



Quick Operation Guide

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 Overview
- 2 Product Overview
 - 2.1 What is ADMQ for Kafka
 - 2.2 Core Features
 - 2.3 Applicable Scenarios
 - 2.4 Product Components
- 3 Deployment Architecture
 - 3.1 Standalone Deployment
 - 3.2 Cluster Deployment (Recommended for Production)
- 4 Quick Installation
 - 4.1 System Requirements
 - 4.2 Port Planning
 - 4.3 Standalone Installation (5 Minutes)
 - 4.4 Cluster Installation (3 Nodes)
- 5 Quick Command Reference
 - 5.1 Process Management
 - 5.2 Topic Management
 - 5.3 Message Testing
 - 5.4 Consumer Group Management
 - 5.5 Cluster Management
- 6 Routine Operations
 - 6.1 Inspection Checklist
 - 6.2 Core Configuration Files
 - 6.3 Fault Handling
- 7 Performance Tuning
 - 7.1 System-Level Optimization
 - 7.2 Broker High-Throughput Configuration
 - 7.3 Producer Optimization
 - 7.4 Consumer Optimization
 - 7.5 JVM Tuning
- 8 Monitoring Metrics
 - 8.1 Key Alert Metrics
 - 8.2 Monitoring Tools
- 9 Directory Structure
- 10 Technical Support

1 Overview

This document is a concise operations guide for ADMQ for Kafka, covering deployment architecture, quick installation, common commands, routine operations, and performance tuning essentials.

2 Product Overview

2.1 What is ADMQ for Kafka

ADMQ for Kafka (Apusic Distributed Message Queue for Kafka) is a cloud-native message middleware designed as a domestic alternative for Kafka users. Built on Apache Kafka 3.9.1 kernel, it fully conforms to Kafka's core architecture design and communication protocol specifications, achieving full compatibility with protocols, clients, and ecosystem tools.

2.2 Core Features

Feature	Description
Full Protocol Compatibility	Fully implements Kafka produce/consume protocol, partition assignment protocol, and offset management protocol, behaving identically to native Kafka
Zero-Modification Replacement	Supports official Kafka clients in Java, Go, Python, C++, etc., requiring no code changes for replacement
Domestic OS Adaptation	Compatible with Kylin, UOS and other domestic operating systems, supporting ARM, RISC-V and other domestic chips
High-Availability Architecture	Supports multi-replica redundancy, automatic failover, and horizontal scaling, meeting production reliability requirements
Ecosystem Reuse	Compatible with native ecosystem tools such as Kafka Connect, MirrorMaker, Prometheus, etc.

