



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_kafka

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 Overview
- 3 Deployment Architecture
 - 3.1 Standalone Deployment Architecture
 - 3.2 Cluster Deployment Architecture
 - 3.3 Hybrid Deployment Architecture
- 4 Pre-installation Preparation
 - 4.1 System Requirements
 - 4.2 Port Check
 - 4.3 Disable Firewall (Test Environment)
- 5 Standalone Deployment
 - 5.1 1. Create Deployment Directory
 - 5.2 2. Download and Extract the Package
 - 5.3 3. Directory Structure
 - 5.4 4. Configuration
 - 5.5 5. License Authorization
 - 5.6 6. Start Services
 - 5.6.1 6.1 Start ADCC for zk
 - 5.6.2 6.2 Start Broker
 - 5.7 7. Service Verification
 - 5.7.1 7.1 Check Process Status
 - 5.7.2 7.2 Check Port Listening
 - 5.7.3 7.3 Functional Test
 - 5.8 8. Stop Services
- 6 Cluster Deployment
 - 6.1 1. Deployment Planning
 - 6.2 2. Execute on All Nodes: Basic Installation
 - 6.3 3. Place License File
 - 6.4 4. Configure ADCC for zk Cluster
 - 6.4.1 4.1 Modify ADCC for zk Configuration File
 - 6.4.2 4.2 Create MyID File
 - 6.5 5. Configure Broker Cluster

- 6.5.1 5.1 Node 1 Configuration (192.168.1.10)
- 6.5.2 5.2 Node 2 Configuration (192.168.1.11)
- 6.5.3 5.3 Node 3 Configuration (192.168.1.12)
- 6.6 6. Start the Cluster
 - 6.6.1 6.1 Step 1: Start ADCC for zk on All Nodes
 - 6.6.2 6.2 Step 2: Start Broker on All Nodes
- 6.7 7. Verification and Testing
 - 6.7.1 7.1 Verify ADCC for zk Cluster Status
 - 6.7.2 7.2 Verify Broker Cluster Metadata
 - 6.7.3 7.3 Functional Test
- 6.8 8. Stop Services
- 7 Service Deployment (systemd)
 - 7.1 Register System Services
 - 7.2 Service Management Commands
- 8 Common Issues
 - 8.1 Installation Issues
 - 8.2 Startup Issues
- 9 Rollback Plan
 - 9.1 Standalone Rollback
 - 9.2 Cluster Rollback
 - 9.3 Configuration Rollback
- 10 Security Hardening
 - 10.1 1. Access Control
 - 10.1.1 1.1 Firewall Configuration
 - 10.1.2 1.2 Disable Unnecessary Protocols
 - 10.2 2. Authentication Configuration
 - 10.2.1 2.1 SASL/SCRAM Authentication
 - 10.2.2 2.2 Create Secure Users
 - 10.3 3. Authorization Configuration
 - 10.3.1 3.1 Configure ACL
 - 10.4 4. SSL/TLS Encryption
 - 10.4.1 4.1 Generate Certificates
 - 10.4.2 4.2 Configure SSL
 - 10.5 5. Audit Logging
 - 10.6 6. OS Security
 - 10.6.1 6.1 File Permissions
 - 10.6.2 6.2 Disable Root Execution

• 11 Technical Support

1 Preface

This document is the installation manual for Apusic Distributed Message Queue for Kafka (ADMQ for Kafka). It provides guidance on deploying the product in various modes (including standalone and cluster) across different operating systems and container environments, along with related configuration instructions.

1.1 Intended Audience

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

1.2 Related Documents

For more information about ADMQ for Kafka, please refer to the following ADMQ for Kafka documentation set:

No.	Document	Description
1	Apusic Distributed Message Queue for Kafka Quick Start Guide	A brief introduction on how to quickly get started with ADMQ for Kafka.
2	Apusic Distributed Message Queue for Kafka Installation Guide	Detailed instructions on installing ADMQ for Kafka on various operating systems, including starting and stopping services.
3	Apusic Distributed Message Queue for Kafka Message Engine User Guide	Detailed instructions on using, configuring, and managing ADMQ for Kafka message engine features and related tools.
4	Apusic Distributed Message Queue for Kafka Console User Guide	Detailed instructions on using the ADMQ for Kafka management console.
5	Apusic Distributed Message Queue for Kafka Developer Guide	Detailed instructions on developing ADMQ for Kafka client applications using various programming languages.
6	Apusic Distributed Message Queue for Kafka Migration Guide	Detailed instructions on migrating from Kafka to ADMQ for Kafka.
7	Apusic Distributed Message Queue for Kafka Operations Guide	Detailed instructions on monitoring, operations, and security hardening for ADMQ for Kafka.
8	Apusic Distributed Message Queue for Kafka Performance Optimization Guide	Detailed instructions on performance tuning for ADMQ for Kafka.

