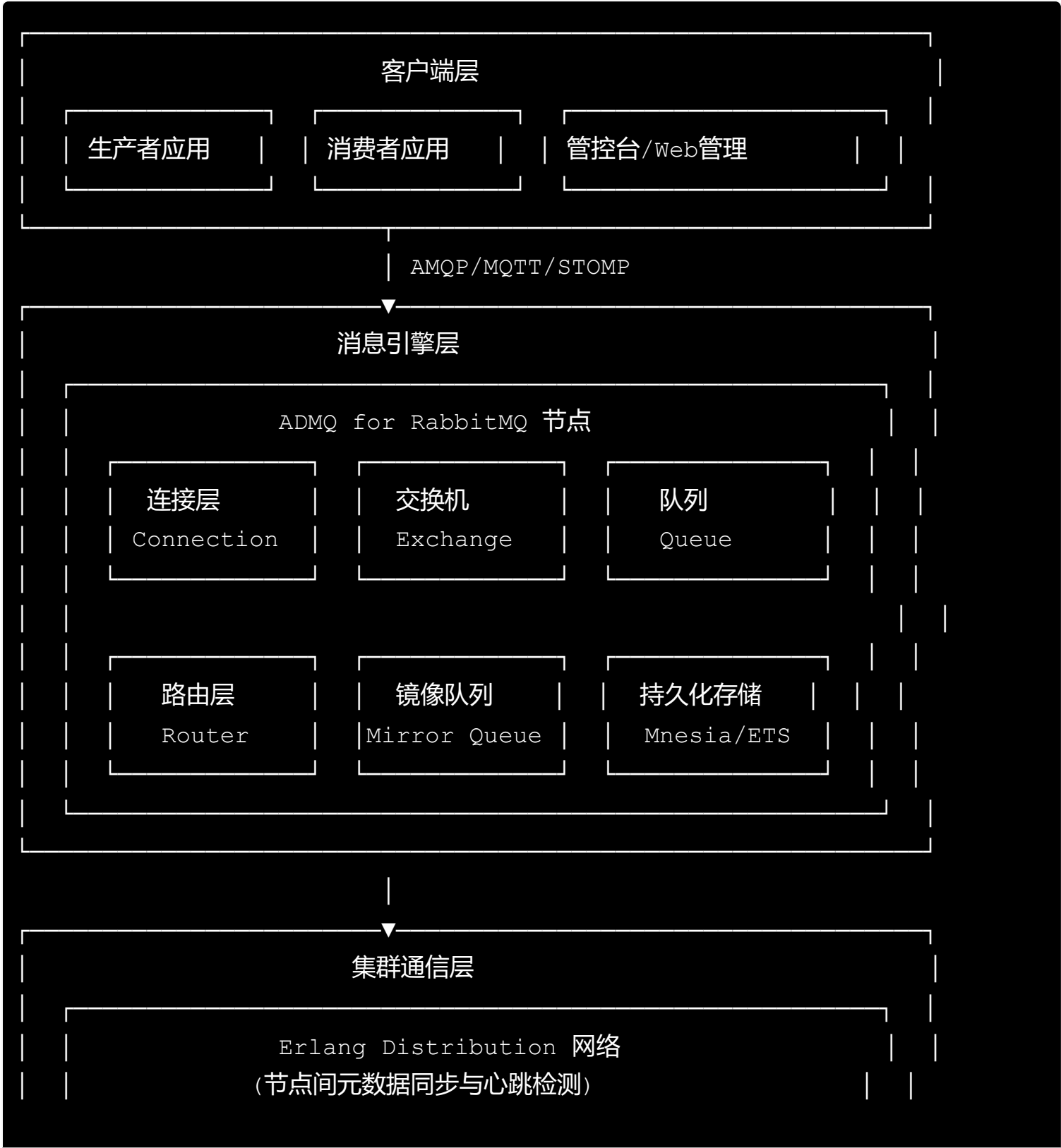
1. Your name
2. Company information and contact details
3. Operating system and version
4. Product version number
5. Detailed information such as error logs, screenshots, etc.