2.3 Applicable Scenarios

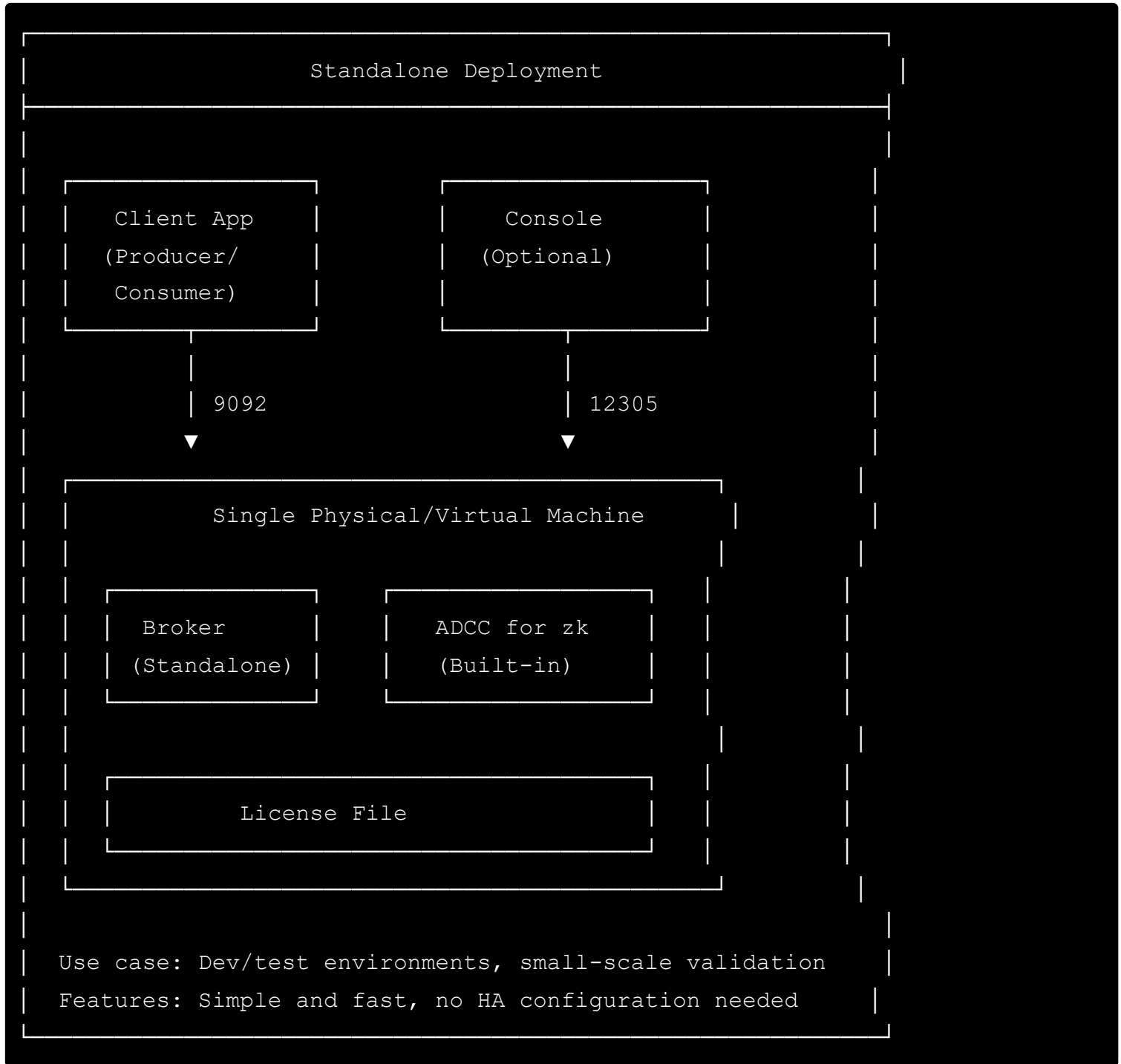
- **Domestic Replacement:** Direct replacement of existing Kafka clusters in government/enterprise digital transformation
- **High-Throughput Data Transfer:** Log collection, business data synchronization, real-time data pipelines
- **Streaming Data Processing:** Compatible with Flink, Spark Streaming and other stream processing engines
- **System Decoupling:** Asynchronous communication between distributed systems based on publish-subscribe model

2.4 Product Components

Component	Description	Required
Broker (Core Engine)	Message processing and storage core, controlled by License	Yes
ADCC for zk (Distributed Coordination)	Cluster metadata management and coordination	Yes
Console	Visual operations management interface	No (Optional)