1.3 Technical Support

ADMQ for Kafka provides comprehensive technical support services. You can reach us 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 seeking 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 and screenshots

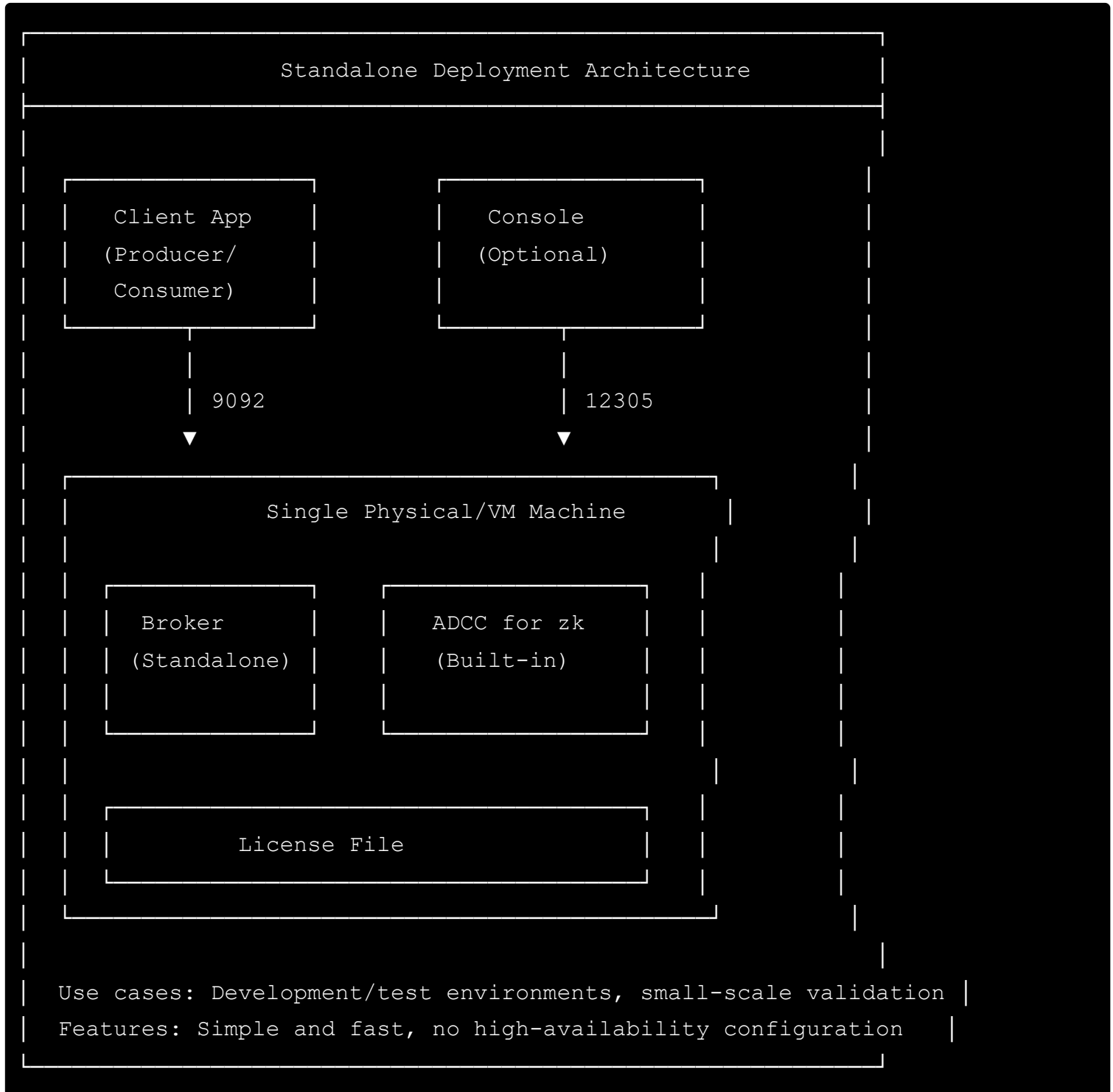
ADMQ for Kafka supports deployment via the management console. Please refer to the [Quick Start] section of the "Apusic Distributed Message Queue V2.0.6_Unified Management Console User Guide.pdf".