ADMQ for Kafka supports deployment via the console. Please refer to the [Quick Start] section in the "Kingdee Apusic Distributed Message Queue V2.0.6 Unified Console User Guide.pdf".

2 Deployment Architecture

2.1 Overall Architecture

ADMQ for RabbitMQ adopts a modular design, with the overall architecture consisting of the following core components:





2.2 Component Description

Component	Description	Function
Connection Layer	Connection	Handles client connections, protocol parsing, and security authentication
Exchange	Exchange	Receives messages and routes them to queues based on routing rules
Queue	Queue	Stores messages, supports standard queues and Quorum Queues
Mirror Queue	Mirror Queue	Synchronizes queue data across multiple nodes in cluster mode
Persistent Storage	Mnesia/ETS	Stores metadata, configuration information, and persistent messages
Cluster Communication	Erlang Distribution	Inter-node communication, synchronizes metadata and cluster state

2.3 Deployment Mode

ADMQ for RabbitMQ supports the following deployment modes:

Deployment Mode	Nodes	Use Case	High Availability
Standalone	1 node	Development/testing, low throughput scenarios	None
Standard Cluster	2+ nodes	Production environment, horizontal scaling needed	Metadata HA
Mirror Queue Cluster	3+ nodes	Production environment, message data HA needed	Metadata + Data HA

2.4 Architecture Features

- 1. **Distributed Design:** Supports multi-node cluster deployment with automatic metadata synchronization
- 2. **High Availability:** Supports mirror queues to ensure message data is not lost during node failures
- 3. **Elastic Scaling:** Nodes can be dynamically added to expand cluster processing capacity

4. **Multi-Protocol Support:** Natively supports AMQP 0-9-1 protocol
5. **Separation of Control and Data:** Message engine and console can be deployed independently

3 Environment Preparation

3.1 System Requirements

Component	Minimum Requirement	Recommended Configuration
CPU	4 cores	8 cores or more
Memory	8 GB	16 GB or more
Disk	100 GB SSD	500 GB or more SSD
Network	1 Gbps	10 Gbps
OS	Linux (CentOS 7+, Ubuntu 18.04+, Kylin, UOS)	-

3.2 Network Requirements

Port	Purpose	Description
5672	AMQP protocol	Producer and consumer connections
5671	AMQP over SSL	SSL encrypted connection
15672	Management HTTP	HTTP API and Web management interface
15671	Management HTTPS	HTTPS API and Web management interface
15692	Prometheus Metrics	Monitoring metrics collection
25672	Erlang Distribution	Cluster inter-node communication
12305	Console HTTP	Console Web interface (HTTP)
12306	Console HTTPS	Console Web interface (HTTPS)

4 Standalone Deployment

This document guides users through the standalone deployment of ADMQ RabbitMQ in a Linux environment, including environment preparation, installation and configuration, license application, service startup, and status verification.

4.1 Create Deployment Directory

First, create the application root directory and navigate to it:

```
mkdir -p /apusic
cd /apusic
```

4.2 Download and Extract Package

Download the specified version of the installation package from the internal repository and extract it.

```
# Download installation package
# Provided by implementation personnel

# Extract installation package
tar zxvf ADMQ-V2.0.6.31214-RabbitMQ-20260227.tar.gz
```

4.3 Directory Structure Description

After extraction, the `admq-rabbitmq` directory will be created in the current directory. The standard directory structure is as follows:

```
admq-rabbitmq/
├── bin/           # Executable scripts directory
├── config/        # Configuration files directory
├── data/          # Data storage directory
├── logs/          # Log files directory
├── lib/           # Dependency libraries directory
└── license.xml    # License file (needs to be placed manually)
```

4.4 Configuration Changes

4.4.1 Modify Hostname

The hostname may only contain:

- Letters (a-z A-Z)
- Digits (0-9)
- Hyphens (-)

```
# Check hostname
hostname

# Modify hostname
hostname node84
```

4.4.2 Modify Listening Port etc. (Optional)

Edit `config/rabbitmq.conf` to modify listening ports, username, password, etc.

```
# Example: Modify listening port and default user
listeners.tcp.default = 5672
default_user = admq
default_pass = apusic_123
```

4.5 License

ADMQ products require a License file to run properly.

