



## Installation Manual

Kingdee Apusic Distributed Message Queue  
V2.0.6\_for\_rocketmq

## 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 Intended Audience
  - 1.2 Related Documents
  - 1.3 Technical Support
- 2 Deployment Architecture
  - 2.1 ADMQ for RocketMQ Network Deployment Characteristics
- 3 Environment Preparation
  - 3.1 System Requirements
  - 3.2 Hardware Recommendations
  - 3.3 Dependency Software
  - 3.4 Network Requirements
- 4 Standalone Deployment
  - 4.1 Install JDK
  - 4.2 Install ADMQ for RocketMQ
  - 4.3 Start NameServer
  - 4.4 Start Broker
  - 4.5 Verify Installation
- 5 Standalone Deployment (Single Master Deployment)
  - 5.1 NameServer Deployment
  - 5.2 Broker Deployment
- 6 Dual Master Deployment
  - 6.1 Deployment Architecture Diagram
  - 6.2 NameServer Deployment
  - 6.3 Broker Deployment
- 7 Multi-Master Multi-Slave Deployment
  - 7.1 Deployment Architecture Diagram
  - 7.2 NameServer Deployment
  - 7.3 Broker Configuration
- 8 Console Deployment
  - 8.1 Install Console
  - 8.2 Access Console
  - 8.3 Initialize Cluster

## •9 Common Issues and Troubleshooting

- 9.1 NameServer Startup Failure
- 9.2 Broker Startup Failure
- 9.3 Console Inaccessible
- 9.4 Message Send Failure

# 1 Preface

This document is the installation manual for Kingdee Apusic Distributed Message Queue for RocketMQ (Apusic Distributed Message Queue for RocketMQ, abbreviation: ADMQ for RocketMQ). It provides guidance on the installation and deployment process for various deployment modes (including standalone, dual-master, multi-master multi-slave) across different operating systems and container environments, along with related configuration instructions.

## 1.1 Intended Audience

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

## 1.2 Related Documents

For more information about ADMQ for RocketMQ products, please refer to the following ADMQ for RocketMQ product documentation set:

| No. | Document                                                                         | Description                                                                                                                                      |
|-----|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | Kingdee Apusic Distributed Message Queue for RocketMQ Quick Start Manual         | A brief introduction on how to quickly get started with ADMQ for RocketMQ.                                                                       |
| 2   | Kingdee Apusic Distributed Message Queue for RocketMQ Installation Manual        | Detailed instructions on installing ADMQ for RocketMQ on various operating systems, as well as starting and stopping ADMQ for RocketMQ services. |
| 3   | Kingdee Apusic Distributed Message Queue for RocketMQ Message Engine User Manual | Detailed description of the usage, configuration, management, and supporting tools of the ADMQ for RocketMQ message engine.                      |
| 4   | Kingdee Apusic Distributed Message Queue for RocketMQ Console User Manual        | Detailed description of the usage and operation instructions for the ADMQ for RocketMQ console.                                                  |
| 5   | Kingdee Apusic Distributed Message Queue for RocketMQ Development Manual         | Detailed instructions for developing ADMQ for RocketMQ client applications in various programming languages.                                     |
| 6   | Kingdee Apusic Distributed Message Queue for RocketMQ Migration Manual           | Detailed instructions on migrating from RocketMQ to ADMQ for RocketMQ.                                                                           |

|   |                                                                                       |                                                                                                                   |
|---|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| 7 | Kingdee Apusic Distributed Message Queue for RocketMQ Operations Manual               | Detailed description of ADMQ for RocketMQ monitoring, operations, security hardening, and other O&M instructions. |
| 8 | Kingdee Apusic Distributed Message Queue for RocketMQ Performance Optimization Manual | Detailed description of ADMQ for RocketMQ performance tuning.                                                     |