2 Overview

ADMQ for Kafka consists of three components: the core engine, distributed coordination, and the management console. The core engine is controlled by a license, which must be applied for and imported during installation. The management console is an optional component; it can be installed separately if visual management is required.

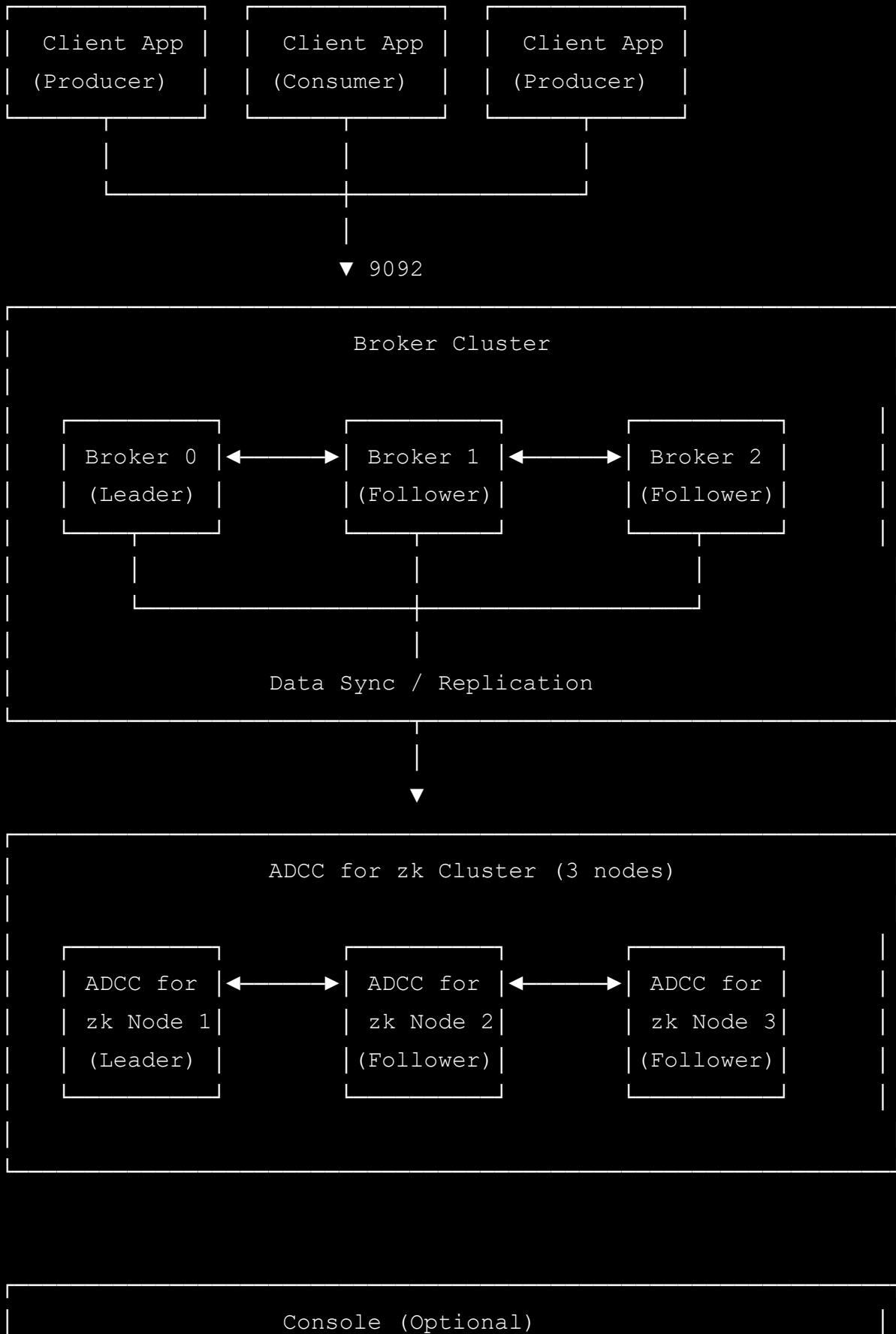
3 Deployment Architecture

3.1 Standalone Deployment Architecture



3.2 Cluster Deployment Architecture

Cluster Deployment Architecture



Port: 12305

Use cases: Production environments, high-availability business scenarios

Features: Multi-replica redundancy, automatic failover, horizontal scaling

3.3 Hybrid Deployment Architecture

Hybrid Deployment Architecture

Client App
(Producer)

Client App
(Consumer)

Client App
(Producer)

▼ 9092

Broker Cluster

Node 1

Broker 0

ADCC for

zk Node 1

Node 2

Broker 1

ADCC for

zk Node 2

Node 3

Broker 2

ADCC for

zk Node 3

```
| Use cases: Resource-constrained production environments, small to medium clusters |
| Features: Saves server resources, ZK and Broker share nodes, lower deployment cost |
| Note: Ensure sufficient node resources (CPU/memory/disk) to avoid resource contention |
```

4 Pre-installation Preparation

4.1 System Requirements

Component	Minimum Requirement	Recommended Configuration
Operating System	Linux (CentOS 7+, Ubuntu 18.04+, Kylin V10, UOS)	-
CPU	4 cores	8 cores or more
Memory	8 GB	16 GB or more
Disk Space	100 GB	500 GB or more SSD
JDK	JDK 1.8+ (Built-in OpenJDK 11)	JDK 11/17

4.2 Port Check

Ensure the following ports are available:

Port	Service	Description
9092	Broker	Producer and consumer connections
2181	ADCC for zk	ADCC for zk client connections
2888	ADCC for zk	Leader and Follower communication
3888	ADCC for zk	Election communication
12305	Console	Web interface (HTTP)

4.3 Disable Firewall (Test Environment)

```
# CentOS/RHEL
systemctl stop firewalld
systemctl disable firewalld

# Ubuntu
ufw disable
```

5 Standalone Deployment

5.1 1. Create Deployment Directory