1. **Apply for License:** Contact the administrator or apply for the corresponding version of the `license.xml` file through the authorization platform.
2. **Place the File:** Upload the obtained `license.xml` file to the root path of the deployment directory.

```
# Confirm file location
ls -l /apusic/admq-rabbitmq/license.xml
```

4.6 Start Service

Start the message sending and receiving service:

```
bin/admq-daemon start rabbitmq server
```

```
[root@localhost admq-rabbitmq]# bin/admq-daemon start rabbitmq server
Starting rabbitmq-server, logging to /apusic/admq-rabbitmq/logs
  Base directory: /apusic/admq-rabbitmq/rabbitmq/bin
  Root directory: /apusic/admq-rabbitmq/rabbitmq
  Libs directory: /apusic/admq-rabbitmq/rabbitmq/libs
  Config path: /apusic/admq-rabbitmq/config/rabbitmq.conf
  Starting Apusic RabbitMQ...
[root@localhost admq-rabbitmq]#
```

4.7 Service Verification

4.7.1 Check Process Status

Use the `ps` command to check the process. Under normal conditions, relevant processes should be visible:

```
ps -ef | grep admq-rabbitmq
```

Expected output example:

```
[root@localhost admq-rabbitmq]# ps -ef |grep admq-rabbitmq
root      19084      1  0 14:01 ?        00:00:00 /apusic/admq-rabbitmq/rabbitmq/exec/xdkQuM/erts-14.2.5.11/bin/epmd -daemon
root      19997      1  5 14:09 pts/4      00:00:01 /apusic/admq-rabbitmq/jdk/aarch64/bin/java -Dconfig.path=/apusic/admq-rabbitmq/config/rabbitmq.conf -cp ./apusic/admq-rabbitmq/rabbitmq/libs/acls-client-1.0.9.jar:/apusic/admq-rabbitmq/rabbitmq/libs/admq-monitor-1.0.0-apusic.jar:/apusic/admq-rabbitmq/rabbitmq/libs/apiguardian-api-1.1.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/apusic-license-core-1.7.jar:/apusic/admq-rabbitmq/rabbitmq/libs/apusic-license-lib-1.7.jar:/apusic/admq-rabbitmq/rabbitmq/libs/commons-codec-1.11.jar:/apusic/admq-rabbitmq/rabbitmq/libs/commons-lang3-3.18.0.jar:/apusic/admq-rabbitmq/rabbitmq/libs/commons-logging-1.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/httpclient-4.5.13.jar:/apusic/admq-rabbitmq/rabbitmq/libs/httpcore-4.4.13.jar:/apusic/admq-rabbitmq/rabbitmq/libs/jcommander-1.82.jar:/apusic/admq-rabbitmq/rabbitmq/libs/junit-jupiter-5.10.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/junit-jupiter-api-5.10.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/junit-jupiter-engine-5.10.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/junit-jupiter-params-5.10.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/junit-platform-commons-1.10.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/junit-platform-engine-1.10.2.jar:/apusic/admq-rabbitmq/rabbitmq/libs/log4j-api-2.25.3.jar:/apusic/admq-rabbitmq/rabbitmq/libs/log4j-core-2.25.3.jar:/apusic/admq-rabbitmq/rabbitmq/libs/log4j-slf4j2-impl-2.25.3.jar:/apusic/admq-rabbitmq/rabbitmq/libs/lombok-1.18.28.jar:/apusic/admq-rabbitmq/rabbitmq/libs/opentest4j-1.3.0.jar:/apusic/admq-rabbitmq/rabbitmq/libs/slf4j-api-2.0.12.jar:/apusic/admq-rabbitmq/rabbitmq/libs/starter-3.12.14.jar com.apusic.admq.rabbitmq.Starter
root      20034      0 14:09 pts/4      00:00:00 /bin/sh /apusic/admq-rabbitmq/rabbitmq/core/sbin/rabbitmq-server
root      20042      20 14:09 pts/4      00:00:06 /apusic/admq-rabbitmq/rabbitmq/exec/xdkQuM/erts-14.2.5.11/bin/beam.smp -W w -MBas ageffcbf -MMHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMmcs 30 -pc unicode -P 1048576 -t 5000000 -stbt db -zdbbl 128000 -sbwt none -sbwtdcpu none -sbwtdio none -B i -- -root /apusic/admq-rabbitmq/rabbitmq/exec/xdkQuM -bindir /apusic/admq-rabbitmq/rabbitmq/exec/xdkQuM/erts-14.2.5.11/bin -prognome erl -- -home /apusic/admq-rabbitmq -- -pa -noshell -noinput -s rabbit boot -boot start_sasl -syslog logger [] -syslog syslog_error_logger false -kernel prevent_overlapping_partitions false -enable-feature maybe_expr
root      20355      0 14:09 ?          00:00:00 /apusic/admq-rabbitmq/rabbitmq/exec/xdkQuM/erts-14.2.5.11/bin/inet_gethost 4
root      20356      0 14:09 ?          00:00:00 /apusic/admq-rabbitmq/rabbitmq/exec/xdkQuM/erts-14.2.5.11/bin/inet_gethost 4
root      20378      0 14:09 pts/4      00:00:00 grep --color=auto admq-rabbitmq
[root@localhost admq-rabbitmq]#
```