## 1.3 Technical Support

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

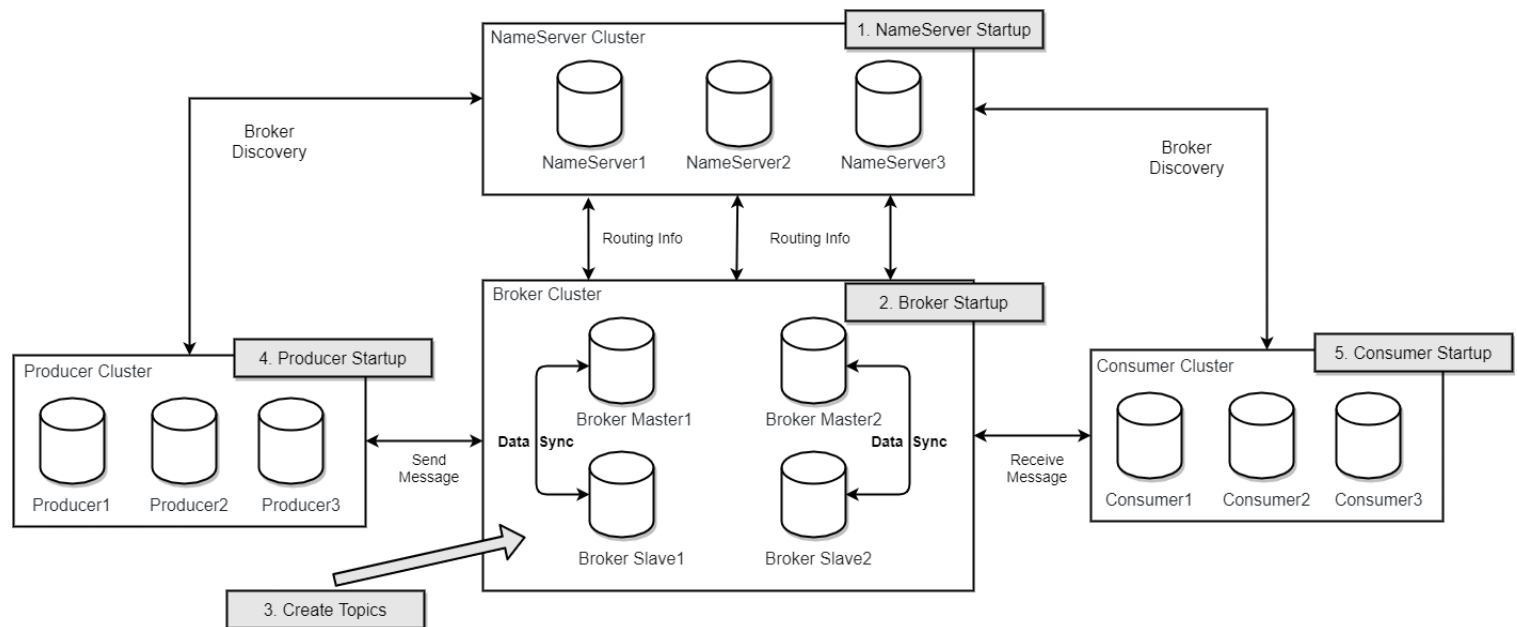
- Website: [www.apusic.com](http://www.apusic.com)
- Phone: 400-855-5800
- Email: [support@apusic.com](mailto:support@apusic.com)
- Kingdee Cloud Community: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

When contacting technical support, please provide the following information:

1. Your name
2. Company information and contact details
3. Operating system and its version
4. Product version number
5. Detailed information such as logs and screenshots of anomalies and errors

