Apusic 金蝶天燕

生成时间：2026-05-07

用户手册

金蝶Apusic分布式消息队列V2.0.6_for_kafka

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 版本更新说明

2 产品简介

- 2.1 产品定位
- 2.2 核心价值
- 2.3 核心能力
- 2.4 产品架构
- 2.5 产品优势
- 2.6 应用场景

3 产品安装

- 3.1 安装概述
- 3.2 单机部署
  - 3.2.1 1. 环境准备与软件安装
    - 3.2.1.1 1.1 创建部署目录
    - 3.2.1.2 1.2 下载并解压软件包
    - 3.2.1.3 1.3 目录结构说明
  - 3.2.2 2. 配置修改
    - 3.2.2.1 2.1 编辑监听地址
  - 3.2.3 3. 授权许可
  - 3.2.4 4. 启动服务
    - 3.2.4.1 4.1 启动 ZooKeeper
    - 3.2.4.2 4.2 启动 Kafka Broker
  - 3.2.5 5. 服务验证
    - 3.2.5.1 5.1 检查进程状态
    - 3.2.5.2 5.2 检查端口监听
    - 3.2.5.3 5.3 查看运行日志
  - 3.2.6 6.停止服务
- 3.3 集群部署
  - 3.3.1 1. 部署规划
  - 3.3.2 2. 所有节点执行：基础安装
    - 3.3.2.1 2.1 创建目录并解压
    - 3.3.2.2 2.2 放置授权文件
  - 3.3.3 3. 配置 ZooKeeper 集群

- 3.3.3.1 3.1 修改 ZK 配置文件
- 3.3.3.2 3.2 创建 MyID 文件 (每台机器不同)
- 3.3.4 4. 配置 Kafka Broker 集群
- 3.3.4.1 4.1 Node 1 配置 (192.168.1.10)
- 3.3.4.2 4.2 Node 2 配置 (192.168.1.11)
- 3.3.4.3 4.3 Node 3 配置 (192.168.1.12)
- 3.3.5 5. 启动集群
- 3.3.5.1 5.1 第一步：启动所有节点的 ZooKeeper
- 3.3.5.2 5.2 第二步：启动所有节点的 Kafka Broker
- 3.3.6 6. 验证与测试
- 3.3.6.1 6.1 验证 ZooKeeper 集群状态
- 3.3.6.2 6.2 验证 Kafka 集群元数据
- 3.3.6.3 6.3 功能测试 (创建 Topic 并生产消费)
- 3.3.7 7. 停止服务
- 3.4 管理控制台部署
- 3.4.1 安装前准备
- 3.4.2 开始安装
- 3.4.2.1 获取上传产品包
- 3.4.2.2 管理控制台安装
- 3.4.2.3 启动管理控制台服务
- 3.4.2.4 停止管理控制台服务
- 3.4.3 验证

4 管控台使用

- 4.1 概述
- 4.2 系统管理
- 4.2.1 导入licnese
- 4.2.2 上传产品包
- 4.2.3 对接集群
- 4.2.4 开通运维用户
- 4.3 运维管理
- 4.3.1 使用运维用户登录
- 4.3.2 概览与切换
- 4.3.3 主题管理
- 4.3.3.1 主题创建

- 4.3.3.2 主题维护
- 4.3.3.3 主题详情
- 4.3.4 订阅组管理
- 4.3.4.1 订阅组与主题
- 4.3.4.2 主题消费者
- 4.3.5 消息查询

5 配置与维护

- 5.1 核心引擎目录说明
- 5.2 Broker 配置说明
- 5.2.1 基础配置 (Basic Configuration)
- 5.2.2 副本与可靠性 (Replication & Reliability)
- 5.2.3 日志保留策略 (Log Retention)
- 5.2.4 性能调优 (Performance Tuning)
- 5.2.5 ZooKeeper 连接 (ZooKeeper Connection)
- 5.2.6 安全 (Security)
- 5.2.7 其他 (Others)
- 5.3 ZK 配置说明
- 5.3.1 基础配置 (Basic Timing)
- 5.3.2 数据存储 (Data Storage)
- 5.3.3 连接信息 (Connection Info)
- 5.3.4 集群配置 (Cluster Config)
- 5.3.5 自动清理 (Auto Purge)
- 5.3.6 TCP 网络优化 (Network Tuning)
- 5.3.7 监控和管理 (Admin & 4lw)
- 5.3.8 其他配置 (Others)
- 5.4 常见命令
- 5.4.1 Topic 管理
- 5.4.1.1 创建 Topic
- 5.4.1.2 查看 Topic 列表
- 5.4.1.3 查看 Topic 详情 (分区、副本分布)
- 5.4.1.4 增加分区数 (只能增加，不能减少)
- 5.4.1.5 删除 Topic
- 5.4.2 生产与消费 (数据测试)
- 5.4.2.1 启动控制台生产者 (Producer)

- 5.4.2.2 启动控制台消费者 (Consumer)
- 5.4.3 消费者组管理 (Consumer Groups)
- 5.4.3.1 查看消费者组列表
- 5.4.3.2 查看组内详情 (延迟、Offset)
- 5.4.3.3 重置 Offset (慎用)
- 5.4.4 集群与元数据管理
- 5.4.4.1 查看集群 Broker 信息
- 5.4.4.2 首选副本选举 (Preferred Leader Election)
- 5.4.4.3 查看集群配置

6 常见问题

- 6.0.1 常见错误排查

1 版本更新说明

本文档最新版本包含历史修改记录如下：

更新日期	手册版本	适用产品	更新说明
2026年02月	V1.0	金蝶Apusic分布式消息队列V2.0.5 for kafka	初版发布
2026年03月	V1.1	金蝶Apusic分布式消息队列V2.0.6 for kafka	增加了2.0.6版本的管控台说明

2 产品简介

2.1 产品定位

金蝶天燕分布式消息中间件（ADMQ for kafka)是专为Kafka用户打造的信创替代型云原生消息中间件，完全遵循 Kafka 核心架构设计与特性规范，实现协议、客户端、生态工具、业务场景的全兼容，无需改造即可替换原有 Kafka 集群，满足信创项目落地需求。

2.2 核心价值

遵循 Kafka 分布式、高吞吐、低延迟的核心特性，同时适配信创软硬件环境，提供更稳定的运行保障、更便捷的运维支持及更全面的安全能力，成为 Kafka 信创替代的优选方案。

2.3 核心能力

核心能力	解释
Kafka 协议完全兼容	整实现 Kafka 生产消费协议、分区分配协议、Offset 管理协议、集群协调协议等核心协议，行为与原生 Kafka 一致。
全版本客户端适配	支持 Java、Go、Python、C++ 等 Kafka 官方客户端（2.x-3.x 版本）及第三方开源客户端，无代码改造即可接入。
生态工具无缝复用	兼容 Kafka-console-producer/consumer、kafka-topics.sh、kafka-configs.sh 等原生工具，支持 Kafka Connect、MirrorMaker 等数据同步工具。
信创环境深度适配	适配主流信创操作系统（麒麟、统信等）、信创芯片（ARM、RISC-V 等），兼容信创数据库及中间件生态。
Kafka 核心特性继承	支持主题分区、副本机制、消费者组、消息压缩、Offset 提交（自动 / 手动）等 Kafka 原生核心特性。

2.4 产品架构

1.架构概述

ADMQ for kafka产品采用 Kafka 经典分布式架构，完全遵循 Kafka 通信协议与与存储模型。产品整体架构分为管控台、核心引擎和分布式协调三大模块，具备高可靠、高可用、可扩展的特性。

一、管控台（Console）

- 管控台是面向运维人员和管理员的可视化操作界面，提供全生命周期管理能力：
- 集群管理：对消息集群节点进行接入和状态监控。
- 用户权限：基于角色的访问控制（RBAC），保障操作安全。
- 主题管理：创建、删除、配置消息主题（Topic），管理分区和副本。
- 订阅管理：管理消费者组（Consumer Group）、订阅关系和消费位点。
- 消息检索：支持按时间、Key、内容等条件查询历史消息。

二、核心引擎（Core Engine） 核心引擎是消息处理的核心，兼容 Kafka 协议，保障消息的可靠传输与存储：

Kafka 协议接入 & 安全访问控制

- 提供标准 Kafka 协议兼容层，支持现有 Kafka 客户端无缝迁移。
- 集成认证（如 SASL）、授权和传输加密（SSL/TLS），保障访问安全。

消息处理服务

- 消息生产接入：接收生产者（Producer）发送的消息，进行校验和路由。
- 消息消费服务：向消费者（Consumer）推送或拉取消息，支持顺序消费和批量消费。
- 消息持久化存储：将消息写入磁盘，支持高吞吐写入和高效检索，保证数据不丢失。

- 元数据服务：管理主题、分区、副本、消费者组等元数据信息，确保集群状态一致。

高可用与负载管理

- 多副本同步：通过多副本机制（如 ISR）保证数据冗余，防止单点故障。
- 故障恢复与副本迁移：节点故障时自动切换，支持副本在集群内迁移以平衡负载。
- 负载均衡：将生产和消费请求均匀分配到集群节点，提升整体吞吐量。
- 状态监控：实时监控节点、分区和副本的健康状态，为故障恢复提供依据。

三、分布式协调（Distributed Coordination）

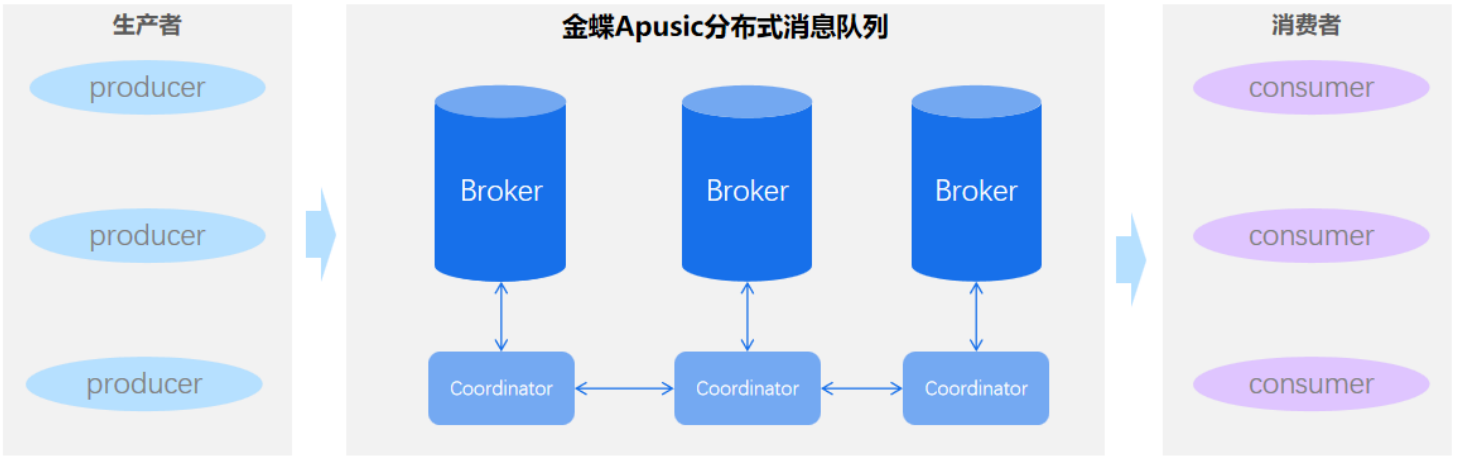
分布式协调模块（通常基于 ZooKeeper 或自研协调器）负责维护集群的一致性和状态：

- 集群成员管理：维护集群中所有 Broker 节点的在线 / 离线状态。
- 元数据持久存储：存储并同步主题、分区、副本分配等核心元数据。
- Controller 选举：在集群中选举出 Controller 节点，负责分区管理和故障处理。
- 分区 Leader 选举：当分区 Leader 节点故障时，自动从 Follower 中选举新的 Leader。
- 集群状态监听通知：当集群状态（如节点上下线、Leader 变更）发生变化时，向相关组件发送通知，触发相应处理。

2.5 产品优势

- 零改造信创替代：无需修改业务代码、无需调整客户端配置（仅需变更连接地址），即可直接替换原有 Kafka 集群，迁移成本趋近于零。
- 生态完全兼容：Kafka 原生工具、监控系统（Prometheus+Grafana）、数据同步工具等均可直接复用，无需额外适配开发。
- 信创环境适配：全面兼容信创软硬件生态，满足政企信创项目落地要求，提供端到端的信创解决方案。
- 特性原汁原味：完全遵循 Kafka 核心特性规范，支持用户熟悉的所有业务场景（如高吞吐数据传输、流式数据处理等），业务无感知切换。
- 安全合规增强：在保留 Kafka 原有安全能力基础上，新增安全特性，符合信创项目安全合规标准。

2.6 应用场景



- 信创项目替代场景：政企信创改造中，直接替换原有 Kafka 集群，无需重构业务系统，快速满足信创落地要求。
- 高吞吐数据传输场景：如日志采集、业务数据同步等，继承 Kafka 高吞吐特性，支撑海量数据实时传输。
- 流式数据处理场景：兼容 Flink、Spark Streaming 等流式计算引擎，无缝对接原有流式数据处理链路。
- 分布式系统解耦场景：基于 Kafka 原生的发布订阅模式，实现分布式系统间异步通信，降低系统耦合度。

3 产品安装

3.1 安装概述

ADMQ for kafka包含核心引擎、分布式协调、管控台三个部分，其中核心引擎受license控制，需要在安装时申请并导入license。管控台为可选部件，如需要可视化管控时，可单独安装管控台。

3.2 单机部署

本文档指导用户在 Linux 环境下完成 ADMQ for Kafka 的单机模式部署，包含环境准备、安装配置、授权申请、服务启动及状态验证。

3.2.1 1. 环境准备与软件安装

3.2.1.1 1.1 创建部署目录

首先创建应用根目录并进入：

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

3.2.1.2 1.2 下载并解压软件包

可通过产品光盘拷贝产品介质或向项目的销售或实施人员获取。

```
# 拷贝到部署服务器

产品安装前需要将产品包拷贝到对应目录下。产品包名称可能随版本和发布日期而变化，这里以20260319发布的产品包为例。

# 解压安装包
tar zxvf ADMQ-V2.0.6.391-Kafka-20260319.tar.gz
```