4.7.2 Check Port Listening

Use `netstat` to verify that port 5672 is in `LISTEN` state:

```
netstat -nltup | grep 5672
```

Expected output example:

```
[root@localhost admq-rabbitmq]# netstat -nltp | grep 5672
tcp        0      0 0.0.0.0:15672        0.0.0.0:*           LISTEN      20042/beam.smp
tcp        0      0 0.0.0.0:25672        0.0.0.0:*           LISTEN      20042/beam.smp
tcp6       0      0 :::5672              :::*                 LISTEN      20042/beam.smp
[root@localhost admq-rabbitmq]#
```

If no output is seen, check the firewall settings or review the logs to troubleshoot startup failure.

4.7.3 View Runtime Logs

If the service starts abnormally or you need to monitor runtime status, check the log files in the `logs` directory:

```
# View real-time logs
tail -f logs/server.log

# Or view directory structure
ls -lh logs/
```

Pay special attention to any `ERROR` level messages in the logs.

```
[root@localhost admq-rabbitmq]# tail -f logs/server.log
2026-03-03 14:09:24.850516+08:00 [info] <0.790.0> Management plugin: HTTP (non-TLS) listener started on port 15672
2026-03-03 14:09:24.850715+08:00 [info] <0.818.0> Statistics database started.
2026-03-03 14:09:24.850807+08:00 [info] <0.817.0> Starting worker pool 'management_worker_pool' with 3 processes in it
2026-03-03 14:09:24.851385+08:00 [info] <0.731.0> Ready to start client connection listeners
2026-03-03 14:09:24.913612+08:00 [info] <0.842.0> started TCP listener on [::]:5672
2026-03-03 14:09:25.016642+08:00 [info] <0.731.0> Server startup complete; 3 plugins started.
2026-03-03 14:09:25.016642+08:00 [info] <0.731.0> * rabbitmq_management
2026-03-03 14:09:25.016642+08:00 [info] <0.731.0> * rabbitmq_management_agent
2026-03-03 14:09:25.016642+08:00 [info] <0.731.0> * rabbitmq_web_dispatch
2026-03-03 14:09:25.207324+08:00 [info] <0.9.0> Time to start RabbitMQ: 6005200 us
```

4.7.4 Create User

Authentication is enabled by default. Clients need to add a username and password to send and receive messages.