*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 Manual.pdf".*

## 2 Deployment Architecture



### 2.1 ADMQ for RocketMQ Network Deployment Characteristics

- NameServer is an almost stateless node that can be deployed in a cluster, with no information synchronization between nodes.
- Broker deployment is relatively complex. Brokers are divided into Master and Slave. One Master can correspond to multiple Slaves, but one Slave can only correspond to one Master. The Master-Slave relationship is defined by specifying the same BrokerName with different BrokerId. A BrokerId of 0 indicates a Master, and non-zero indicates a Slave. Multiple Masters can also be deployed. Each Broker establishes persistent connections with all nodes in the NameServer cluster and periodically registers Topic information to all NameServers. Note: The current RocketMQ version supports one Master with multiple Slaves in its deployment architecture, but only the slave server with BrokerId=1 participates in message read load balancing.
- Producer establishes persistent connections with one of the nodes (randomly selected) in the NameServer cluster, periodically obtains Topic routing information from NameServer, establishes persistent connections with the Master providing the Topic service, and periodically sends heartbeats to the Master. Producer is completely stateless and can be deployed in clusters.
- Consumer establishes persistent connections with one of the nodes (randomly selected) in the NameServer cluster, periodically obtains Topic routing information from NameServer, establishes persistent connections with the Master and Slave providing the Topic service, and periodically sends heartbeats to the Master and Slave. Consumers can subscribe to messages from both Master and Slave. When a

consumer pulls messages from the Master, the Master server determines whether the next pull should be from the Master or Slave based on factors such as the distance between the pull offset and the maximum offset (determining whether to read old messages, resulting in read I/O), and whether the slave server is readable.

Based on the deployment architecture diagram, the cluster workflow is described as follows:

- Start NameServer. After NameServer is started, it listens on a port and waits for Broker, Producer, and Consumer to connect, acting as a routing control center.
- Broker starts, maintains persistent connections with all NameServers, and periodically sends heartbeat packets. The heartbeat packets contain current Broker information (IP+port, etc.) and all stored Topic information. After successful registration, the mapping relationship between Topics and Brokers exists in the NameServer cluster.
- Before sending and receiving messages, first create a Topic. When creating a Topic, you need to specify which Brokers the Topic should be stored on. You can also automatically create a Topic when sending messages.
- Producer sends messages. When starting, it first establishes a persistent connection with one of the NameServer cluster nodes, obtains from NameServer which Brokers the current Topic to send exists on, polls to select a queue from the queue list, then establishes a persistent connection with the Broker where the queue is located to send messages to the Broker.
- Consumer, similar to Producer, establishes a persistent connection with one NameServer, obtains which Brokers the currently subscribed Topic exists on, then directly establishes a connection channel with the Broker to start consuming messages.

## 3 Environment Preparation

### 3.1 System Requirements

Before starting deployment, ensure your environment meets the following requirements:

- **Operating System:** 64-bit Linux (recommended CentOS 7+, Ubuntu 18.04+, domestic Kylin v10+, OpenEuler 22.03+)
- **Java Version:** JDK 8 and above
- **Hardware Configuration:**
  - Minimum production environment requires 7 servers (or virtual machines)
    - 3 for running Name Server
    - 4 for running Broker (2 Master and 2 Slave)
  - Test environment can run all components on a single server
- **Network:** All nodes can communicate with each other with a stable network connection
- **DNS:** Optional, configure a common DNS name for all Broker nodes

### 3.2 Hardware Recommendations

The following table provides hardware recommendations for different cluster sizes (per node):

| Component   | Small Cluster (250 Topics)                      | Medium Cluster (1000 Topics)                    |
|-------------|-------------------------------------------------|-------------------------------------------------|
| Name Server | • CPU: 4 cores • Memory: 16GB • Disk: 200GB SSD | • CPU: 8 cores • Memory: 32GB • Disk: 500GB SSD |
| Broker      | • CPU: 8 cores • Memory: 32GB • Disk: 500GB SSD | • CPU: 16 cores • Memory: 64GB • Disk: 1TB SSD  |

Hardware Selection Recommendations

- Name Server is only used for metadata storage and can use relatively smaller servers
- Broker computation requires fast CPU and network; a 10Gbps network interface is recommended for networking; storage requires fast storage devices; SSDs or RAID controllers with battery-backed cache are recommended

### 3.3 Dependency Software

| Software | Version Requirement | Description                              |
|----------|---------------------|------------------------------------------|
| JDK      | 1.8+                | Broker and NameServer runtime dependency |

|      |     |                            |
|------|-----|----------------------------|
| Java | 17+ | Console runtime dependency |
|------|-----|----------------------------|