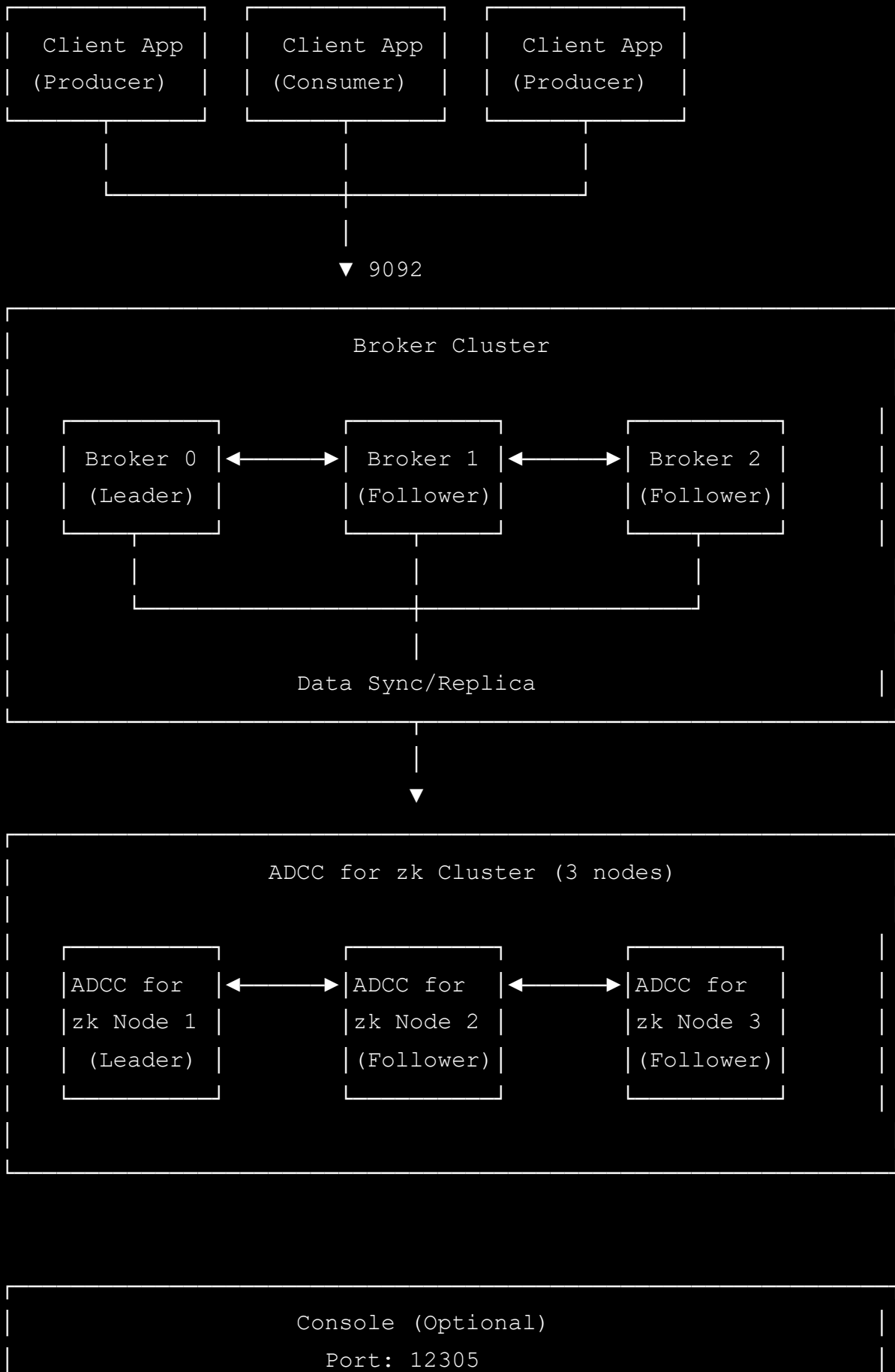
3 Deployment Architecture

3.1 Standalone Deployment



3.2 Cluster Deployment (Recommended for Production)





Use case: Production environment, HA business scenarios

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

4 Quick Installation

4.1 System Requirements

Component	Minimum Requirement	Recommended Configuration
OS	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

4.2 Port Planning

Port	Service	Description
9092	Broker	Producer and consumer connections
2181	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 Standalone Installation (5 Minutes)

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

# 2. Configure (modify to local IP)
vi config/kafka-standalone.conf
# listeners=PLAINTEXT://192.168.1.10:9092

# 3. Place License
cp /path/to/license.xml ./

# 4. Start (ZK first, then Broker)
bin/admq-daemon start kafka zk
```

```
sleep 3
bin/admq-daemon start kafka standalone

# 5. Verify
netstat -nltup | grep 9092
kafka/bin/kafka-topics.sh --list --bootstrap-server 192.168.1.10:9092
```

4.4 Cluster Installation (3 Nodes)

Node Planning:

Node	IP	ADCC for zk MyID	Broker ID
Node 1	192.168.1.10	1	0
Node 2	192.168.1.11	2	1
Node 3	192.168.1.12	3	2

Execute on each node:

```
# 1. Extract and configure
mkdir -p /apusic && cd /apusic
tar zxvf ADMQ-V2.0.6.391-Kafka-20260319.tar.gz
cd admq-kafka

# 2. Configure ADCC for zk (same on all nodes)
cat > config/kafka-zk.conf << 'EOF'
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
EOF

# 3. Create myid (different on each node)
mkdir -p data/zk/data
echo "1" > data/zk/data/myid # Node 1 write 1, Node 2 write 2, Node 3 write 3

# 4. Configure Broker (modify broker.id and listeners per node)
cat > config/kafka-broker.conf << 'EOF'
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
EOF
```

Start the Cluster:

```
# Start ADCC for zk on all nodes
bin/admq-daemon start kafka zk

# Start Broker on all nodes
bin/admq-daemon start kafka broker

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

5 Quick Command Reference

5.1 Process Management

```
# Start/Stop ADCC for zk
bin/admq-daemon start kafka zk
bin/admq-daemon stop kafka zk

# Start/Stop Broker (Standalone)
bin/admq-daemon start kafka standalone
bin/admq-daemon stop kafka standalone

# Start/Stop Broker (Cluster)
bin/admq-daemon start kafka broker
bin/admq-daemon stop kafka broker
```

5.2 Topic Management

```
# Create Topic
kafka/bin/kafka-topics.sh --create \
  --bootstrap-server localhost:9092 \
  --topic my-topic --partitions 6 --replication-factor 3

# List Topics
kafka/bin/kafka-topics.sh --list --bootstrap-server localhost:9092

# Describe Topic (including ISR)
kafka/bin/kafka-topics.sh --describe \
  --bootstrap-server localhost:9092 --topic my-topic

# Delete Topic
kafka/bin/kafka-topics.sh --delete \
  --bootstrap-server localhost:9092 --topic my-topic

# Increase Partitions (can only increase)
kafka/bin/kafka-topics.sh --alter \
```

```
--bootstrap-server localhost:9092 \  
--topic my-topic --partitions 12
```

5.3 Message Testing

```
# Produce messages  
kafka/bin/kafka-console-producer.sh \  
  --bootstrap-server localhost:9092 --topic my-topic  
  
# Consume messages (from beginning)  
kafka/bin/kafka-console-consumer.sh \  
  --bootstrap-server localhost:9092 \  
  --topic my-topic --from-beginning  
  
# Consume messages (latest)  
kafka/bin/kafka-console-consumer.sh \  
  --bootstrap-server localhost:9092 --topic my-topic
```

5.4 Consumer Group Management

```
# List consumer groups  
kafka/bin/kafka-consumer-groups.sh \  
  --bootstrap-server localhost:9092 --list  
  
# View consumption progress (key: monitor LAG)  
kafka/bin/kafka-consumer-groups.sh \  
  --bootstrap-server localhost:9092 \  
  --group my-group --describe  
  
# Reset Offset to earliest  
kafka/bin/kafka-consumer-groups.sh \  
  --bootstrap-server localhost:9092 \  
  --group my-group --topic my-topic \  
  --reset-offsets --to-earliest --execute  
  
# Reset Offset to latest  
kafka/bin/kafka-consumer-groups.sh \  
  --bootstrap-server localhost:9092
```

```
--bootstrap-server localhost:9092 \  
--group my-group --topic my-topic \  
--reset-offsets --to-latest --execute
```

5.5 Cluster Management

```
# View Broker information  
kafka/bin/kafka-broker-api-versions.sh --bootstrap-server localhost:9092  
  
# Preferred replica election (when Leader is unbalanced)  
kafka/bin/kafka-leader-election.sh \  
  --bootstrap-server localhost:9092 \  
  --election-type preferred --all-topic-partitions  
  
# Replica reassignment (use after scaling out)  
kafka/bin/kafka-reassign-partitions.sh \  
  --bootstrap-server localhost:9092 --generate
```

6 Routine Operations

6.1 Inspection Checklist

Check Item	Command	Normal Status
Broker process	<code>ps -ef grep admq-kafka</code>	Process exists
Port listening	<code>netstat -nltup grep 9092</code>	LISTEN state
Disk usage	<code>df -h</code>	< 70%
I/O usage	<code>iostat -x</code>	< 80%
ISR status	<code>kafka-topics.sh --describe</code>	ISR = Replicas
Consumer lag	<code>kafka-consumer-groups.sh --describe</code>	LAG not continuously growing

6.2 Core Configuration Files

Broker Key Configuration (`config/kafka-broker.conf`):

```
# Identity and network
broker.id=0
listeners=PLAINTEXT://192.168.1.10:9092
advertised.listeners=PLAINTEXT://192.168.1.10:9092

# Storage
log.dirs=/data/kafka
num.partitions=6

# Reliability (required for production)
default.replication.factor=3
min.insync.replicas=2
unclean.leader.election.enable=false

# Performance
num.network.threads=16
num.io.threads=32
compression.type=zstd

# Log retention
log.retention.hours=168
```

```
log.segment.bytes=1073741824

# ADCC for zk
zookeeper.connect=192.168.1.10:2181,192.168.1.11:2181,192.168.1.12:2181
```