Create user: test

Set password: test123

```
bin/admq rabbitmq admin ctl add_user test test123
```

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl add_user test test123
⊗ Executing: rabbitmqctl add_user test test123
Adding user "test" ...
Done. Don't forget to grant the user permissions to some virtual hosts! See 'rabbitmqctl help set_permissions' to learn more.
[root@node84 admq-rabbitmq]#
```

4.7.5 Create Virtual Host

Create virtual host: test

```
bin/admq rabbitmq admin ctl add_vhost test
```

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl add_vhost test
⊗ Executing: rabbitmqctl add_vhost test
Adding vhost "test" ...
```

Set permissions for user: test

```
bin/admq rabbitmq admin ctl set_permissions --vhost test test ".*" ".*" ".*"
bin/admq rabbitmq admin ctl set_user_tags test management
```

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl set_permissions --vhost test test ".*" ".*" ".*"
⊗ Executing: rabbitmqctl set_permissions --vhost test test .* .* .*
Setting permissions for user "test" in vhost "test" ...
[root@node84 admq-rabbitmq]#
```

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl set_user_tags test management
⊗ Executing: rabbitmqctl set_user_tags test management
Setting tags for user "test" to [management] ...
[root@node84 admq-rabbitmq]#
```

5 Cluster Deployment

5.1 Deployment Planning

Assume deploying a 3-node high-availability cluster, planned as follows:

Prerequisites:

1. Ensure time synchronization (NTP) across all three machines.
2. Disable the firewall or open ports: `5672` , `15672` .

5.2 All Nodes Execute: Basic Installation

Execute on Node 1, Node 2, Node 3 respectively:

5.2.1 Create Directory and Extract

```
mkdir -p /apusic
cd /apusic
tar zxvf ADMQ-V2.0.6.31214-RabbitMQ-20260306.tar.gz
cd admq-rabbitmq
```

5.2.2 Place License File

Upload `license.xml` to the `/apusic/admq-rabbitmq/` directory.

5.2.3 Modify Hostname

The hostname may only contain:

- Letters (a-z A-Z)
- Digits (0-9)
- Hyphens (-)

```
# Execute on Node 1
hostname node171

# Execute on Node 2
```

```
hostname node172

# Execute on Node 3
hostname node173
```

5.2.4 Modify Server hosts File

Add information for all cluster nodes to `/etc/hosts` .

Add the following to all nodes:

```
172.20.140.171 node171
172.20.140.172 node172
172.20.140.173 node173
```

5.3 Configure Cluster (Optional)

Edit `config/rabbitmq.conf` to configure listening ports, default username, password, etc.

5.3.1 Node 1 Configuration (`172.20.140.171`)

```
listeners.tcp.default = 5672
# Default user
default_user = admq
default_pass = apusic_123
```

5.3.2 Node 2 Configuration (`172.20.140.172`)

```
listeners.tcp.default = 5672
# Default user
default_user = admq
default_pass = apusic_123
```

5.3.3 Node 3 Configuration (`172.20.140.173`)

```
listeners.tcp.default = 5672
# Default user
default_user = admq
default_pass = apusic_123
```

5.4 Start Each Node Service

Log in to Node 1, 2, and 3 sequentially and execute:

```
cd /apusic/admq-rabbitmq
bin/admq-daemon start rabbitmq server
```

The startup log is as follows:

```
[root@localhost admq-rabbitmq]# bin/admq-daemon start rabbitmq server
Starting rabbitmq-server, logging to /apusic/admq-rabbitmq/logs
  Base directory: /apusic/admq-rabbitmq/rabbitmq/bin
  Root directory: /apusic/admq-rabbitmq/rabbitmq
  Libs directory: /apusic/admq-rabbitmq/rabbitmq/libs
  Config path: /apusic/admq-rabbitmq/config/rabbitmq.conf
  Starting Apusic RabbitMQ...
[root@localhost admq-rabbitmq]#
```

After startup, check the node status to confirm successful startup. You can directly verify whether the port is listening.

```
ss -anpl | grep 5672
```