## 3.4 Network Requirements

| Port  | Purpose       | Description                        |
|-------|---------------|------------------------------------|
| 9876  | NameServer    | Service registration and discovery |
| 10911 | Broker        | Producer and consumer connections  |
| 10909 | Broker VIP    | VIP channel                        |
| 10912 | Broker HA     | Master-Slave replication port      |
| 12305 | Console HTTP  | Console web interface (HTTP)       |
| 12306 | Console HTTPS | Console web interface (HTTPS)      |

## 4 Standalone Deployment

### 4.1 Install JDK

```
# CentOS/RHEL
sudo yum install -y java-1.8.0-openjdk

# Ubuntu/Debian
sudo apt-get update
sudo apt-get install -y openjdk-8-jdk
```

### 4.2 Install ADMQ for RocketMQ

```
mkdir -p /apusic
cd /apusic
# Extract installation package
#tar zxvf ADMQ-V2.0.6-RocketMQ-<build-date>.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260625.tar.gz -C /apusic/
cd /apusic
# It is recommended to separate the nameserver and broker component
folder structures
unzip rocketmq-V5.3.4-all.zip
mv rocketmq-V5.3.4-all nameserver
cd /apusic/nameserver

unzip rocketmq-V5.3.4-all.zip
mv rocketmq-V5.3.4-all broker
cd /apusic/broker

# Configure environment variables, change to the corresponding
machine address
export NAMESRV_ADDR=localhost:9876
```

```
# Configure authorization certificate, place the corresponding
authorization certificate in the root directory of the respective
component. Assume the license.xml file (file name must be unified as
license.xml) is placed under /opt
cp /apusic/license.xml /apusic/nameserver
cp /apusic/license.xml /apusic/broker
```

## 4.3 Start NameServer

```
# Start NameServer
bin/rocketmq start nameserver

# Verify status
tail -f logs/namesrv.log
```

## 4.4 Start Broker

```
# Configure Broker, some parameters are default, can be modified as
needed, such as cluster brokerId, set according to the number of
nodes
vi conf/broker.conf
```

```
# broker.conf
brokerClusterName=DefaultCluster
# (The brokerName of cluster nodes needs to be distinguished)
brokerName=broker-a
brokerId=0
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=localhost:9876
# Enable Prometheus
enableMultiDispatch=true
```

```
enablemq=true  
metricsPromExporterHost=0.0.0.0  
metricsExporterType=PROM
```

```
# Start Broker  
bin/rocketmq start broker  
  
# Verify status  
tail -f logs/broker.log
```

## 4.5 Verify Installation

```
# View cluster status  
bin/mqadmin clusterList -n localhost:9876  
  
# View Topic list  
bin/mqadmin topicList -n localhost:9876
```

## 5 Standalone Deployment (Single Master Deployment)

This approach carries high risk. Once the Broker restarts or goes down, the entire service becomes unavailable. It is not recommended for production environments and can be used for local testing.

```
+-----+
| NameServer1 |
+-----+
      |
+-----+
| Broker-a    |
| Master      |
+-----+
```

### 5.1 NameServer Deployment

Start NameServer on node1:

```
# node1
nohup bin/rocketmq start nameserver
```

### 5.2 Broker Deployment

Broker-a (node1)

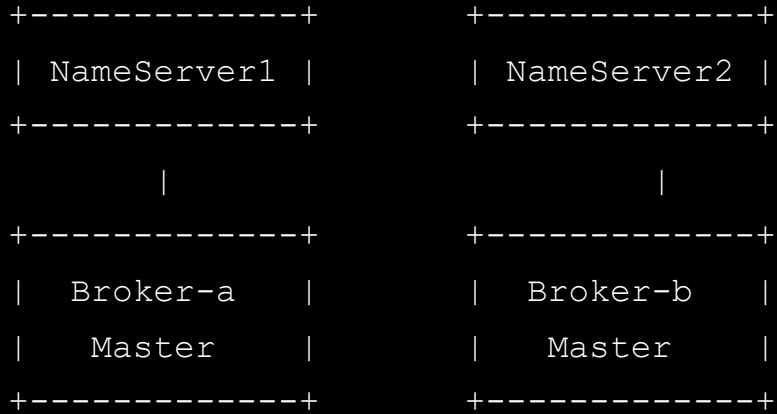
```
# conf/broker-a.properties
brokerClusterName=AdmqCluster
brokerName=broker-a
brokerId=0
brokerRole=SYNC_MASTER
flushDiskType=ASYNC_FLUSH
namesrvAddr=node1:9876
# Enable Prometheus
enableMultiDispatch=true
enablemq=true
```

```
metricsPromExporterHost=0.0.0.0  
metricsExporterType=PROM
```

```
# node1  
bin/rocketmq start broker
```