Create the application root directory and navigate into it:

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

5.2 2. Download and Extract the Package

Obtain the installation package from the product CD or contact sales:

```
# Assuming the package has been copied to the server
tar zxvf ADMQ-V2.0.6.391-Kafka-20260319.tar.gz
```

5.3 3. Directory Structure

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

Directory	Description
<code>bin</code>	Contains startup, shutdown, and management scripts
<code>config</code>	Contains configuration files (e.g., <code>kafka-standalone.conf</code>)
<code>data</code>	Stores message data and ADCC for zk data
<code>jdk</code>	Built-in Java runtime environment, no additional system JDK configuration required
<code>logs</code>	Stores runtime logs (Broker logs, ADCC for zk logs, etc.)
<code>service</code>	System service registration related files (optional)

5.4 4. Configuration

Edit the standalone mode configuration file to bind the specified IP address and port:

```
vi config/kafka-standalone.conf
```

Modify the `listeners` configuration item:

```
listeners=PLAINTEXT://192.168.1.10:9092
```

5.5 5. License Authorization

ADMQ for Kafka requires 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 directory of the deployment path

```
# Verify file location
ls -l /apusic/admq-kafka/license.xml
```

5.6 6. Start Services

ADMQ for Kafka standalone mode depends on ADCC for zk and must be started in order.

5.6.1 6.1 Start ADCC for zk

```
bin/admq-daemon start kafka zk
```

5.6.2 6.2 Start Broker

After ADCC for zk has started successfully (typically wait 3-5 seconds), start the Kafka main service:

```
bin/admq-daemon start kafka standalone
```

5.7 7. Service Verification

5.7.1 7.1 Check Process Status

```
ps -ef | grep admq-kafka
```

Under normal conditions, two main Java processes should be visible (one ADCC for zk, one Broker).

5.7.2 7.2 Check Port Listening

```
netstat -nltup | grep 9092
```

Confirm that port 9092 is in the LISTEN state.