Expected output:

```
[root@openstack-140-171 admq-rabbitmq]# ss -anp | grep 5672
tcp    LISTEN  0      128      *:25672      *.*          users: (("beam.smp",pid=28206,fd=18))
tcp    LISTEN  0      128      *:15672      *.*          users: (("beam.smp",pid=28206,fd=34))
tcp    LISTEN  0      128      [::]:5672    [::]:*       users: (("beam.smp",pid=28206,fd=35))
```

5.5 Create Cluster

Execute the cluster creation command on the first node to join the other two nodes into the cluster:

```
bin/admq rabbitmq cluster add -s 172.20.140.172:58181 -u monitor -p
apusic_123
bin/admq rabbitmq cluster add -s 172.20.140.173:58181 -u monitor -p
apusic_123
```

```
[root@openstack-140-171 admq-rabbitmq]# bin/admq rabbitmq cluster add -s 172.20.140.172:58181 -u monitor -p apusic_123
🔧 Adding node to cluster...
📌 Target node: 172.20.140.172:58181
👤 Username: monitor
🔑 Step 1: Synchronizing authentication (Cookie)...
☑ Cookie synchronized: Cookie received and saved successfully

🌐 Step 2: Discovering and recording hostnames...
📋 Hosts entry already exists:
☑ Hostnames discovered and recorded

📁 Step 3: Synchronizing reverse hostnames...
☑ Reverse hostnames synchronized: Hostname already exists in hosts file: 10.0.0.131 node171

⚙ Step 4: Initializing cluster...
☑ Cluster initialization completed: Successfully joined cluster with master node: rabbit@node171
☑ Node added to cluster successfully!
[root@openstack-140-171 admq-rabbitmq]#
```

```
[root@openstack-140-171 admq-rabbitmq]# bin/admq rabbitmq cluster add -s 172.20.140.173:58181 -u monitor -p apusic_123
🔧 Adding node to cluster...
📌 Target node: 172.20.140.173:58181
👤 Username: monitor
🔑 Step 1: Synchronizing authentication (Cookie)...
☑ Cookie synchronized: Cookie received and saved successfully

🌐 Step 2: Discovering and recording hostnames...
📋 Hosts entry already exists: 10.0.0.6 node173
☑ Hostnames discovered and recorded

📁 Step 3: Synchronizing reverse hostnames...
{"code":0,"message":"Hostname already exists in hosts file:"}
☑ Reverse hostnames synchronized: Hostname already exists in hosts file:

⚙ Step 4: Initializing cluster...
☑ Cluster initialization completed: Successfully joined cluster with master node: rabbit@node171
☑ Node added to cluster successfully!
[root@openstack-140-171 admq-rabbitmq]#
```

5.6 Mirror Mode (Optional)

In cluster mode, each node can only synchronize metadata, not the data under exchanges and queues. If not using the `Quorum Queues` mode, mirror queues can be used to synchronize data across nodes.

```
bin/admq rabbitmq admin ctl set_policy ha-all "^" '{"ha-
mode":"all","ha-sync-mode":"automatic"}'
```

5.7 Cluster Verification and Testing

5.7.1 Verify Port Status of Each Node

Check port listening status on each server:

```
# Check if port 5672 is listening
ss -anpl | grep 5672
```

Expected: The port should be listening.

```
[root@node84 admq-rabbitmq]# ss -anpl | grep 5672
tcp    LISTEN 0      1024          0.0.0.0:15672      0.0.0.0:*        users:(("beam.smp",pid=175644,fd=37))
tcp    LISTEN 0      128        172.20.140.84:5672 0.0.0.0:*        users:(("beam.smp",pid=175644,fd=38))
tcp    LISTEN 0      128          0.0.0.0:25672     0.0.0.0:*        users:(("beam.smp",pid=175644,fd=18))
```

5.7.2 Check Cluster Status

Execute on Node1:

```
bin/admq rabbitmq cluster status
```

All three nodes should be shown as added to the cluster.

```
[root@openstack-140-171 admq-rabbitmq]# bin/admq rabbitmq cluster status
🔍 Checking RabbitMQ cluster status...
🔧 Executing: rabbitmqctl cluster_status
Cluster status of node rabbit@node171 ...
```