## 6 Dual Master Deployment

### 6.1 Deployment Architecture Diagram



### 6.2 NameServer Deployment

Start NameServer on node1 and node2 respectively:

```
# node1
bin/rocketmq start nameserver

# node2
bin/rocketmq start nameserver
```

### 6.3 Broker Deployment

Broker-a (node1)

```
# conf/broker-a.properties
brokerClusterName=AdmqCluster
# (The brokerName of cluster nodes needs to be distinguished)
brokerName=broker-a
brokerId=0
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
```

```
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=node1:9876;node2:9876
# Enable Prometheus
enableMultiDispatch=true
enablemq=true
metricsPromExporterHost=0.0.0.0
metricsExporterType=PROM
```

#### Broker-b (node2)

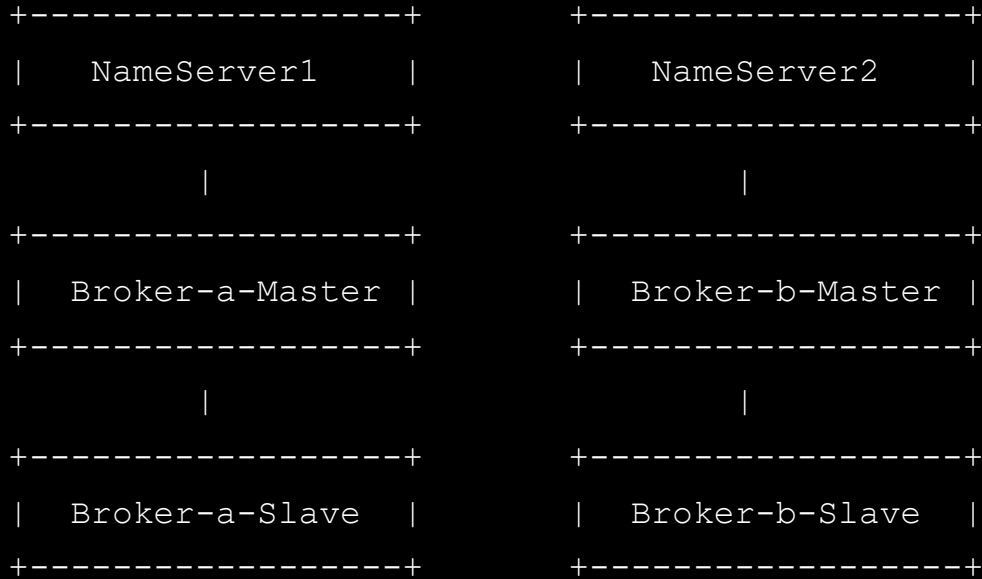
```
# conf/broker-b.properties
brokerClusterName=AdmqCluster
# (The brokerName of cluster nodes needs to be distinguished)
brokerName=broker-b
brokerId=0
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=node1:9876;node2:9876
# Enable Prometheus
enableMultiDispatch=true
enablemq=true
metricsPromExporterHost=0.0.0.0
metricsExporterType=PROM
```

```
# node1
bin/rocketmq start broker

# node2
bin/rocketmq start broker
```

## 7 Multi-Master Multi-Slave Deployment

### 7.1 Deployment Architecture Diagram



### 7.2 NameServer Deployment

First start the corresponding nameserver component

```
./bin/rocketmq start nameserver
```