3.2.1.3 1.3 目录结构说明

解压完成后，当前目录下将生成 `admq-kafka` 。标准目录结构如下：

目录名	说明
`bin`	存放启动、停止及管理脚本
`config`	存放配置文件 (如 `kafka-standalone.conf`)
`data`	存放 Kafka 消息数据及 ZooKeeper 数据
`jdk`	内置的 Java 运行环境，无需额外配置系统 JDK
`logs`	存放运行日志 (Broker 日志、ZK 日志等)
`service`	系统服务注册相关文件 (可选)

3.2.2 2. 配置修改

3.2.2.1 2.1 编辑监听地址

修改单机模式配置文件，绑定指定的 IP 地址和端口。

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

修改内容:
找到 `listeners` 配置项，将其修改为服务器的实际 IP 地址（示例中使用 `192.168.1.10`）：

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

3.2.3 3. 授权许可

ADMQ for kafka 产品需要 License 文件才能正常运行。

1. **申请授权**：联系管理员或通过授权平台申请对应版本的 `license.xml` 文件。
2. **放置文件**：将获取到的 `license.xml` 文件上传至部署目录根路径。

```
# 确认文件位置
ls -l /apusic/admq-kafka/license.xml
```

3.2.4 4. 启动服务

ADMQ for Kafka 单机模式依赖 ZooKeeper，需按顺序启动。所有操作请在安装根目录下执行。

3.2.4.1 4.1 启动 ZooKeeper

首先启动内置的 ZooKeeper 服务：

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

```
[root@master admq-kafka]# bin/admq-daemon start kafka zk
Starting kafka-zk, logging to /tmp/admq-kafka/logs
[root@master admq-kafka]#
```

3.2.4.2 4.2 启动 Kafka Broker

待 ZooKeeper 启动成功后（通常等待 3-5 秒），启动 Kafka 主服务：

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

```
[root@master admq-kafka]# bin/admq-daemon start kafka standalone
Starting kafka-standalone, logging to /tmp/admq-kafka/logs
授权用户：深圳市金蝶天燕云计算股份有限公司
开始时间：2025-12-15
失效时间：2026-03-31
[root@master admq-kafka]#
```

3.2.5 5. 服务验证

3.2.5.1 5.1 检查进程状态

使用 `ps` 命令查看进程，正常状态下应能看到 **两个** 主要 Java 进程（一个 ZK，一个 Broker）：

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

预期输出示例：

```
[root@master admq-kafka]# ps -ef |grep admq-kafka
root       7290     1  1 14:22 pts/4    00:00:02 /tmp/admq-kafka/jdk/aarch64/bin/java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGCInvokesConcurrent -XX:MaxInlineLevel=15 -Djava.awt.headless=true -Xlog:gc*:file=/tmp/admq-kafka/logs/zookeeper-gc.log:time,tags:filecount=10,filesize=100M -Dcom.sun.management.jmxremote=true -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dkafka.logs.dir=/tmp/admq-kafka/logs -Dlog4j.configuration=file:/tmp/admq-kafka/config/log4j-zk.properties -cp /tmp/admq-kafka/kafka/bin/../run_libs/* org.apache.zookeeper.server.quorum.QuorumPeerMain /tmp/admq-kafka/config/kafka-zk.conf
root      24794     1 50 14:24 pts/4    00:00:06 /tmp/admq-kafka/jdk/aarch64/bin/java -Xmx4G -Xms4G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+ExplicitGCInvokesConcurrent -XX:MaxInlineLevel=15 -Djava.awt.headless=true -Xlog:gc*:file=/tmp/admq-kafka/logs/kafkaServer-gc.log:time,tags:filecount=10,filesize=100M -Dcom.sun.management.jmxremote=true -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false -Dkafka.logs.dir=/tmp/admq-kafka/logs -Dlog4j.configuration=file:/tmp/admq-kafka/config/log4j.properties -cp /tmp/admq-kafka/kafka/bin/../run_libs/* com.apusic.admq.kafka.ApusicKafka /tmp/admq-kafka/config/kafka-standalone.conf
root      26491 15860  0 14:25 pts/4    00:00:00 grep --color=auto admq-kafka
```

3.2.5.2 5.2 检查端口监听

使用 `netstat` 确认 9092 端口是否处于 `LISTEN` 状态：

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

预期输出示例：

```
[root@master admq-kafka]# netstat -nltup |grep 9092
tcp6       0      0 172.20.140.224:9092     :::*                    LISTEN      24794/java
[root@master admq-kafka]#
```

如果未看到输出，请检查防火墙设置或查看日志排查启动失败原因。

3.2.5.3 5.3 查看运行日志

如果服务启动异常或需监控运行状态，请查看 `logs` 目录下的日志文件：

```
# 查看实时日志
tail -f logs/server.log
tail -f logs/zookeeper.log

# 或者查看目录结构
ls -lh logs/
```

重点关注日志中是否有 `ERROR` 级别的信息。

3.2.6 6.停止服务

操作	命令
停止 ZooKeeper	`bin/admq-daemon stop kafka zk`
停止 Kafka	`bin/admq-daemon stop kafka standalone`

3.3 集群部署

3.3.1 1. 部署规划

假设部署 3 节点 高可用集群（ZooKeeper 与 Kafka Broker 混合部署），规划如下：

节点	IP 地址	ZK MyID	Broker ID	ZK 端口	Kafka 端口
Node 1	192.168.1.10	1	0	2181	9092
Node 2	192.168.1.11	2	1	2181	9092
Node 3	192.168.1.12	3	2	2181	9092

> ***前置要求***：
>
> 1. *确保三台机器时间同步（NTP）。*
> 2. *关闭防火墙或开放端口：* `2181` *(ZK),* `2888` *(ZK 选举),* `3888` *(ZK 选举),* `9092` *(Kafka)。*

3.3.2 2. 所有节点执行：基础安装

在 Node 1, Node 2, Node 3 上分别执行：

3.3.2.1 2.1 创建目录并解压

```
mkdir -p /apusic
cd /apusic
tar zxvf ADMQ-V2.0.6.391-Kafka-20260212.tar.gz
cd admq-kafka
```

3.3.2.2 2.2 放置授权文件

将 `license.xml` 上传至 `/apusic/admq-kafka/` 目录。

3.3.3 3. 配置 ZooKeeper 集群

ZooKeeper 集群需要修改配置文件并在数据目录创建 `myid` 文件。

3.3.3.1 3.1 修改 ZK 配置文件

编辑 `config/kafka-zk.conf` 。

在所有三个节点上，配置内容完全一致：

```
### --- 集群配置 --- ###
# 注意：这里的 1, 2, 3 对应后面要创建的 myid 文件内容
#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
```

3.3.3.2 3.2 创建 MyID 文件 (每台机器不同)

在 Node 1 (192.168.1.10) 上执行：

```
# 创建数据和日志目录
mkdir -p /apusic/admq-kafka/data/zk/data
mkdir -p /apusic/admq-kafka/data/zk/logs
# 写入 MyID (必须与配置文件中 server.1 的 1 对应)
echo "1" > /apusic/admq-kafka/data/zk/data/myid
```

在 Node 2 (192.168.1.11) 上执行：

```
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
```

在 Node 3 (192.168.1.12) 上执行：

```
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
```

3.3.4 4. 配置 Kafka Broker 集群

编辑 `config/kafka-broker.conf` 。

3.3.4.1 4.1 Node 1 配置（`192.168.1.10`）

```
# 唯一 Broker ID（集群内不可重复）
broker.id=0

# 监听地址（绑定本机 IP）
listeners=PLAINTEXT://192.168.1.10:9092

# 连接 ZK 集群地址（所有节点配置必须一致）
zookeeper.connect=192.168.1.10:2181,192.168.1.11:2181,192.168.1.12:2181
zookeeper.connection.timeout.ms=18000
```

3.3.4.2 4.2 Node 2 配置（`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
```

3.3.4.3 4.3 Node 3 配置（`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
```

3.3.5 5. 启动集群

严格遵循启动顺序：先 ZK 集群，后 Kafka 集群。

3.3.5.1 5.1 第一步：启动所有节点的 ZooKeeper

依次登录 Node 1, 2, 3 执行：

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

3.3.5.2 5.2 第二步：启动所有节点的 Kafka Broker

依次登录 Node 1, 2, 3 执行：

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

3.3.6 6. 验证与测试

3.3.6.1 6.1 验证 ZooKeeper 集群状态

在任意节点执行四字母命令 `ruok` ：

```
# 检查 ZK 角色 (Leader/Follower)
echo ruok | nc 192.168.1.10 2181
echo ruok | nc 192.168.1.11 2181
echo ruok | nc 192.168.1.12 2181
```

预期执行结果，返回imok。

```
echo ruok | nc 192.168.1.10 2181
imok

echo ruok | nc 192.168.1.11 2181
imok

echo ruok | nc 192.168.1.12 2181
imok
```

3.3.6.2 6.2 验证 Kafka 集群元数据

使用命令行工具查看集群 Broker 列表：

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

3.3.6.3 6.3 功能测试 (创建 Topic 并生产消费)

1. **创建 Topic** (3 副本，确保数据分布在所有节点)：

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

2. **查看 Topic 详情** (确认 ISR 和 Leader 分布)：

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

预期输出示例：

```
Topic: test-cluster  PartitionCount: 3  ReplicationFactor: 3
  Partition: 0  Leader: 0  Replicas: 0,1,2  Isr: 0,1,2
  Partition: 1  Leader: 1  Replicas: 1,2,0  Isr: 1,2,0
  Partition: 2  Leader: 2  Replicas: 2,0,1  Isr: 2,0,1
```

(Leader 均匀分布在不同节点，ISR 包含所有节点，说明集群健康)

3. **生产与消费测试**：

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

# 消费者 （新开终端）
kafka/bin/kafka-console-consumer.sh --topic test-cluster --from-
beginning --bootstrap-server 192.168.1.10:9092
```

3.3.7 7. 停止服务

```
# 先停 Kafka
bin/admq-daemon stop kafka broker
```

```
# 后停 ZK
bin/admq-daemon stop kafka zk
```

3.4 管理控制台部署

3.4.1 安装前准备

- 获取安装包：从 http://www.apusic.com/下载金蝶 Apusic分布式消息队列安装包，或从金蝶Apusic 分布式消息队列产品光盘中获得相应的安装包文件。
- 系统要求：Java环境（管控台JDK17以上，节点JDK1.8及以上版本）、内存（16GB+）、硬盘空间（100GB+）、浏览器（IE8及以上，FireFox，Chrome）

3.4.2 开始安装

3.4.2.1 获取上传产品包

ADMQ for kafka管控台安装包为admq-manager开头的压缩包，例如admq-manager-V3.0.1.tar.gz

获取安装包后，可将其上传到服务器的任意目录下。

3.4.2.2 管理控制台安装

解压安装包

```
mv admq-manager-V3.0.1.tar.gz /opt/
cd /opt
tar -zxvf admq-manager-V3.0.1.tar.gz
```

修改数据库类型（可选）

默认使用H2数据库，如果正式环境部署需要换成其他数据库，例如MySQL。

/opt/admq-manager-V3.0.1/config/application.properties 文件中添加了多种数据库配置，可以把使用的数据库配置打开，注释掉其他不使用的数据库配置项。

1. 注释掉H2数据库相关配置，打开mysql的相关配置

```
# 数据库配置（多选一）
#spring.config.import.03=file:./config/other-config/db-h2.properties
spring.config.import.03=file:./config/other-config/db-mysql.properties
#spring.config.import.03=file:./config/other-config/db-kingbase.properties
#spring.config.import.03=file:./config/other-config/db-oscar.properties
#spring.config.import.03=file:./config/other-config/db-dm.properties
#spring.config.import.03=file:./config/other-config/db-pgsql.properties
```

2. 配置MYSQL的地址、端口、数据库名称、用户名、密码，路径如上图 config/other-config/db-mysql.properties

```
# mysql
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://ip:port?useUnicode=true&characterEncoding=utf-8&allowMultiQueries=true&useSSL=false
spring.sql.init.schema-locations=classpath:/META-INF/sql/mysql-schema.sql
spring.datasource.username=root
spring.datasource.password=root
```

3. 管控台启动的时候会自动加载数据库初始化脚本

其他数据库参照此步骤即可

3.4.2.3 启动管理控制台服务