5.7.3 7.3 Functional Test

```
# Create a test topic
kafka/bin/kafka-topics.sh --create \
  --bootstrap-server 192.168.1.10:9092 \
  --topic test-topic \
  --partitions 3 \
  --replication-factor 1

# Produce messages
kafka/bin/kafka-console-producer.sh \
  --bootstrap-server 192.168.1.10:9092 \
  --topic test-topic

# Consume messages
kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server 192.168.1.10:9092 \
  --topic test-topic \
  --from-beginning
```

5.8 8. Stop Services

Operation	Command
Stop ADCC for zk	bin/admq-daemon stop kafka zk
Stop Broker	bin/admq-daemon stop kafka standalone

6 Cluster Deployment

6.1 1. Deployment Planning

Assume a 3-node high-availability cluster (ADCC for zk and Broker hybrid deployment):

Node	IP Address	ADCC for zk MyID	Broker ID	ADCC for zk Ports	Broker Port
Node 1	192.168.1.10	1	0	2181,2888,3888	9092
Node 2	192.168.1.11	2	1	2181,2888,3888	9092
Node 3	192.168.1.12	3	2	2181,2888,3888	9092

Prerequisites:

1. Ensure time synchronization (NTP) across all three machines
2. Disable firewall or open ports: 2181, 2888, 3888, 9092

6.2 2. Execute on All Nodes: Basic Installation

Execute the following on Node 1, Node 2, and Node 3 respectively:

```
mkdir -p /apusic
cd /apusic
tar zxvf ADMQ-V2.0.6.391-Kafka-20260319.tar.gz
cd admq-kafka
```

6.3 3. Place License File

Upload `license.xml` to the `/apusic/admq-kafka/` directory (all nodes).

6.4 4. Configure ADCC for zk Cluster

6.4.1 4.1 Modify ADCC for zk Configuration File

Edit `config/kafka-zk.conf`. The configuration content is identical on all three nodes:

```
server.1=192.168.1.10:2888:3888:participant;2181
server.2=192.168.1.11:2888:3888:participant;2181
server.3=192.168.1.12:2888:3888:participant;2181
```

6.4.2 4.2 Create MyID File

Node 1 (192.168.1.10):

```
mkdir -p /apusic/admq-kafka/data/zk/data
mkdir -p /apusic/admq-kafka/data/zk/logs
echo "1" > /apusic/admq-kafka/data/zk/data/myid
```

Node 2 (192.168.1.11):

```
mkdir -p /apusic/admq-kafka/data/zk/data
mkdir -p /apusic/admq-kafka/data/zk/logs
echo "2" > /apusic/admq-kafka/data/zk/data/myid
```

Node 3 (192.168.1.12):

```
mkdir -p /apusic/admq-kafka/data/zk/data
mkdir -p /apusic/admq-kafka/data/zk/logs
echo "3" > /apusic/admq-kafka/data/zk/data/myid
```

6.5 5. Configure Broker Cluster

Edit `config/kafka-broker.conf` .

6.5.1 5.1 Node 1 Configuration (192.168.1.10)

```
broker.id=0
listeners=PLAINTEXT://192.168.1.10:9092
zookeeper.connect=192.168.1.10:2181,192.168.1.11:2181,192.168.1.12:2181
zookeeper.connection.timeout.ms=18000
```

6.5.2 5.2 Node 2 Configuration (192.168.1.11)

```
broker.id=1
listeners=PLAINTEXT://192.168.1.11:9092
zookeeper.connect=192.168.1.10:2181,192.168.1.11:2181,192.168.1.12:2181
```

6.5.3 5.3 Node 3 Configuration (192.168.1.12)

```
broker.id=2
listeners=PLAINTEXT://192.168.1.12:9092
zookeeper.connect=192.168.1.10:2181,192.168.1.11:2181,192.168.1.12:2181
```

6.6 6. Start the Cluster

Strictly follow the startup order: ADCC for zk cluster first, then Broker cluster.

6.6.1 6.1 Step 1: Start ADCC for zk on All Nodes

Login to Node 1, 2, and 3 sequentially and execute:

```
cd /apusic/admq-kafka
bin/admq-daemon start kafka zk
```

6.6.2 6.2 Step 2: Start Broker on All Nodes

Login to Node 1, 2, and 3 sequentially and execute:

```
cd /apusic/admq-kafka
bin/admq-daemon start kafka broker
```

6.7 7. Verification and Testing

6.7.1 7.1 Verify ADCC for zk Cluster Status

```
echo ruok | nc 192.168.1.10 2181
echo ruok | nc 192.168.1.11 2181
echo ruok | nc 192.168.1.12 2181
```

Expected response: `imok` .