### 7.3 Broker Configuration

Broker-a-Master (node1)

```
brokerClusterName=AdmqCluster
# (The brokerName of the cluster master node needs to be
distinguished)
brokerName=broker-a
brokerId=0
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=node1:9876;node2:9876
```

```
# Enable Prometheus
enableMultiDispatch=true
enablemq=true
metricsPromExporterHost=0.0.0.0
metricsExporterType=PROM
```

#### Broker-a-Slave (node3)

```
brokerClusterName=AdmqCluster
brokerName=broker-a
# (Slave node id, since master and slave have the same name, they
are in the same group)
brokerId=1
brokerRole=SLAVE
flushDiskType=SYNC_FLUSH
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=node1:9876;node2:9876
# Enable Prometheus
enableMultiDispatch=true
enablemq=true
metricsPromExporterHost=0.0.0.0
metricsExporterType=PROM
```

#### Broker-b-Master (node2)

```
brokerClusterName=AdmqCluster
# (The brokerName of the cluster master node needs to be
distinguished)
brokerName=broker-b
brokerId=0
brokerRole=SYNC_MASTER
flushDiskType=SYNC_FLUSH
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=node1:9876;node2:9876
# Enable Prometheus
```

```
enableMultiDispatch=true
enablemq=true
metricsPromExporterHost=0.0.0.0
metricsExporterType=PROM
```

#### Broker-b-Slave (node4)

```
brokerClusterName=AdmqCluster
brokerName=broker-b
# (Slave node id, since master and slave have the same name, they
are in the same group)
brokerId=1
brokerRole=SLAVE
flushDiskType=SYNC_FLUSH
# (For cluster deployment, fill in the port ip of all cluster nodes)
namesrvAddr=node1:9876;node2:9876
# Enable Prometheus
enableMultiDispatch=true
enablemq=true
metricsPromExporterHost=0.0.0.0
metricsExporterType=PROM
```

#### Start via command

```
./bin/rocketmq start broker
```

## 8 Console Deployment

### 8.1 Install Console

```
# Extract console installation package
tar zxvf admq-manager-V2.0.6.tar.gz -C /apusic/
cd /apusic/admq-manager-V2.0.6

# Configure NameServer address
vi config/application.properties
```

```
# application.properties
server.port=12306
```

```
# Start console
bin/admq-service start

# Verify status
bin/admq-service status
```

### 8.2 Access Console

```
Address: http://<Console IP>:12305(https://<Console IP>:12306)
Default account: admq / 11111111
```

### 8.3 Initialize Cluster

You can quickly deploy a cluster directly through the console; for already deployed clusters, you can manage them by connecting through the console

## 9 Common Issues and Troubleshooting

### 9.1 NameServer Startup Failure

| Possible Cause           | Solution                                                   |
|--------------------------|------------------------------------------------------------|
| Port conflict            | Check if port 9876 is occupied                             |
| JDK not installed        | Install JDK 1.8+                                           |
| Insufficient permissions | Ensure the admq-rocketmq directory has correct permissions |

### 9.2 Broker Startup Failure

| Possible Cause          | Solution                                                 |
|-------------------------|----------------------------------------------------------|
| NameServer unreachable  | Check namesrvAddr configuration and network connectivity |
| Port conflict           | Check if port 10911 is occupied                          |
| Insufficient disk space | Check disk space to ensure minimum requirements are met  |
| Configuration error     | Check if broker.conf configuration is correct            |

### 9.3 Console Inaccessible

| Possible Cause         | Solution                                              |
|------------------------|-------------------------------------------------------|
| NameServer unreachable | Check network connectivity from console to NameServer |
| Port conflict          | Modify server.port in config/application.properties   |
| Firewall blocking      | Open ports 12305/12306                                |

### 9.4 Message Send Failure

| Possible Cause           | Solution                                     |
|--------------------------|----------------------------------------------|
| Topic not created        | Use mqadmin to create Topic                  |
| Broker unreachable       | Check Broker status and network connectivity |
| Insufficient permissions | Check ACL configuration                      |

Disk full

Clean up disk space or adjust message retention policy

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](http://www.apusic.com)