```
cd /opt/admq-manager-V3.0.1/
bin/admq-manager start
```

3.4.2.4 停止管理控制台服务

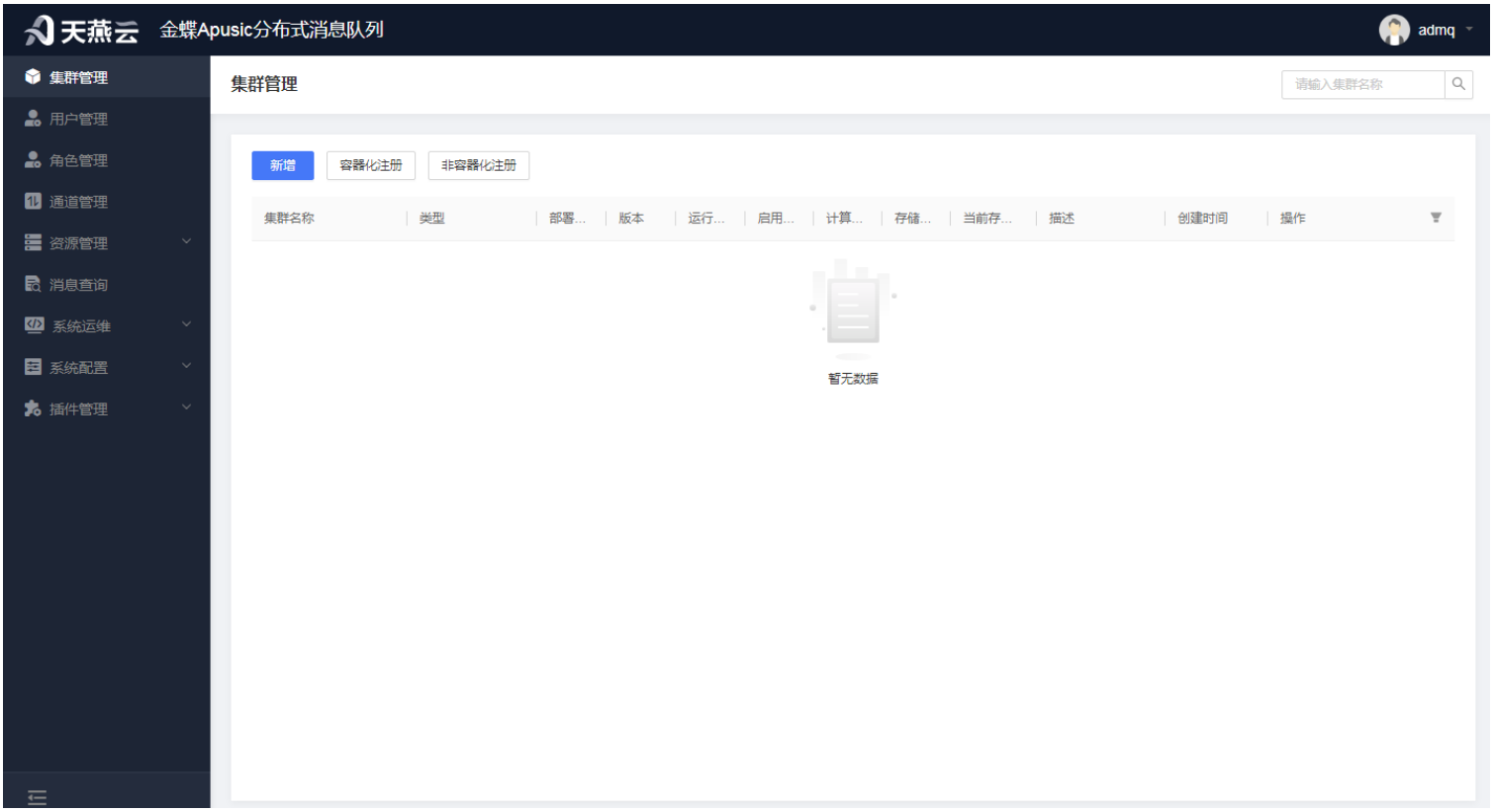
```
cd /opt/admq-manager-V3.0.1/
bin/admq-manager stop
```

3.4.3 验证

1.在浏览器中键入URL：http://ip:12305 或者 https://ip:12306

2.在登录界面输入用户名（admq）,密码(11111111)进行登录，首次登录需要修改密码。

3.若登录成功，会跳转到ADMQ管控台首页，则表示安装成功。



4 管控台使用

4.1 概述

ADMQ fro kafka管控台分为系统管理与运维管理，系统管理主要用于核心引擎的接入，接入后可登录运维管理侧查看引擎状态，并对引擎进行管理维护。

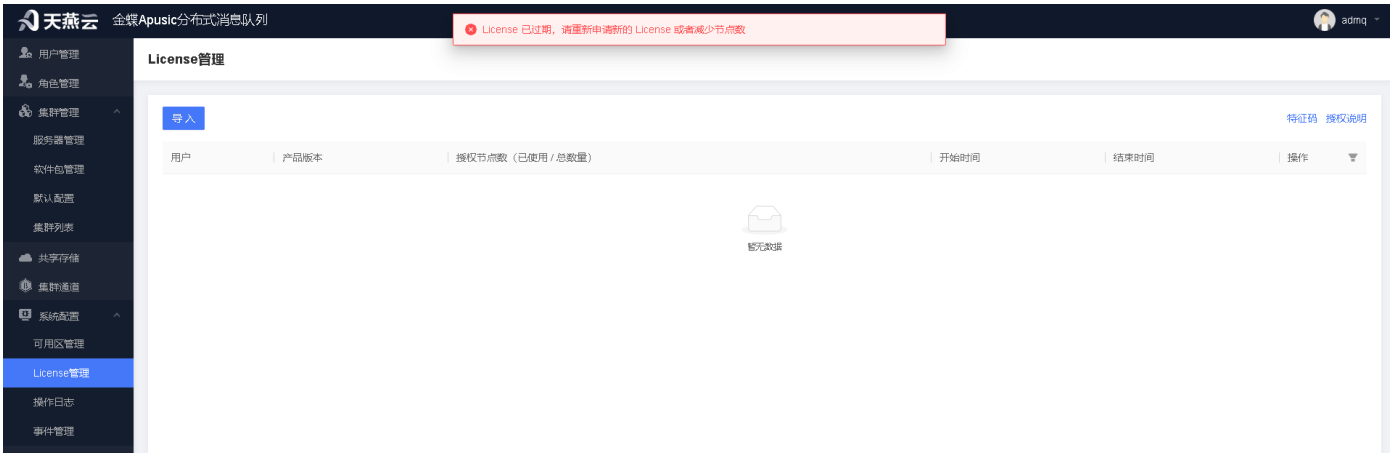
用户需先进入系统管理侧进行集群配置与对接，然后在运维管理侧进行集群管理。

4.2 系统管理

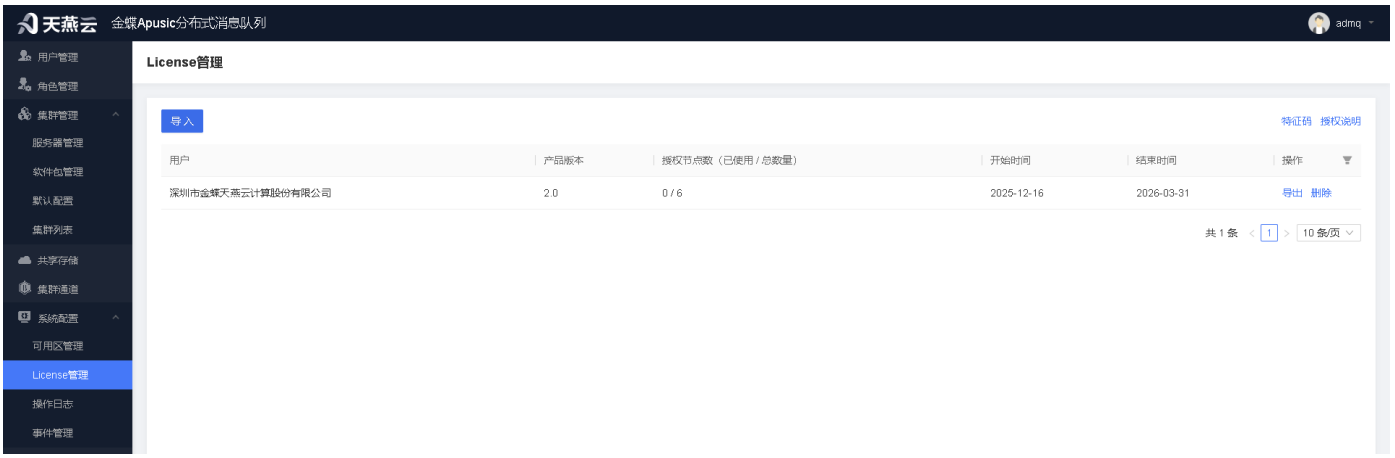
4.2.1 导入licnese

使用管理员登录管控台后，点击菜单会出现“License 已过期，请重新申请新的 License 或者减少节点数”提示。需在系统配置的license管理功能中导入产品的任意license文件。

- 导入前



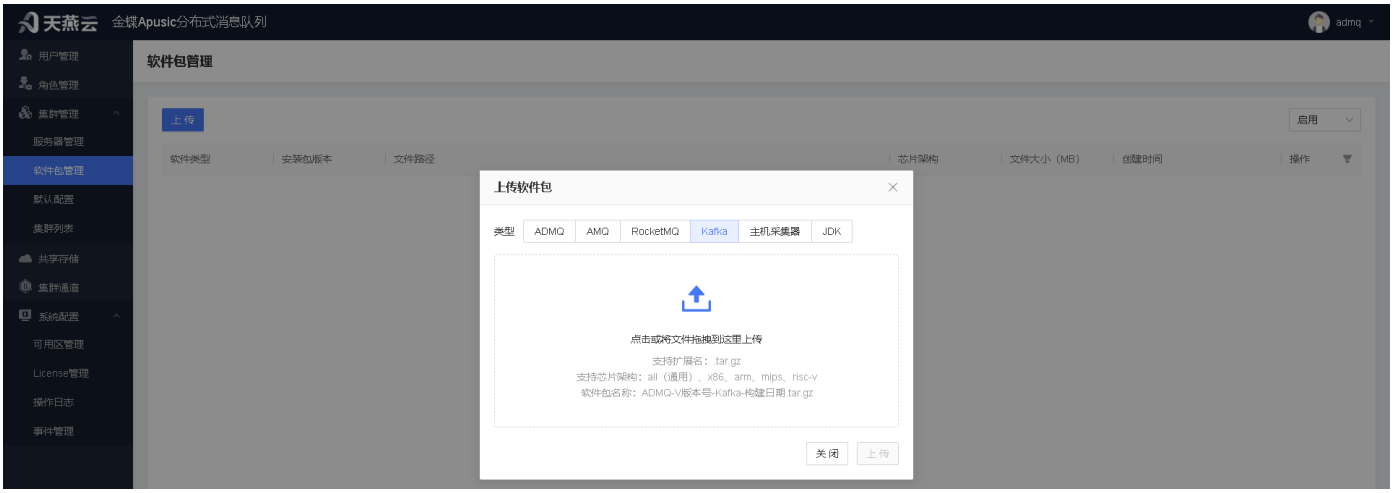
- 导入后



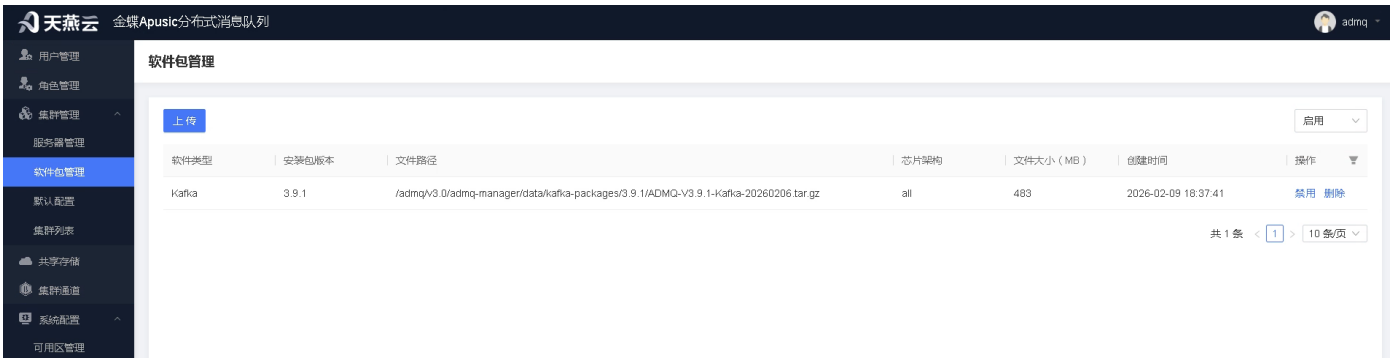
4.2.2 上传产品包

为保障运行一致性，需在软件包管理功能中，上传对应产品的产品包。若未上传产品包，在后续接入时，将无法选择到对应的版本进行接入。

- 上传前

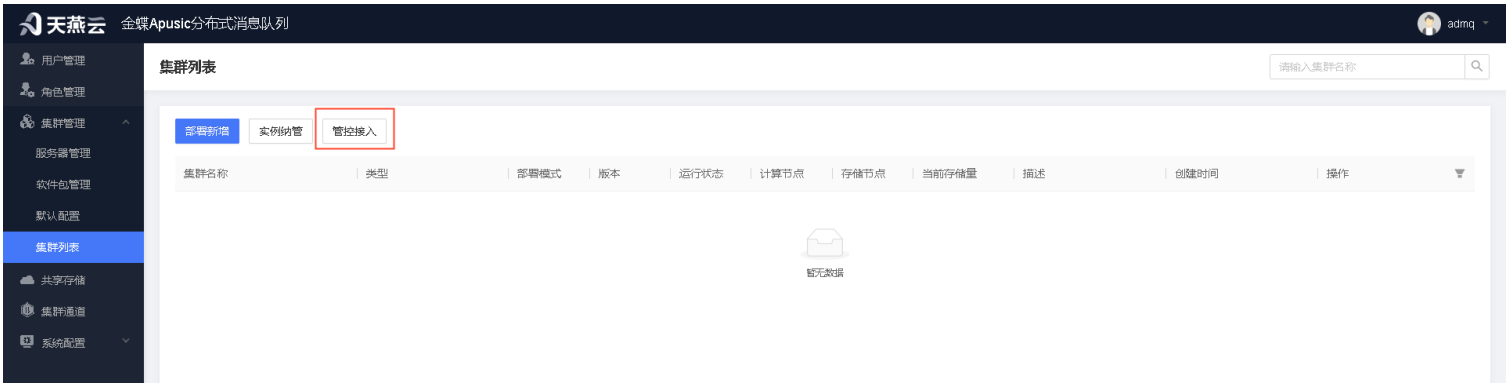


- 上传后

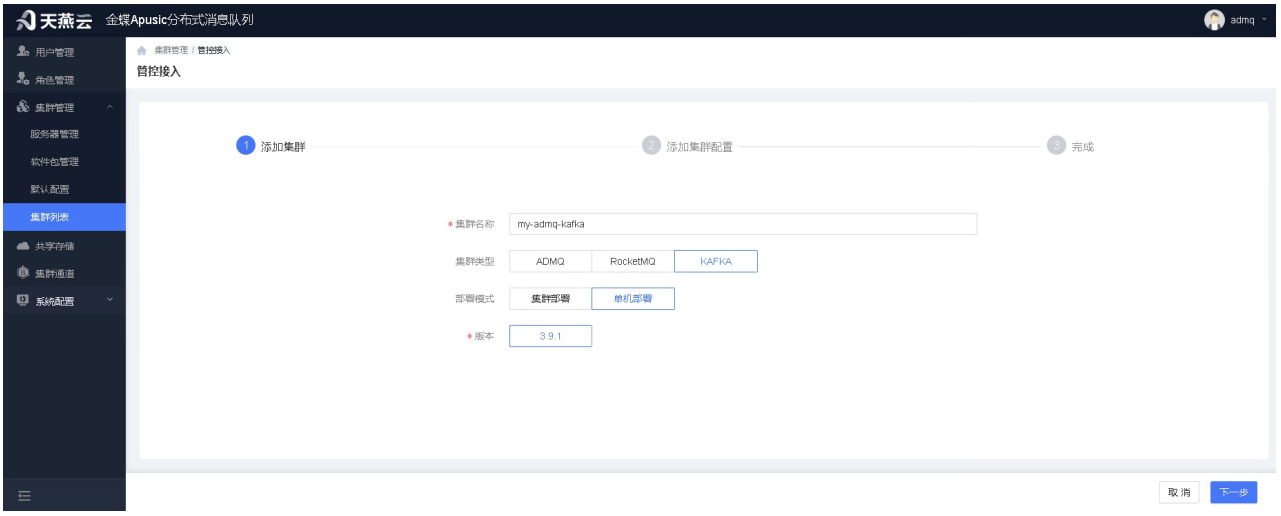


4.2.3 对接集群

如产品引擎已经安装完毕，并确认管控台所在服务器可以访问产品引擎所在的服务器和端口。则可通过集群列表-》管控接入功能对产品引擎进行可视化管理。

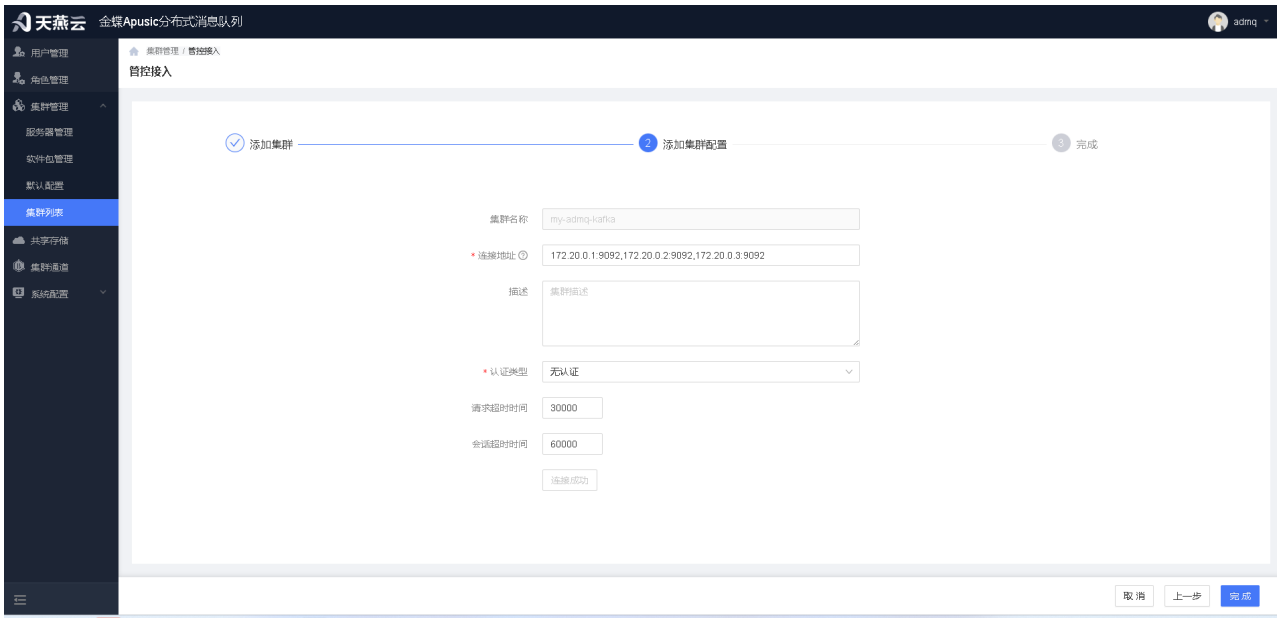


- 填写集群信息
  - 填写集群相关信息，如集群类型、部署模式以及版本



注意：若上传软件包步骤未执行，则无法选择到版本。

- 添加集群配置
    - 添加待连接集群的访问地址和端口。可添加一个集群的多个节点，以便在部分节点失效时，依旧可以保持访问。多个地址之间以逗号分隔。

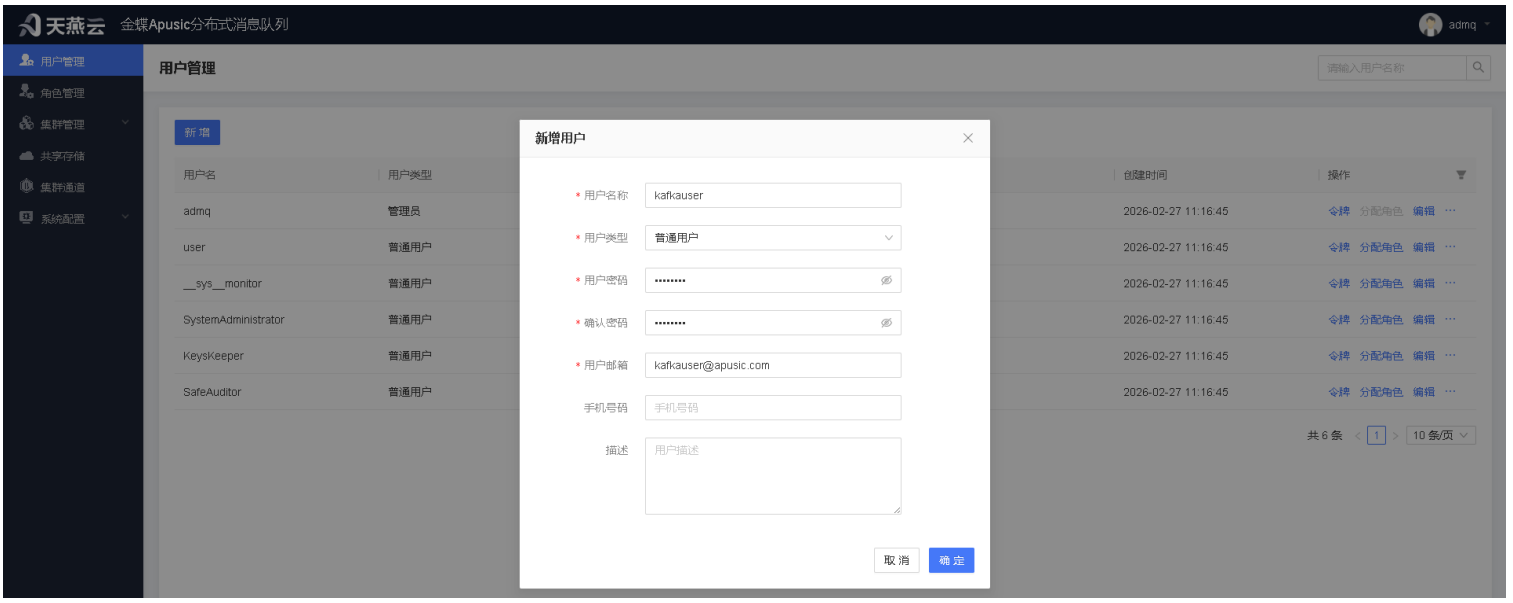


- 查看集群列表与修改配置
    - 添加完成后，回到集群列表，可看到添加的集群信息。并可通过配置功能修改集群接入的配置信息。

4.2.4 开通运维用户

- 添加运维用户

  进入用户管理，新增用户，这里以kafauser为例



- 分配用户角色

点击'分配角色'按钮，打开分配角色界面，为用户分配角色。

4.3 运维管理

4.3.1 使用运维用户登录

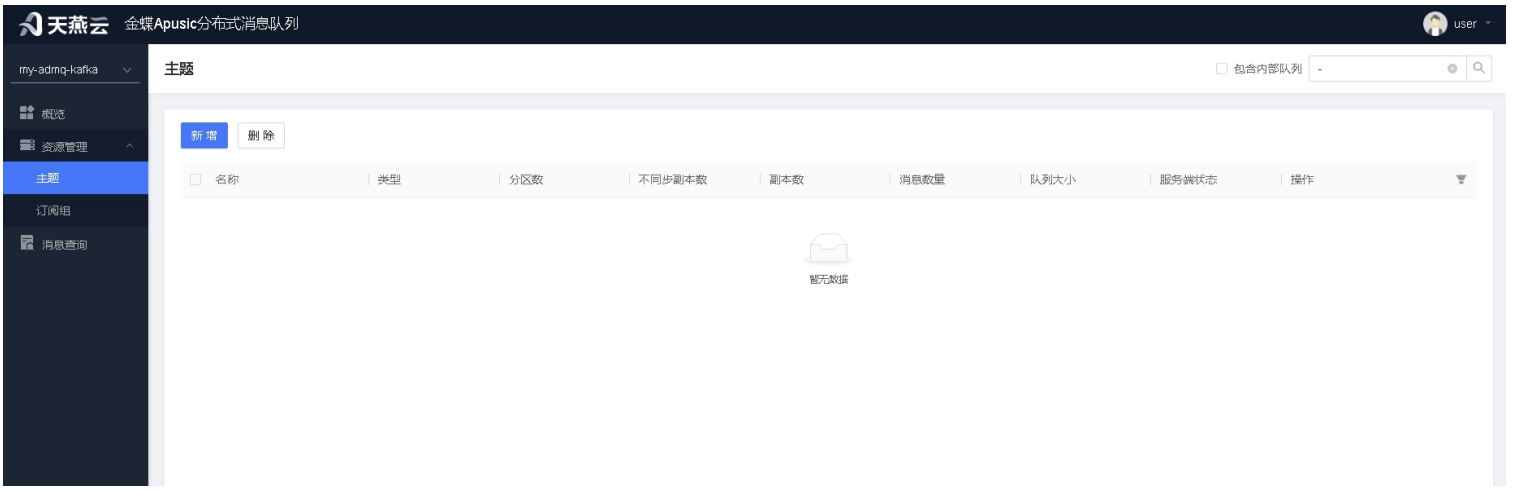
以kafauser为例，输入用户名密码进行登录。用户首次登录时，需要修改密码。用户在修改密码后重新登录，即可进入系统。

4.3.2 概览与切换

运维用户登录后，可同时管理多套系统，通过左上角下拉框进行切换。用户可查看当前实例的连接情况和基本信息。

4.3.3 主题管理

用户可使用管控台进行主题的创建、管理、维护与删除。主题具有缓存功能当服务器离线或者服务器上的主题被删除时，依旧可以通过本地缓存获取主题配置信息，此时字段服务端状态为不可用。对主题的操作也会失败。

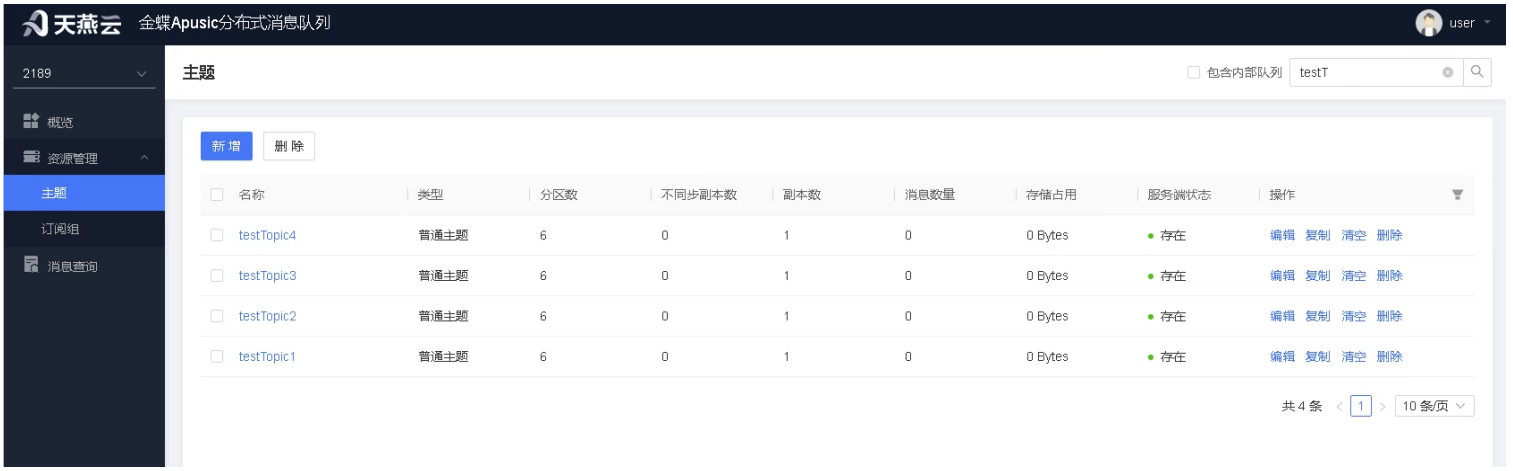


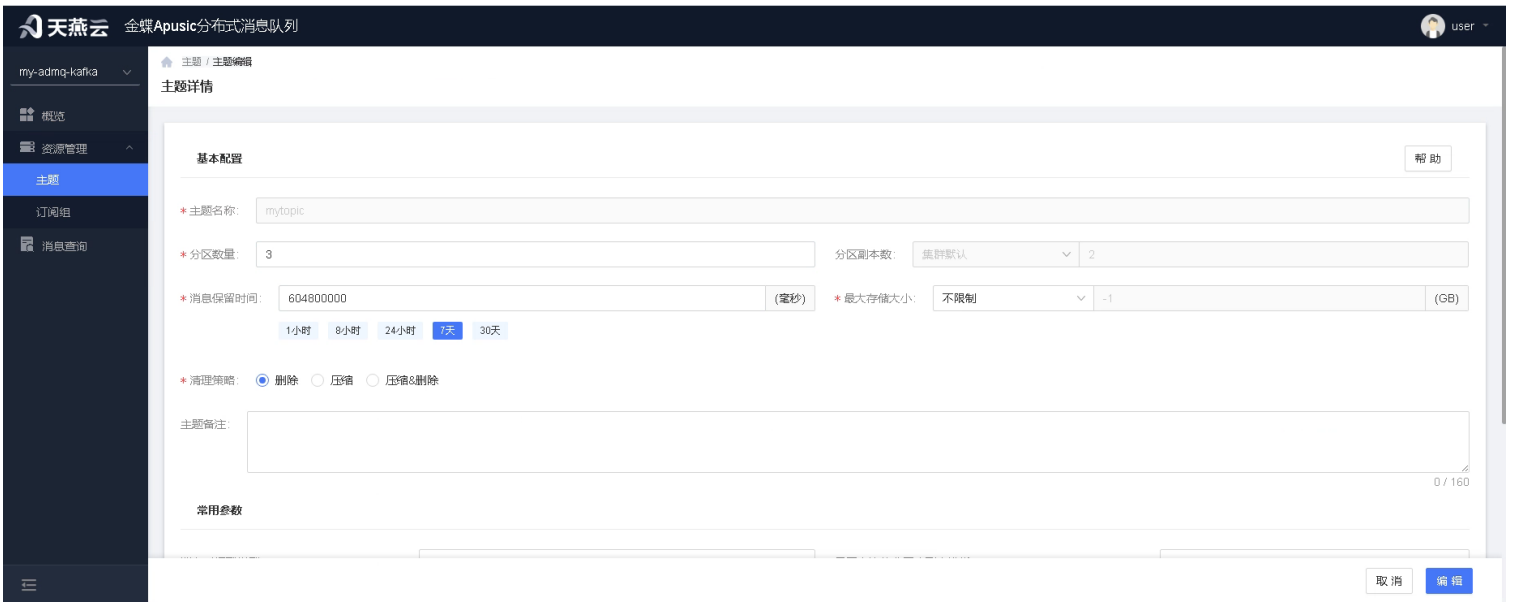
4.3.3.1 主题创建

用户进入主题列表，点击新增按钮，进行主题新增.可根据需要设置分区数量、分区副本数、消息保留时间等参数设置。用户可根据需要进行自定义参数的配置，可参考右上角的帮助，查看可选参数。

4.3.3.2 主题维护

主题创建后可在主题列表中看到，用户可使用修改功能，对主题配置进行修改。主题创建后主题名称与分区副本数不允许修改。



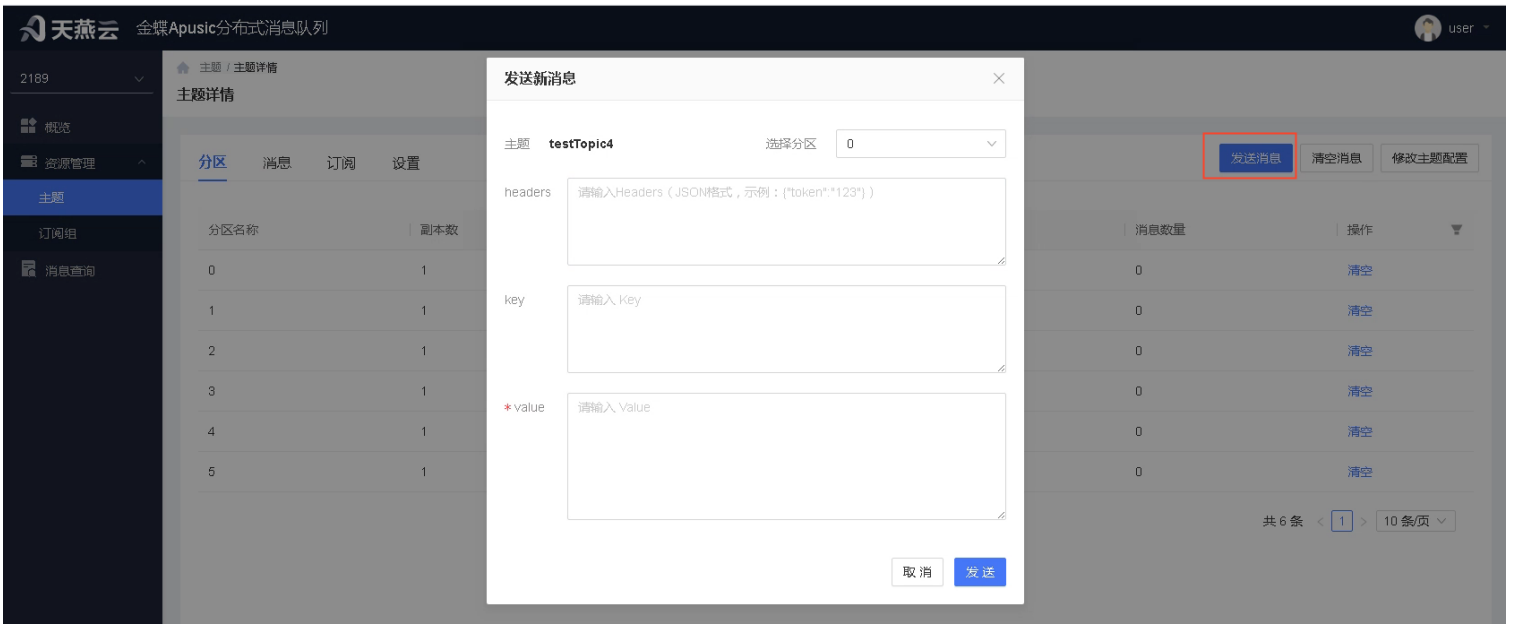


4.3.3.3 主题详情

点击列表中的主题名称，可进入主题详情。主题包含分区、消息、订阅、设置四个部分的功能。用户也可以在主题详情中，进行消息发送和主题清空。

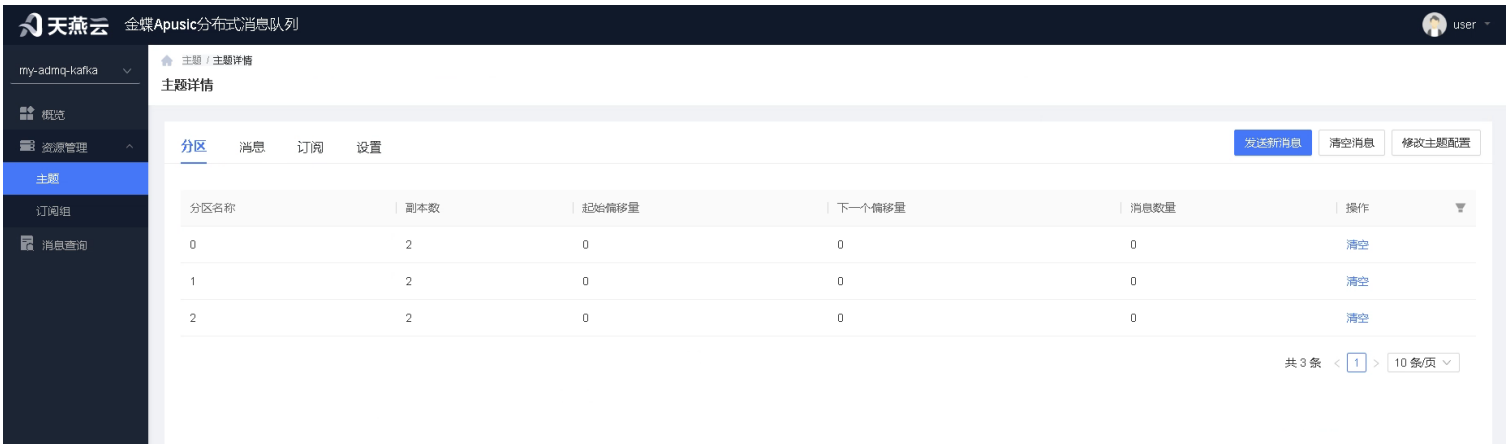
发送消息

用户点击发送新消息后，可填写需发送的消息内容，并点击发送。发送时，需指定发送分区。



分区：

通过分区功能可以查看主题下的所有分区，以及分区的副本数、偏移量信息，也可以清空特定的分区。

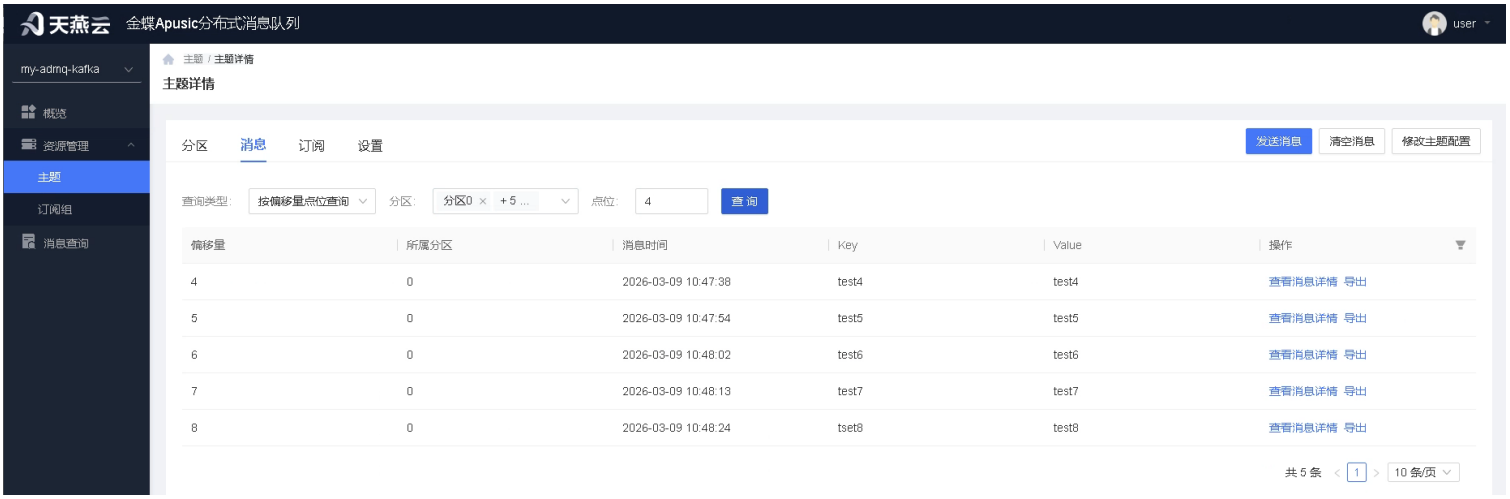


消息：

通过消息功能可检索该主题下的消息，支持两种查询方式：

1.按偏移量点位查询：通过偏移量点位查询，可指定起始偏移量。

2.按时间查询：通过时间查询，可指定开始时间。

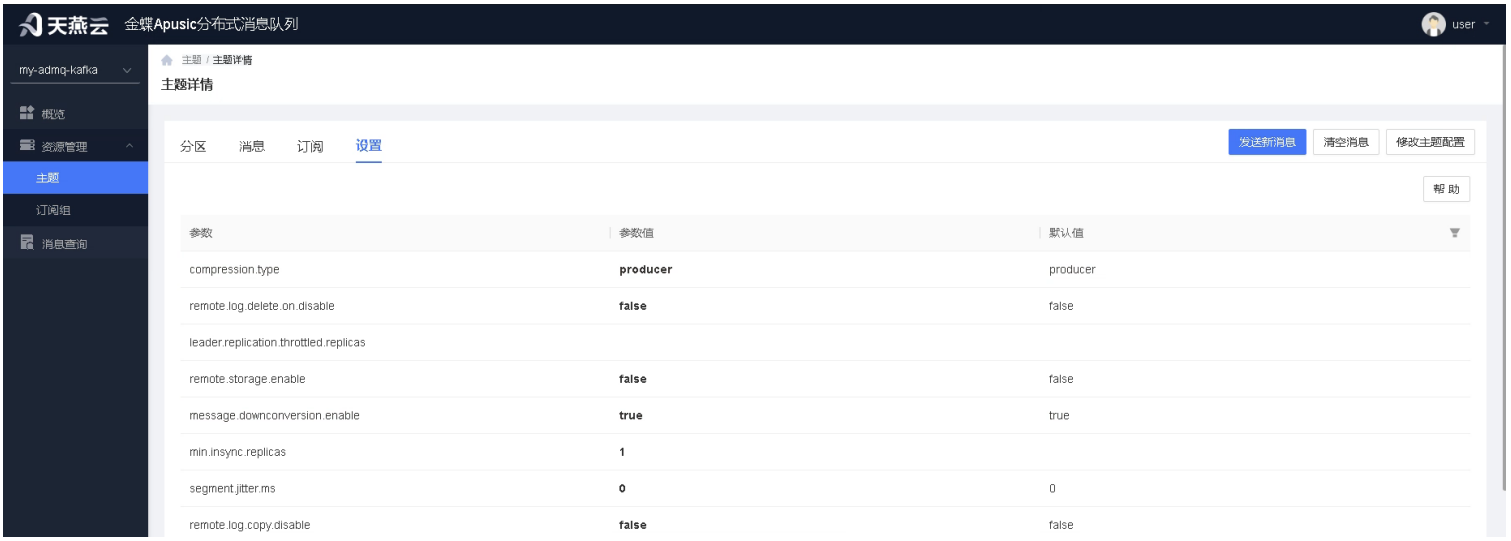


订阅：

用户通过订阅功能，可查看该主题下的所有订阅关系，并查看订阅关系详情。

设置：

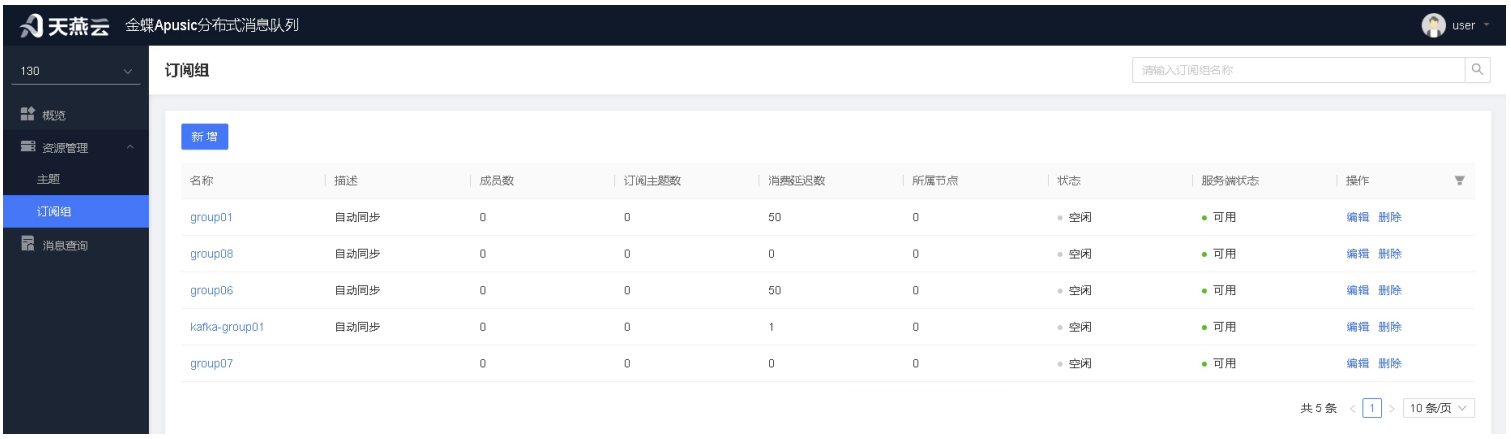
用户通过设置功能，可查看该主题的配置信息，并修改该主题的配置信息。



4.3.4 订阅组管理

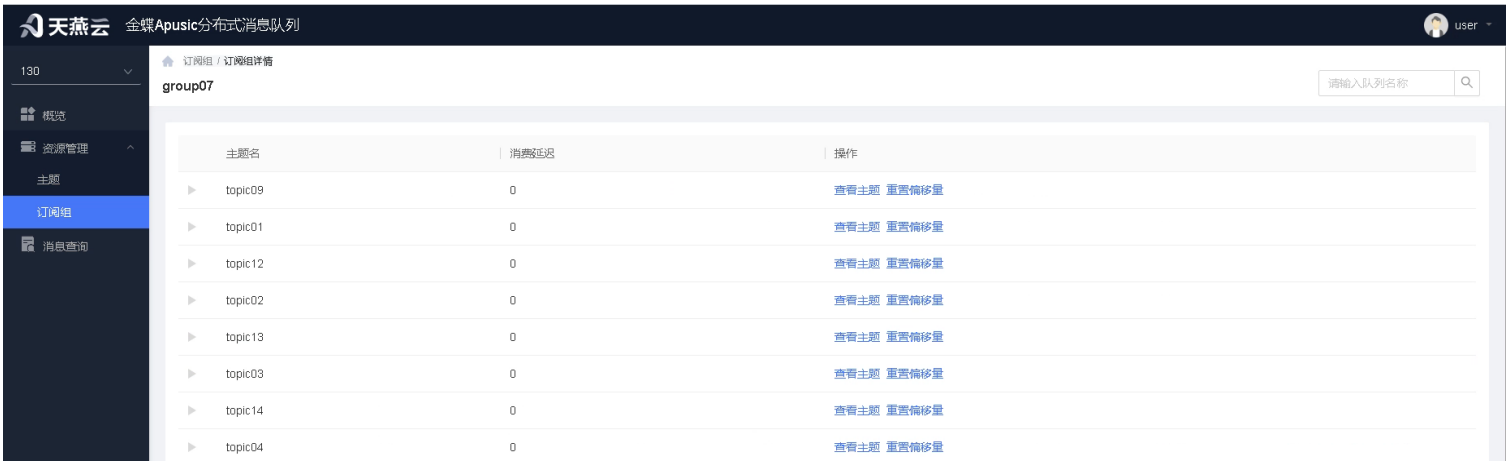
用户可使用管控台进行订阅组的创建、管理、维护与删除。订阅组具有缓存功能，当服务器离线或者服务器上的订阅组被删除时，依旧可以通过本地缓存获取订阅组列表。

注意：为了避免影响数据消费，本地订阅组创建时，不会在服务器上实际创建订阅组。但是当本地订阅组被删除时，服务器上的订阅组也会被同步删除。