Basics

```
Cluster name: rabbit@node171
Total CPU cores available cluster-wide: 36
```

Disk Nodes

```
rabbit@node171
rabbit@node172
rabbit@node173
```

Running Nodes

```
rabbit@node171
rabbit@node172
rabbit@node173
```

Versions

```
rabbit@node171: RabbitMQ 3.12.14 on Erlang 26.2.5.15
rabbit@node172: RabbitMQ 3.12.14 on Erlang 26.2.5.15
rabbit@node173: RabbitMQ 3.12.14 on Erlang 26.2.5.15
```

CPU Cores

```
Node: rabbit@node171, available CPU cores: 12
Node: rabbit@node172, available CPU cores: 12
Node: rabbit@node173, available CPU cores: 12
```

5.7.3 Functional Test

1. Create User:

Create user: test

Set password: test123

```
bin/admq rabbitmq admin ctl add_user test test123
```

Expected output example:

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl add_user test test123
⊗ Executing: rabbitmqctl add_user test test123
Adding user "test" ...
Done. Don't forget to grant the user permissions to some virtual hosts! See 'rabbitmqctl help set_permissions' to learn more.
[root@node84 admq-rabbitmq]#
```

2. Create Virtual Host:

Create virtual host: test

```
bin/admq rabbitmq admin ctl add_vhost test
```

Expected output example:

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl add_vhost test
⊗ Executing: rabbitmqctl add_vhost test
Adding vhost "test" ...
```

Set permissions for user: test

```
bin/admq rabbitmq admin ctl set_permissions --vhost test test ".*" ".*" ".*"
bin/admq rabbitmq admin ctl set_user_tags test management
```

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl set_permissions --vhost test test ".*" ".*" ".*"
⊗ Executing: rabbitmqctl set_permissions --vhost test test .* .* .*
Setting permissions for user "test" in vhost "test" ...
[root@node84 admq-rabbitmq]#
```

```
[root@node84 admq-rabbitmq]# bin/admq rabbitmq admin ctl set_user_tags test management
⊗ Executing: rabbitmqctl set_user_tags test management
Setting tags for user "test" to [management] ...
[root@node84 admq-rabbitmq]#
```

5.8 Cluster Failure Simulation and Maintenance

5.8.1 Stop Service

```
bin/admq-daemon stop rabbitmq server
```

5.8.2 Common Error Troubleshooting

Cluster node addition failed, node connection failed

```
[root@openstack-140-173 admq-rabbitmq]# tail -f logs/server.log
2026-03-06 16:57:43.728869+08:00 [info] <0.614.0> Stopping application 'mnesia'
2026-03-06 16:57:43.732265+08:00 [notice] <0.44.0> Application mnesia exited with reason: stopped
2026-03-06 16:57:43.732379+08:00 [info] <0.614.0> Successfully stopped RabbitMQ and its dependencies
2026-03-06 16:57:44.214957+08:00 [notice] <0.44.0> Application mnesia exited with reason: stopped
2026-03-06 16:57:44.215292+08:00 [notice] <0.634.0> Feature flags: checking nodes `rabbit@node173` and `rabbit@node171` compatibility...
2026-03-06 16:57:44.285836+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:57:44.409266+08:00 [warning] <0.664.0> 'global' at rabbit@node173 failed to connect to rabbit@node172
2026-03-06 16:57:44.409266+08:00 [warning] <0.664.0>
2026-03-06 16:57:49.292214+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:57:54.297877+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:57:59.317362+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:04.321727+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:09.342998+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:14.347594+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:19.354276+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:24.359365+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:29.366830+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
2026-03-06 16:58:34.371249+08:00 [warning] <0.662.0> Feature flags: no connection to node `rabbit@node172`; retrying in 5000 milliseconds
```

Solution: Configure `/etc/hosts` , add the node172 information, then restart the node.

Cluster node addition failed, node stop failed