ADCC for zk Key Configuration (`config/kafka-zk.conf`):

```
tickTime=2000
initLimit=10
syncLimit=5

dataDir=/apusic/admq-kafka/data/zk/data
dataLogDir=/apusic/admq-kafka/data/zk/logs

# Cluster 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

# Auto cleanup
autopurge.snapRetainCount=3
autopurge.purgeInterval=6
```

6.3 Fault Handling

Issue	Diagnostic Command	Solution
Broker down	<code>ps -ef grep admq-kafka</code>	<code>bin/admq-daemon start kafka broker</code>
Leader unavailable	<code>kafka-topics.sh --describe</code>	Run preferred replica election
ISR out of sync	<code>kafka-topics.sh --describe</code>	Check disk I/O, network latency
Consumer lag (LAG)	<code>kafka-consumer-groups.sh --describe</code>	Increase Consumer/Partition
Disk full	<code>df -h</code>	Expand disk or reduce log retention time
ADCC for zk connection failure	Check <code>myid</code> , ports, firewall	Fix configuration and restart

7 Performance Tuning

7.1 System-Level Optimization

```
# File descriptors
ulimit -n 1000000

# Network parameters
sysctl -w net.core.rmem_max=67108864
sysctl -w net.core.wmem_max=67108864
sysctl -w net.ipv4.tcp_rmem="4096 87380 67108864"
sysctl -w net.ipv4.tcp_wmem="4096 16384 67108864"

# Disable swap
sysctl -w vm.swappiness=0

# File system (mount options)
mount -o noatime,nodiratime /dev/sda1 /data/kafka
```

7.2 Broker High-Throughput Configuration

```
# Network and threads
num.network.threads=16
num.io.threads=32
socket.send.buffer.bytes=307200
socket.receive.buffer.bytes=307200
queued.max.requests=2000

# Log
log.segment.bytes=524288000
log.retention.check.interval.ms=300000

# Compression and replicas
compression.type=zstd
replica.socket.timeout.ms=30000
replica.fetch.max.bytes=1048576
```

7.3 Producer Optimization

```
Properties props = new Properties();
props.put("bootstrap.servers", "192.168.1.10:9092");
props.put("acks", "all");
props.put("compression.type", "zstd");
props.put("batch.size", 32768);
props.put("linger.ms", 10);
props.put("buffer.memory", 67108864);
props.put("max.in.flight.requests.per.connection", 5);
```

7.4 Consumer Optimization

```
Properties props = new Properties();
props.put("bootstrap.servers", "192.168.1.10:9092");
props.put("group.id", "my-group");
props.put("max.poll.records", 1000);
props.put("fetch.min.bytes", 1);
props.put("fetch.max.wait.ms", 500);
```

7.5 JVM Tuning

```
# Large heap + G1 GC
KAFKA_HEAP_OPTS="-Xms16g -Xmx16g -XX:+UseG1GC \
  -XX:MaxGCPauseMillis=20 \
  -XX:InitiatingHeapOccupancyPercent=35 \
  -XX:G1HeapRegionSize=16m"

# JMX monitoring
JMX_PORT=9999
```

8 Monitoring Metrics

8.1 Key Alert Metrics

Metric	Alert Threshold	Description
UnderReplicatedPartitions	> 0	Under-replicated partitions
OfflinePartitionsCount	> 0	Offline partitions
RequestLatencyAvg	> 100ms	Average request latency
MessagesInPerSec	Per business	Messages per second
LAG	Continuously growing	Consumer lag

8.2 Monitoring Tools

```
# JMX connection
jconsole localhost:9999

# Prometheus configuration
scrape_configs:
  - job_name: 'kafka'
    static_configs:
      - targets: ['localhost:12305']
```

9 Directory Structure

```
admq-kafka/
├── bin/                # Start/stop scripts
├── config/             # Configuration files
│   ├── kafka-standalone.conf    # Standalone configuration
│   ├── kafka-broker.conf       # Broker configuration
│   └── kafka-zk.conf           # ADCC for zk configuration
├── data/               # Data directories
│   ├── kafka/           # Message data
│   └── zk/              # ADCC for zk data
├── jdk/                # Built-in JDK
├── logs/               # Runtime logs
└── service/            # systemd service files
```

10 Technical Support

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