4.3.4.1 订阅组与主题

点击订阅组进入订阅组详情，可看到该订阅组的订阅关系，每个订阅主题会有单独的记录，并可查看对应主题的消费延迟情况。

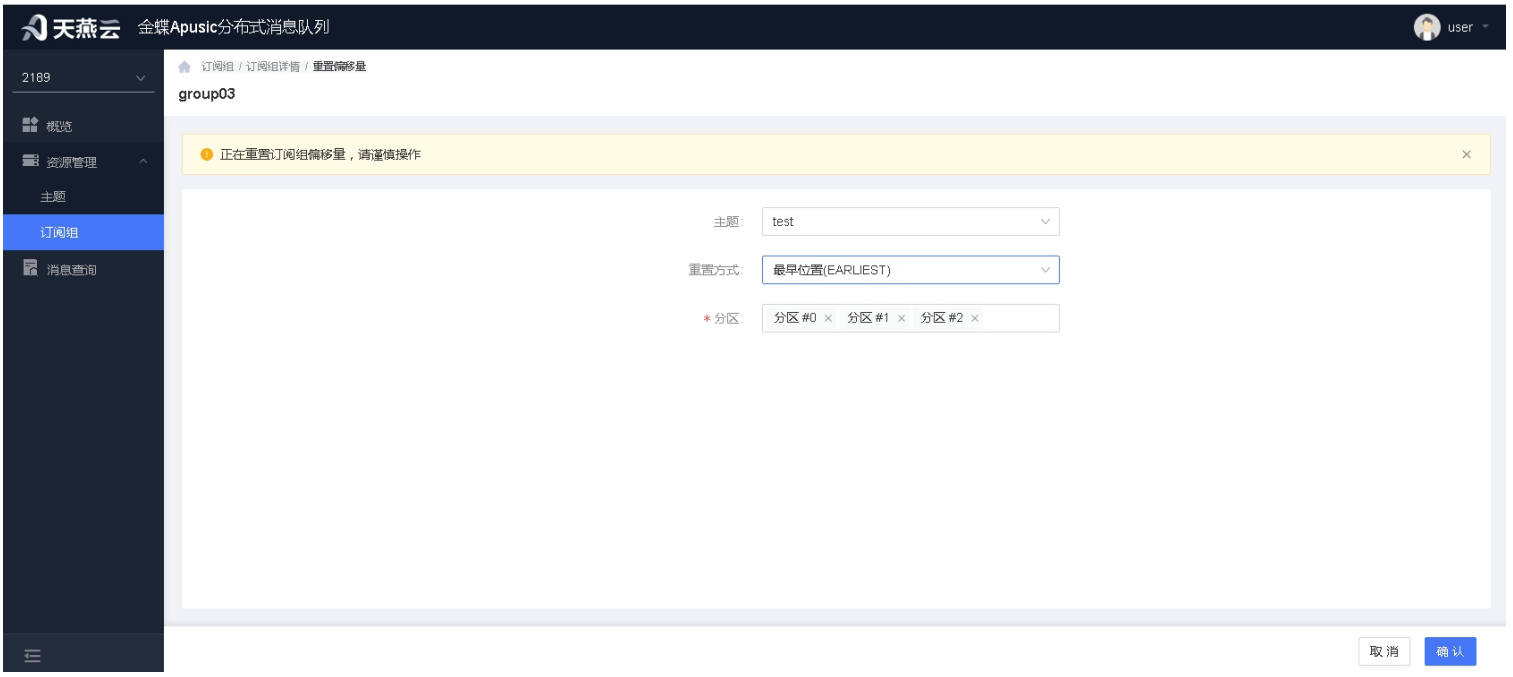


重置偏移量

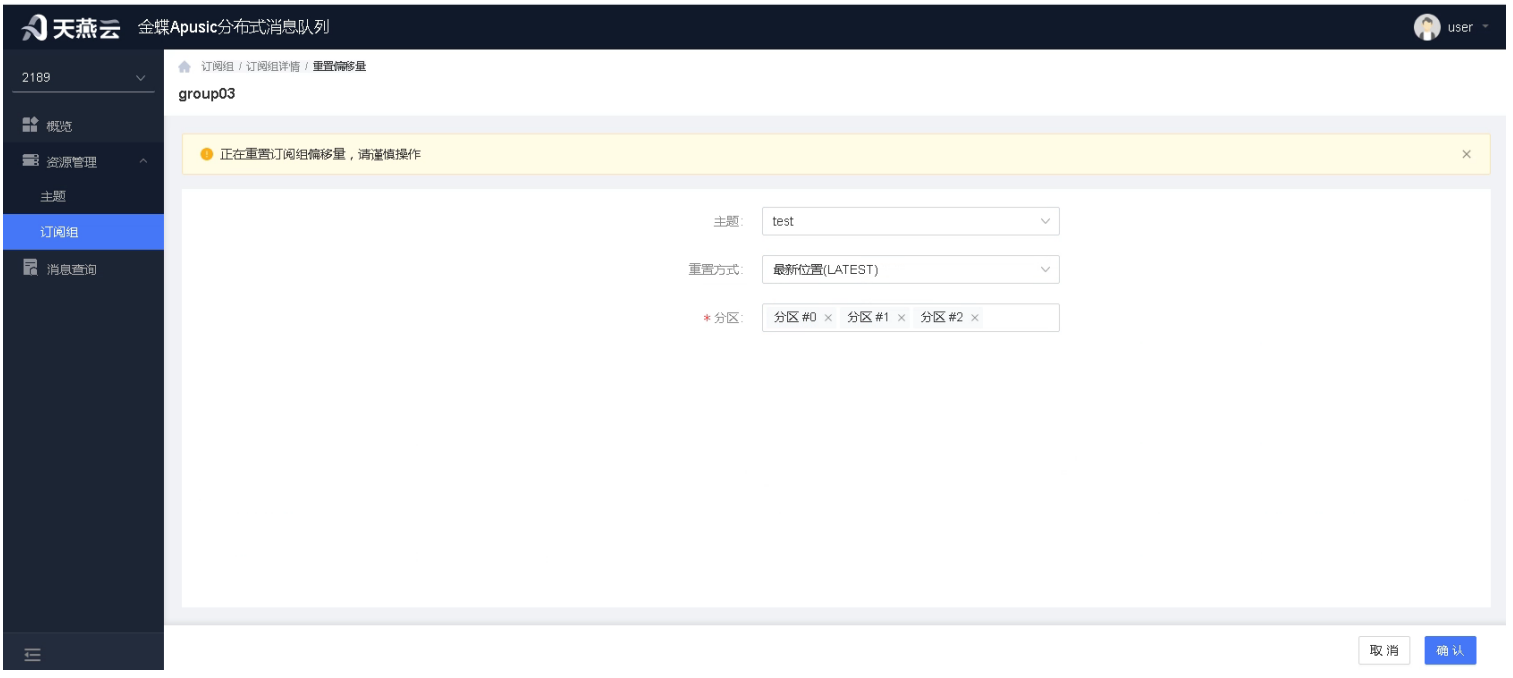
当某个主题消息存在问题，需要重新消费时，可通过重置偏移量功能进行偏移量重置。

重置包含四种方式：

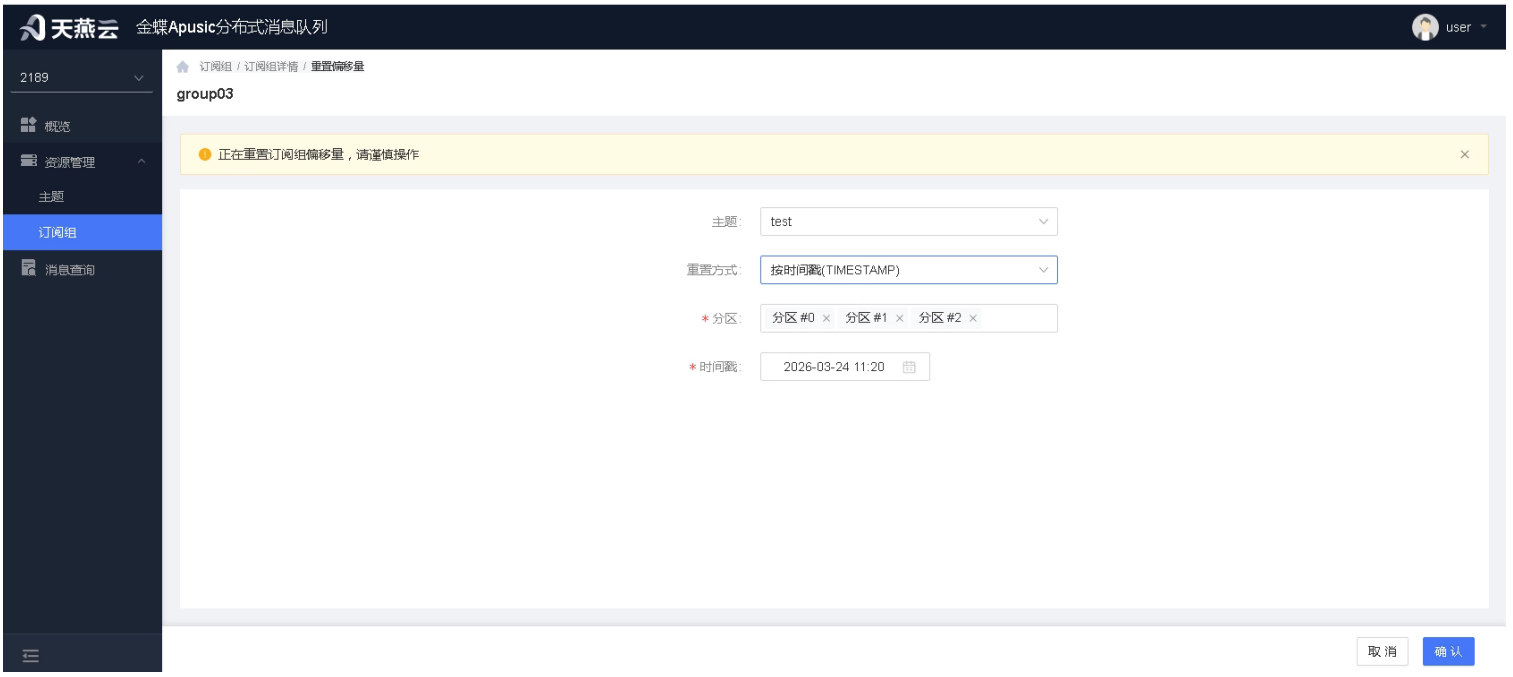
1.**最早位置**：直接将偏移量设置为可达到的最早偏移量，实现现存所有消息的重消费。



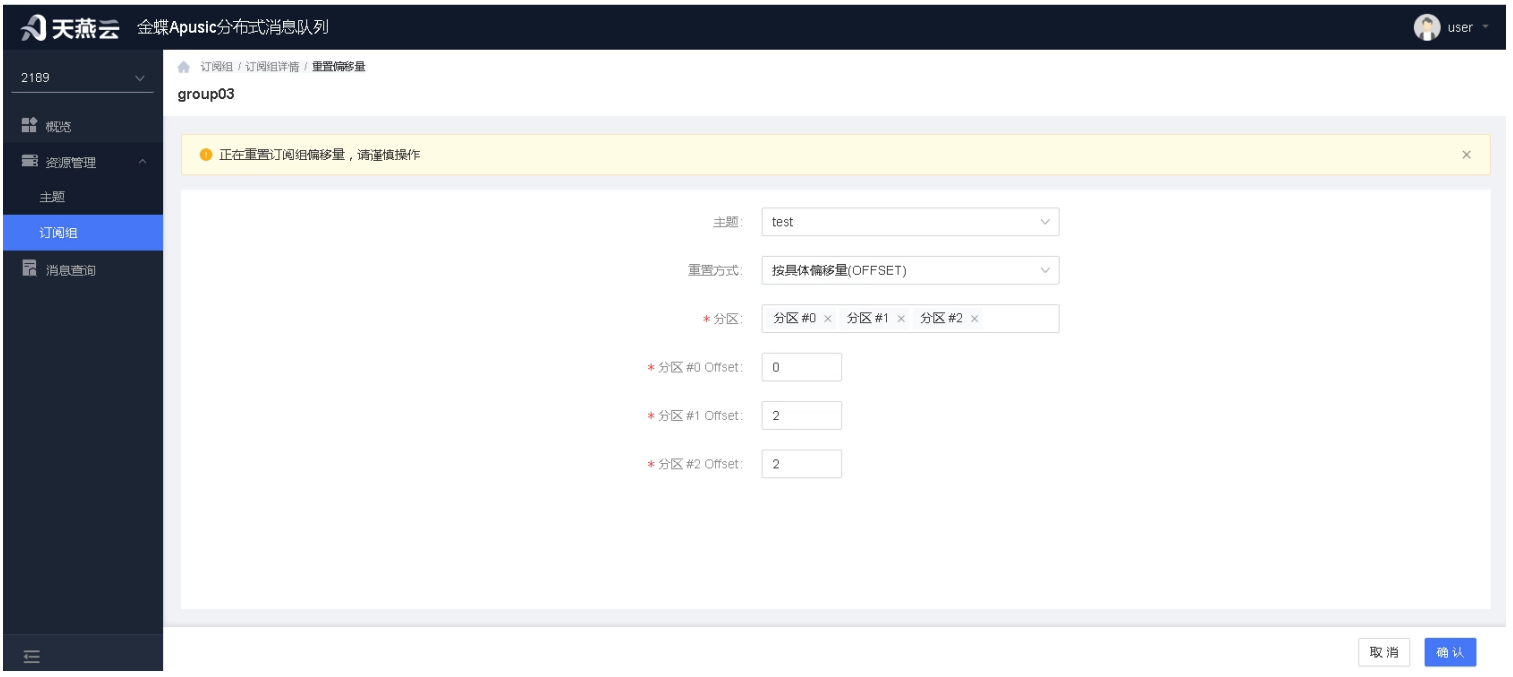
2.**最新位置**：将偏移量设置为最新位置，实现略过所有未消息消息，从最新消息开始消费。



3.**按时间戳**：将偏移量设置为指定时间，从该时间后重新消费消息。

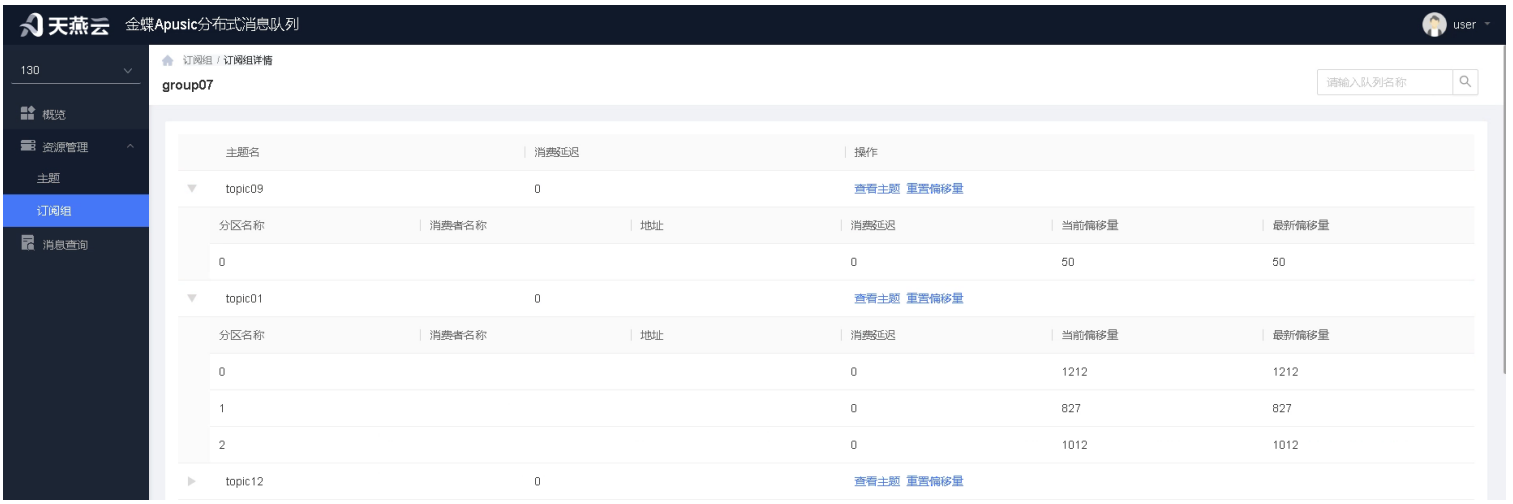


4.**按具体偏移量**：通过为每个分区或所有分区设置偏移量，实现对分区消费点的全面调整和控制。



4.3.4.2 主题消费者

通过展开订阅者详情，可查看订阅组下所有消费者信息。



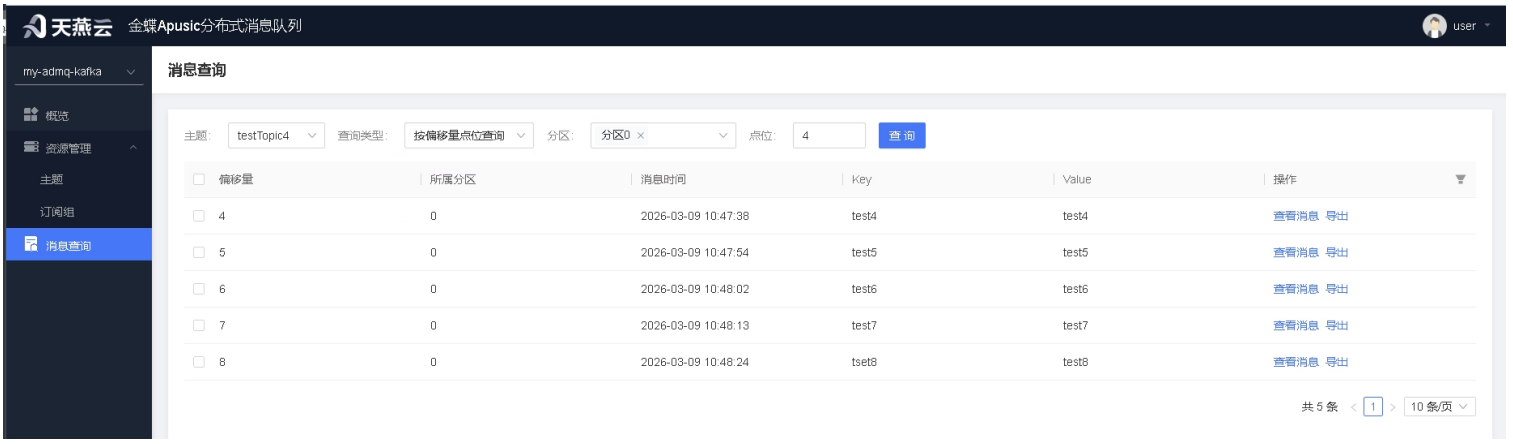
4.3.5 消息查询

用户可通过消息查询对当前实例下的所有主题进行检索，支持两种查询方式：

1.按偏移量点位查询：通过偏移量点位查询，可指定起始偏移量。

2.按时间查询：通过时间查询，可指定开始时间。

为了避免对引擎产生较大的查询压力，故一次返回的数据量有限制，默认为100条。完成查询后，对应的查询结果会显示在列表中，可点击查看消息详情，查看消息的详细内容。



5 配置与维护

5.1 核心引擎目录说明

目录名	说明
`bin`	存放启动、停止及管理脚本
`config`	存放配置文件 (如 `kafka-standalone.conf`)
`data`	存放 Kafka 消息数据及 ZooKeeper 数据
`jdk`	内置的 Java 运行环境，无需额外配置系统 JDK
`logs`	存放运行日志 (Broker 日志、ZK 日志等)
`service`	系统服务注册相关文件 (可选)

5.2 Broker 配置说明

5.2.1 基础配置 (Basic Configuration)

核心作用：定义 Broker 的身份、网络暴露方式及存储路径。

参数	说明	生产环境建议
`broker.id`	**集群唯一标识**。Kafka 通过此 ID 在 ZooKeeper 中注册自己。	**必须唯一**。若重启后 ID 改变，Broker 会被视为新节点，导致旧数据不可见或集群元数据混乱。通常写死在配置文件中，不要动态生成。
`listeners`	**监听地址**。定义 Broker 绑定哪些 IP 和端口，以及使用什么协议。格式：`协议://IP:端口`。	- **内网部署**：`PLAINTEXT://0.0.0.0:9092` (绑定所有网卡) 或指定内网 IP。 - **多网卡/混合云**：可配置多个，如 `INTERNAL://192.168.1.10:9092,EXTERNAL://10.0.0.5:9093`。

`advertised.listeners`	**广播地址**。客户端连接时，Broker 返回给客户端的地址。**至关重要**。	- 如果客户端和 Broker 不在同一网络（如 Docker、云服务器），**必须**配置此项为客户端可访问的公网 IP 或域名。 - 若不配，默认取 `listeners` 的值，可能导致客户端连接内网 IP 失败。
`log.dirs`	**数据存储目录**。消息日志文件存放路径。	- **强烈建议使用高性能 SSD**。 - 可配置多个目录（逗号分隔）以利用多块磁盘 IO，如 `/data1/kafka,/data2/kafka`。
`num.partitions`	**默认分区数**。创建 Topic 时未指定分区数时的默认值。	建议设置为 3~6（与集群 Broker 数量匹配或略多），以提高并行消费能力。
`auto.create.topics.enable`	**自动创建 Topic**。当生产者向不存在的 Topic 发消息时，是否自动创建。	**生产环境强烈建议设为 ****`false`****。 原因：防止因代码拼写错误创建大量无用 Topic，且自动创建的 Topic 参数（分区/副本）往往不符合生产标准。

5.2.2 副本与可靠性 (Replication & Reliability)

核心作用：保证数据不丢失、服务高可用。

参数	说明	生产环境建议
`default.replication.factor`	**默认副本数**。新建 Topic 时的默认副本数量。	建议 **≥ 3**。允许同时挂掉 2 台机器而不丢失数据。需确保集群 Broker 数量 ≥ 该值。
`min.insync.replicas`	**最小同步副本数**。配合 `acks=all` 使用，表示写入成功至少需要多少个副本确认。	- 若设为 1：只要 Leader 写入成功即返回，Leader 挂掉可能丢数据。 - **推荐设为 2** (配合 3 副本)：保证即使挂掉 1 台，仍有 1 个完整副本存活，且写入时需 2 个节点成功，兼顾安全与可用。
`offsets.topic.replication.factor`	`__consumer_offsets` 内部 Topic 的副本数。	必须 **≥ 2** (最好 3)。若该 Topic 挂了，所有消费者无法提交位移，导致重复消费或消费停滞。

`replica.lag.time.max.ms`	**ISR 踢出阈值**。Follower 落后 Leader 超过此时间，将被移出 ISR 列表。	默认 30s。网络抖动大时可适当调大，但过大会导致故障切换变慢。
`unclean.leader.election.enable`	**非 ISR 选举**。允许不在 ISR 中的副本（数据落后的）成为 Leader。	**必须设为 `**false**` (默认)。 若设为 `true` ：当所有 ISR 挂掉，强制让数据落后的副本上位，会导致数据丢失**。生产环境严禁开启。