```
[root@openstack-140-171 admq-rabbitmq]# bin/admq rabbitmq cluster add -s 172.20.140.173:58181 -u monitor -p apusic_123
✎ Adding node to cluster...
✎ Target node: 172.20.140.173:58181
✎ Username: monitor
✎ Step 1: Synchronizing authentication (Cookie)...
☑ Cookie synchronized: Cookie received and saved successfully

✎ Step 2: Discovering and recording hostnames...
☐ Hosts entry already exists: 10.0.0.6 node173
☑ Hostnames discovered and recorded

☐ Step 3: Synchronizing reverse hostnames...
{"code":0,"message":"Hostname received and added to hosts file: \n10.0.0.6 node173\n\n10.0.0.131 node171\n"}
⚠ Failed to sync reverse hostnames: Hostname received and added to hosts file:
10.0.0.6 node173

10.0.0.131 node171

✎ Step 4: Initializing cluster...
✗ Failed to initialize cluster: Error stopping application
```

Solution: Restart the node to be added.

6 Common Issues and Troubleshooting

6.1 Service Startup Failure

Possible Cause	Solution
Port conflict	Check if ports 5672, 15672, 25672 are occupied
Insufficient permissions	Ensure correct permissions on the admq-rabbitmq directory
Insufficient memory	Check system memory, ensure minimum requirements are met
License not placed or expired	Confirm license.xml is placed in the deployment root directory and has not expired
OpenSSL version incompatibility	Systems like Ubuntu 24.04 use OpenSSL 3.x by default, while the Erlang crypto module depends on libcrypto.so.1.1. Install the OpenSSL 1.1 compatibility library (libssl1.1), then restart the service

OpenSSL 1.1 Compatibility Library Installation Example (Ubuntu/Debian):

```
# Ubuntu 24.04 requires installing libssl1.1 first
wget
http://security.ubuntu.com/ubuntu/pool/main/o/openssl/libssl1.1_1.1.1f-
1ubuntu2.24_amd64.deb
dpkg -i libssl1.1_1.1.1f-1ubuntu2.24_amd64.deb
```

Note: If you see errors like `crypto:strong_rand_bytes` or `libcrypto.so.1.1: cannot open shared object file` in `logs/rabbitmq-server-*.out`, this is an OpenSSL compatibility issue.

6.2 Cluster Join Failure

Possible Cause	Solution
Hostname does not meet specifications	Ensure hostname contains only letters, digits, and hyphens
Network unreachable	Check network connectivity and firewall settings between nodes
Hostname resolution failure	Ensure /etc/hosts or DNS configuration is correct

Node not started properly

Check that each node service has started normally

6.3 Console Inaccessible

Possible Cause	Solution
Port conflict	Modify server.port in config/application.properties
Firewall blocking	Open ports 12305/12306
Service not started	Check console service status

6.4 Datasource 403 Error

Possible Cause	Solution
Insufficient RabbitMQ user permissions	Use a user with the administrator tag and configure .* permissions
Message sending and receiving errors	Check routing key and binding relationships

UNIFIED SERVICE HOTLINE

4008-555-800

Kingdee Apusic Cloud Computing Co., Ltd. (referred to as “Kingdee Apusic”) was founded in 2000. Its predecessor was “Kingdee Middleware Company.” It is a basic software platform provider under Kingdee Group, a national standards-setting enterprise for cloud computing, and a core software enterprise in the national IT application innovation industry. Kingdee Apusic is an enterprise undertaking the National 863 Key R&D Program and the National Major Science and Technology Project for Core Electronic Components, High-end General-purpose Chips and Fundamental Software, and is also a basic platform provider for the national key projects of “Two Networks, One Website, Four Databases, and Twelve Golden Projects.” Its products are widely used in key industries such as government, military, finance, and energy, and it has cumulatively served more than 100,000 customers.



National standard setter for cloud computing
Basic software enterprise under Kingdee Group
Core enterprise for IT application innovation
Website: www.apusic.com