6.7.2 7.2 Verify Broker Cluster Metadata

```
kafka/bin/kafka-broker-api-versions.sh --bootstrap-server 192.168.1.10:9092
```

6.7.3 7.3 Functional Test

1. Create Topic (3 replicas):

```
kafka/bin/kafka-topics.sh --create \
  --topic test-cluster \
  --bootstrap-server 192.168.1.10:9092 \
  --partitions 3 \
  --replication-factor 3
```

2. View Topic Details:

```
kafka/bin/kafka-topics.sh --describe \
  --topic test-cluster \
  --bootstrap-server 192.168.1.10:9092
```

3. Produce and Consume Test:

```
# Producer
kafka/bin/kafka-console-producer.sh \
  --topic test-cluster \
  --bootstrap-server 192.168.1.10:9092

# Consumer
kafka/bin/kafka-console-consumer.sh \
  --topic test-cluster \
  --from-beginning \
  --bootstrap-server 192.168.1.10:9092
```

6.8 8. Stop Services

```
# Stop Kafka first
bin/admq-daemon stop kafka broker
# Then stop ADCC for zk
bin/admq-daemon stop kafka zk
```

7 Service Deployment (systemd)

7.1 Register System Services

```
# Copy service files
cp service/admq-kafka.service /etc/systemd/system/
cp service/admq-zookeeper.service /etc/systemd/system/

# Reload daemon
systemctl daemon-reload

# Start services
systemctl start admq-zookeeper
systemctl start admq-kafka

# View status
systemctl status admq-kafka
```

7.2 Service Management Commands

Operation	Command
Start ADCC for zk	<code>systemctl start admq-zookeeper</code>
Start Broker	<code>systemctl start admq-kafka</code>
Stop	<code>systemctl stop admq-kafka</code>
Restart	<code>systemctl restart admq-kafka</code>
Status	<code>systemctl status admq-kafka</code>
Enable on boot	<code>systemctl enable admq-kafka</code>

8 Common Issues

8.1 Installation Issues

Issue	Cause	Solution
Port in use	Port 9092 or 2181 is already in use	Check the port and release it or modify the configuration port
Insufficient memory	Server memory is below the minimum requirement	Increase server memory
Insufficient disk space	Disk space is full	Clean up disk or add storage
Invalid License	License file is incorrect or expired	Apply for a valid license and replace it

8.2 Startup Issues

Issue	Cause	Solution
ADCC for zk fails to start	Incorrect myid file configuration	Check the myid file content
Broker cannot connect to ADCC for zk	Incorrect ADCC for zk address configuration	Check the zookeeper.connect configuration
Broker ID conflict	Duplicate broker.id	Check and modify broker.id

9 Rollback Plan

9.1 Standalone Rollback

To roll back to the pre-installation state:

```
# 1. Stop services
bin/admq-daemon stop kafka standalone
bin/admq-daemon stop kafka zk

# 2. Back up and delete data directory (use with caution)
mv /apusic/admq-kafka /apusic/admq-kafka.bak.$(date +%Y%m%d)

# 3. To restore, re-extract the installation package and configure
```

9.2 Cluster Rollback

```
# 1. Stop all node services in order
bin/admq-daemon stop kafka broker
bin/admq-daemon stop kafka zk

# 2. Back up data on each node
mv /apusic/admq-kafka /apusic/admq-kafka.bak.$(date +%Y%m%d)

# 3. Clean up ADCC for zk data
cd /apusic/admq-kafka.bak.*/data/zk
cp -r data /backup/zk-data-$(date +%Y%m%d)

# 4. Re-deploy and restore configuration
```

9.3 Configuration Rollback

```
# Back up current configuration
cp config/kafka-broker.conf config/kafka-broker.conf.bak
cp config/kafka-zk.conf config/kafka-zk.conf.bak

# Restore configuration
```

```
cp config/kafka-broker.conf.bak config/kafka-broker.conf
cp config/kafka-zk.conf.bak config/kafka-zk.conf

# Restart services to apply changes
bin/admq-daemon restart kafka broker
```