`controller.quorum.voters`	**控制器投票节点**。仅用于 **KRaft 模式** (无 ZK)。	ZK 模式下留空。若迁移到 KRaft 模式，需配置 Controller 节点的 ID 和地址。

5.2.3 日志保留策略 (Log Retention)

核心作用：控制磁盘空间使用，清理旧数据。

参数	说明	生产环境建议
`log.retention.hours`	**按时间保留**。日志文件保留的小时数。	默认 168h (7天)。根据业务需求调整，如合规要求保留 30 天则设为 720。
`log.retention.bytes`	**按大小保留**。每个 Partition 最大保留的字节数。	注释该配置则默认为 -1 (不限制)。建议设置上限（如 50GB），防止单 Topic 写爆磁盘。
`log.segment.bytes`	**段文件大小**。日志切分的大小阈值。	默认 1GB。太大导致清理粒度粗，太小导致文件句柄过多。一般保持默认即可。
`log.retention.check.interval.ms`	**检查间隔**。后台线程检查是否要删除旧日志的频率。	默认 5 分钟。无需频繁修改。

5.2.4 性能调优 (Performance Tuning)

核心作用：平衡吞吐量与延迟，适配硬件资源。

参数	说明	生产环境建议
`num.network.threads`	**网络线程数**。处理 Socket 读写、编解码的线程数。	默认 8。高并发场景（如万级 TPS）可调至更大或与 CPU 核数匹配。

num.io.threads	**IO 线程数**。处理磁盘读写请求的线程数。	默认 16。建议设置为 **磁盘数量的 2 倍** 或 CPU 核数，如 **16**。
socket.send/receive.buffer.bytes	**Socket 缓冲区**。发送/接收缓冲区大小。	默认 1024KB。高吞吐场景可适当调大（如 2MB），依赖 OS 内核参数配合。
log.flush.interval.messages/ms	**刷盘策略**。多少条消息或多少毫秒强制刷盘到磁盘。	**默认值极大** (即不主动强制刷盘)。 **最佳实践**：依赖 OS 的 Page Cache 异步刷盘，性能最高。仅在极度追求数据零丢失且能接受性能大幅下降时才调小此值。

5.2.5 ZooKeeper 连接 (ZooKeeper Connection)

核心作用：连接元数据管理中心（仅限 ZK 模式）。

参数	说明	生产环境建议
zookeeper.connect	**ZK 地址列表**。格式 `host:port`，多节点用逗号分隔。	填写所有 ZK 节点地址。支持 chroot 路径。
zookeeper.session.timeout.ms	**会话超时**。Broker 与 ZK 心跳丢失多久判定为下线。	默认 18s-30s。调小可加快故障发现，但网络抖动易导致误判重平衡；调大则故障恢复慢。
zookeeper.chroot.path	**根路径**。将 Kafka 元数据存储在 ZK 的某个子路径下。	**多集群共用 ZK 时必配**。例如 `/kafka-cluster-01`，避免不同 Kafka 集群元数据冲突。

5.2.6 安全 (Security)

核心作用：认证与加密（注释部分为示例）。

参数	说明	生产环境建议
security.inter.broker.protocol	Broker 间通信协议。	生产环境建议启用 `SASL_SSL` 或至少 `SASL_PLAINTEXT`，防止内部窃听或伪造节点。
sasl.enabled.mechanisms	认证机制。	常用 `PLAIN` (简单账号密码), `SCRAM-SHA-256/512`

		(更安全), GSSAPI (Kerberos)。
listener.name...jaas.config	JAAS 配置。	定义用户名密码。建议将敏感信息移至独立的 JAAS 文件，而非直接写在 properties 中。

5.2.7 其他 (Others)

参数	说明	生产环境建议
delete.topic.enable	删除 Topic 开关。	**建议 ****true**。方便运维清理测试数据或废弃业务线数据。若为 false，删除命令仅标记无效，不释放磁盘。
JMX_PORT	监控端口。	必须配置。用于 Prometheus/JConsole 采集指标。需在启动脚本中通过 -D 参数设置，并注意防火墙放行。
replica.fetch.wait.max.ms	Follower 拉取等待时间。	默认 500ms。适当调大可减少 Follower 对 Leader 的空轮询压力，但会增加同步延迟。一般保持默认。

5.3 ZK 配置说明

5.3.1 基础配置 (Basic Timing)

核心作用：定义心跳节奏和集群容错时间窗口，直接影响故障检测速度。

参数	说明	生产环境建议
tickTime	心跳基准时间 (ms)。ZK 内部所有时间计算的基础单位。	默认 **2000ms **(2秒)。 - 设太小：网络抖动易导致误判节点下线。 - 设太大：故障检测慢。 建议保持 2000，除非网络极差可调至 3000-4000。
initLimit	初始化同步时限。Follower 启动时从 Leader 同步数据的最大 tickTime 倍数。	默认 10 (即 20秒)。 若数据量大或磁盘慢，需调大 (如 20)，否则 Follower 启动会超时失败。
syncLimit	心跳同步时限。Follower 与 Leader 心跳响应的最大 tickTime 倍数。	默认 5 (即 10秒)。 超过此时间未响应，Leader 认为该 Follower 已死，将其踢出集群。网络不稳定时可适当调大至 8-10。

5.3.2 数据存储 (Data Storage)

核心作用：控制数据落盘策略，直接影响 IO 性能和磁盘寿命。

参数	说明	生产环境建议
`dataDir`	**快照数据目录**。存储内存数据库的快照 (snapshot) 和 `myid` 文件。	**必须配置在高性能 SSD 上**。不要与事务日志 (`dataLogDir`) 混用，避免 IO 争抢。
`dataLogDir`	**事务日志目录**。存储写入操作的事务日志 (transaction log)。	**关键性能点**： 1. **务必独立挂载磁盘** (与 `dataDir` 物理隔离)。 2. ZK 对顺序写要求高，独立的 SSD 能极大提升吞吐量。
`preAllocSize`	**预分配文件大小** (KB)。事务日志文件预分配的大小。	默认 65536 (64MB)。 配置中设为 131072 (128MB)。**作用**：减少文件系统碎片和频繁的文件扩展系统调用。大文件有利于顺序写性能。
`snapCount`	**快照触发频率**。每处理多少事务生成一次快照。	默认 100,000。 配置中设为 200,000。 **作用**：减少快照生成频率，降低磁盘 IO 压力。但会导致重启恢复时间变长（因为要重放更多日志）。生产环境可适当调大。
`fsync.warningthresholdms`	**刷盘警告阈值** (ms)。当 `fsync` 耗时超过此值时记录警告日志。	用于监控磁盘健康状况。若频繁出现警告，说明磁盘 IO 瓶颈严重，需更换硬件或优化负载。

5.3.3 连接信息 (Connection Info)

核心作用：定义客户端接入点和并发控制。

参数	说明	生产环境建议
`clientPort`	**客户端监听端口**。Kafka Broker 和 业务客户端 连接 ZK 的端口。	默认 2181。防火墙需开放此端口。
`maxClientCnxns`	**单 IP 最大连接数**。限制单个 IP 地址能建立的最大连接数。	默认 60。 配置中设为 0 (表示无限制)。 **风险**：若某台机器异常发起大量连接，可能耗尽 ZK 资源。**生产环境建议设置合理上限** (如 200-500)，防止 DDoS 或代码死循环。

5.3.4 集群配置 (Cluster Config)

核心作用：定义集群拓扑和节点角色。

参数	说明	生产环境建议
`server.x=A:B:C:D;E`	**集群节点定义**。 `x` : MyID (对应 `myid` 文件)。 `A` : IP。 `B` : 2888 (Follower 与 Leader 通信)。 `C` : 3888 (选举通信)。 `D` : 角色 (participant/observer)。 `E` : 客户端端口 (可选，新版支持)。	- **奇数节点**：集群节点数必须为奇数 (3, 5, 7)，以容忍 `(N-1)/2` 台故障。 - Observer：若需增加读性能而不增加投票负担，可将部分节点设为 `observer` 。
`electionPortBindRetry`	**选举端口绑定重试次数**。	默认 3。若端口被占用，重试几次后退出。一般无需修改。

5.3.5 自动清理 (Auto Purge)

核心作用：防止磁盘被历史快照和日志填满。

参数	说明	生产环境建议
`autopurge.snapRetainCount`	**保留快照数量**。保留最近的 N 个快照文件。	配置中设为 3。 **建议**：至少保留 3 个，以防最新快照损坏时有旧版本可回滚。
`autopurge.purgeInterval`	**清理间隔** (小时)。每隔多久执行一次清理任务。	配置中设为 6 (小时)。 **建议**：根据数据量调整。高写入场景可设为 1-2 小时，低负载可设为 12-24 小时。

5.3.6 TCP 网络优化 (Network Tuning)

核心作用：应对高并发连接和会话管理。

参数	说明	生产环境建议
`globalOutstandingLimit`	**全局待处理请求上限**。防止请求堆积导致 OOM。	默认 1000。 配置中设为 5000。 **注意**：调大可提高吞吐，但若后端处理慢，可能导致内存暴涨。需配合监控观察内存使用率。
`minSessionTimeout`	**最小会话超时** (ms)。客户端请求的 timeout 低于此值将被强制设为此值。	默认 2 * tickTime。 设太小会导致网络微抖动就断连。建议 ≥ 4000ms。

`maxSessionTimeout`	**最大会话超时** (ms)。	默认 20 * tickTime。 配置中设为 60000 (60秒)。允许长连接客户端设置更长的超时时间，减少重连频率。

5.3.7 监控和管理 (Admin & 4lw)

核心作用：运维监控与安全控制。

参数	说明	生产环境建议
`admin.enableServer`	**是否开启 AdminServer** (HTTP 接口)。	默认 true (新版)。 配置中设为 `false` 。 **安全建议**：若不需要 HTTP 监控接口，**建议关闭**以减少攻击面。可通过 JMX 或 4lw 命令监控。
`4lw.commands.whitelist`	**四字命令白名单**。ZK 通过 `echo cmd	nc ip 2181` 进行监控。

5.3.8 其他配置 (Others)

参数	说明	生产环境建议
`skipACL`	**跳过 ACL 检查**。	默认 no。 若设为 `yes` ，将忽略所有权限检查，大幅提升性能，但**失去安全性**。仅在完全可信的内网且追求极致性能时使用。
`leaderServers`	**Leader 是否处理读请求**。	配置中 `no` (可能是旧版本参数或特定发行版)。 标准 ZK 中 Leader 默认处理所有写请求和部分读请求。若设为不参与服务，可能影响读吞吐。通常保持默认。
`forceSync`	**是否强制每次事务都 fsync**。	默认 `no` (使用 OS 缓冲)。 若设为 `yes` ，数据最安全但性能急剧下降。**生产环境务必保持 ****`no`**** 。
`sslQuorum` / `portUnification`	**SSL 加密与端口统一**。	若需启用 SSL 加密集群通信，需设为 `true` 并配置证书。明文内网通信可保持 `false` 。

5.4 常见命令

5.4.1 Topic 管理

5.4.1.1 创建 Topic

创建一个名为 `test-topic` 的 Topic，包含 3 个分区，3 个副本。

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

- **注意**：`replication-factor` 不能超过集群中 Broker 的数量。

5.4.1.2 查看 Topic 列表