10 Security Hardening

10.1 1. Access Control

10.1.1 1.1 Firewall Configuration

```
# Only open necessary ports
iptables -A INPUT -p tcp --dport 9092 -s 192.168.1.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 2181 -s 192.168.1.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 12305 -s 192.168.1.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 9092 -j DROP
iptables -A INPUT -p tcp --dport 2181 -j DROP
```

10.1.2 1.2 Disable Unnecessary Protocols

```
# Only enable secure protocols
listeners=SASL_SSL://0.0.0.0:9092
security.inter.broker.protocol=SASL_SSL
```

10.2 2. Authentication Configuration

10.2.1 2.1 SASL/SCRAM Authentication

```
# Enable SCRAM authentication
sasl.enabled.mechanisms=SCRAM-SHA-256,SCRAM-SHA-512
sasl.mechanism.inter.broker.protocol=SCRAM-SHA-512
```

10.2.2 2.2 Create Secure Users

```
# Create admin user
kafka-configs.sh --zookeeper localhost:2181 \
  --alter --add-config 'SCRAM-SHA-256=[password=admin-password]' \
  --entity-type users --entity-name admin

# Create application user
kafka-configs.sh --zookeeper localhost:2181 \
```

```
--alter --add-config 'SCRAM-SHA-256=[password=app-password]' \
--entity-type users --entity-name app-user
```

10.3 3. Authorization Configuration

10.3.1 3.1 Configure ACL

```
# Grant producer permissions
kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 \
--add --allow-principal User:producer \
--operation Write --topic my-topic

# Grant consumer permissions
kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 \
--add --allow-principal User:consumer \
--operation Read --topic my-topic \
--group my-consumer-group

# Deny unauthorized access
kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 \
--add --deny-principal User:Anonymous \
--operation All --topic '*'
```

10.4 4. SSL/TLS Encryption

10.4.1 4.1 Generate Certificates

```
# Generate CA certificate
keytool -keystore kafka.server.keystore.jks \
--alias ca-cert -import -file ca-cert.crt

# Generate Broker certificate
keytool -keystore kafka.server.keystore.jks \
--alias broker-cert -validity 365 -genkey -keyalg RSA

# Import trust certificate
keytool -keystore kafka.server.truststore.jks \
--alias ca-cert -import -file ca-cert.crt
```

10.4.2 4.2 Configure SSL

```
# SSL configuration
listeners=SSL://0.0.0.0:9093
ssl.keystore.location=/path/to/kafka.server.keystore.jks
ssl.keystore.password=keystore-password
ssl.key.password=key-password
ssl.truststore.location=/path/to/kafka.server.truststore.jks
ssl.truststore.password=truststore-password
ssl.client.auth=required
ssl.enabled.protocols=TLSv1.2,TLSv1.3
```

10.5 5. Audit Logging

```
# Enable audit logging
log4j.logger.kafka.authorizer.logger=WARN, authorizerAppender
log4j.appender.authorizerAppender=org.apache.log4j.DailyRollingFileAppender
log4j.appender.authorizerAppender.File=${log.dir}/kafka-authorizer.log
log4j.appender.authorizerAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.authorizerAppender.layout.ConversionPattern=[%d] %p %m (%c)%n
```

10.6 6. OS Security

10.6.1 6.1 File Permissions

```
# Set directory permissions
chmod 750 /apusic/admq-kafka
chmod 600 /apusic/admq-kafka/config/*.conf
chmod 600 /apusic/admq-kafka/license.xml

# Restrict log access
chmod 640 /apusic/admq-kafka/logs/*.log
```

10.6.2 6.2 Disable Root Execution

```
# Create dedicated user
useradd -r -s /bin/false admq-kafka
```

```
chown -R admq-kafka:admq-kafka /apusic/admq-kafka
```

```
# Restrict permissions using systemd  
# Add the following to admq-kafka.service:  
User=admq-kafka  
Group=admq-kafka  
NoNewPrivileges=true  
ProtectSystem=strict  
ProtectHome=true
```

11 Technical Support

If you encounter issues that cannot be resolved, please contact technical support:

- Phone: 400-855-5800
- Email: support@apusic.com

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