```
# 列出所有 Topic
kafka/bin/kafka-topics.sh --list --bootstrap-server
192.168.1.10:9092

# 过滤特定 Topic（支持正则）
kafka/bin/kafka-topics.sh --list --bootstrap-server
192.168.1.10:9092 | grep "test"
```

5.4.1.3 查看 Topic 详情 (分区、副本分布)

这是排查数据倾斜和副本状态最重要的命令。

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

输出解读：

- `Leader`：当前负责读写的副本 ID。
- `Replicas`：该分区所有副本所在的 Broker ID 列表。

- `Isr` (In-Sync Replicas): 当前与 Leader 保持同步的副本列表。**如果 Isr 数量少于 Replicas，说明有副本落后或宕机。**

5.4.1.4 增加分区数 (只能增加，不能减少)

将 `test-topic` 的分区数从 3 增加到 6。

```
kafka/bin/kafka-topics.sh --alter \
  --bootstrap-server 192.168.1.10:9092 \
  --topic test-topic \
  --partitions 6
```

5.4.1.5 删除 Topic

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

- **前提**：配置文件 `server.properties` 中 `delete.topic.enable=true`。

5.4.2 生产与消费 (数据测试)

5.4.2.1 启动控制台生产者 (Producer)

向 `test-topic` 发送消息。输入一行按回车即发送一条消息。

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

- **高级用法**：指定 Key 发送（用于测试分区策略）

```
kafka/bin/kafka-console-producer.sh --bootstrap-server
192.168.1.10:9092 --topic test-topic --property "parse.key=true" --
property "key.separator=:"
# 输入格式：key:value  （例如 user1:login）
```

5.4.2.2 启动控制台消费者 (Consumer)

从 `test-topic` 消费消息。

- **消费最新数据**:

```
kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server 192.168.1.10:9092 \
  --topic test-topic
```

- **从头开始消费 (历史数据)**:

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

- **显示 Key 和 分区信息**:

```
kafka/bin/kafka-console-consumer.sh \
  --bootstrap-server 192.168.1.10:9092 \
  --topic test-topic \
  --from-beginning \
  --property print.key=true \
  --property partition=true \
  --property print.timestamp=true
```

- **限制消费数量** (测试用):

```
kafka/bin/kafka-console-consumer.sh --bootstrap-server
192.168.1.10:9092 --topic test-topic --max-messages 10
```

5.4.3 消费者组管理 (Consumer Groups)

5.4.3.1 查看消费者组列表

```
kafka/bin/kafka-consumer-groups.sh \
  --bootstrap-server 192.168.1.10:9092 \
  --list
```

5.4.3.2 查看组内详情 (延迟、Offset)

运维最常用命令，用于查看消费积压 (Lag)。

```
kafka/bin/kafka-consumer-groups.sh \
  --bootstrap-server 192.168.1.10:9092 \
  --group my-consumer-group \
  --describe
```

输出关键列:

- `CURRENT-OFFSET`：当前已提交的位置。
- `LOG-END-OFFSET`：分区最新的消息位置。
- `LAG`：**积压量** = LOG-END-OFFSET - CURRENT-OFFSET。如果 LAG 持续增大，说明消费速度跟不上生产速度。

5.4.3.3 重置 Offset (慎用)

将消费者组的 Offset 重置到最早、最新或指定时间。

- **重置到最早**:

```
kafka/bin/kafka-consumer-groups.sh --bootstrap-server
192.168.1.10:9092 \
  --group my-consumer-group \
  --topic test-topic \
  --reset-offsets --to-earliest \
  --execute
```

- **重置到最新** (相当于忽略历史积压):

```
kafka/bin/kafka-consumer-groups.sh --bootstrap-server
192.168.1.10:9092 \
```

```
  --group my-consumer-group \
  --topic test-topic \
  --reset-offsets --to-latest \
  --execute
```

- **注意**：执行重置时，消费者组必须处于**非活跃状态**（即没有正在运行的消费者实例）。

5.4.4 集群与元数据管理

5.4.4.1 查看集群 Broker 信息

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

5.4.4.2 首选副本选举 (Preferred Leader Election)

如果因为故障导致 Leader 分布不均匀（例如所有 Leader 都在 Node 1），可以触发重新选举，让 Leader 回到 `Replicas` 列表中的第一个节点（通常是理想分布）。

```
kafka/bin/kafka-leader-election.sh --bootstrap-server
192.168.1.10:9092 \
  --election-type preferred \
  --all-topic-partitions
```

5.4.4.3 查看集群配置

```
kafka/bin/kafka-configs.sh --bootstrap-server 192.168.1.10:9092 \
  --entity-type brokers \
  --entity-name 0 \
  --describe
```

6 常见问题

6.0.1 常见错误排查

- ZK 启动失败，日志报 `Connection refused` 或 ** `Session expired`
  - 检查 `myid` 文件是否存在且内容正确。
  - 检查 `zookeeper.properties` 中 `server.x` 的 IP 是否可达，端口 2888/3888 是否被防火墙拦截。
- Kafka 启动失败，日志报 ** `Broker ID 0 is already registered`
  - 检查是否有残留的旧 ZK 数据。如果是新部署，清空 ZK 的 `dataDir` 和 Kafka 的 `log.dirs` 后重试。
  - 确认各节点 `broker.id` 不重复。
- ISR 集合不全 (Isr: 0,1 而不是 0,1,2)**
  - 说明某个 Follower 同步太慢或网络不通。检查该节点的日志和网络带宽。
- Kafka 启动失败，日志报 Failed to bind to port 9092
  - 检查 9092 端口是否被其他进程占用（使用 netstat -tulpn | grep 9092 或 lsof -i:9092 命令）。
  - 检查 server.properties 中 listeners 配置的端口是否与实际要使用的端口一致，避免配置冲突。
  - 确认防火墙未拦截 9092 端口，或已在防火墙规则中放行该端口。
- Kafka 生产者发送消息失败，报 Leader not available
  - 检查对应的 Topic 是否存在，若不存在需先创建（kafka-topics.sh --create 命令）。
  - 检查 Kafka 集群是否正常启动，Broker 节点是否全部在线。
  - 检查 Topic 的副本配置是否合理，若副本数大于可用 Broker 数，会导致 Leader 无法选举。
- Kafka 消费者消费失败，报 No offset found for partition
  - 若为新消费者组，首次消费无偏移量属于正常情况，可配置 auto.offset.reset 为 earliest 或 latest。
  - 若为旧消费者组，检查是否手动删除过 ZK/ Kafka 中的偏移量数据，或消费者组长时间未消费导致偏移量过期（需调整 offsets.retention.minutes 配置）。
- ZK 集群启动后，部分节点无法加入集群，日志报 Cannot open channel to x at election address
  - 检查所有 ZK 节点的 zoo.cfg 中 server.x 配置的 IP / 端口是否一致，无拼写错误。
  - 检查各节点之间的时间是否同步（时间差超过 2000ms 会导致选举失败），可通过 ntpdate 同步时间。

    - 确认 dataDir 目录权限为 ZK 运行用户可读写，避免权限不足导致 myid 文件无法读取。

- Kafka 日志报 Not enough replicas，消息无法持久化

    - 检查 Topic 的 min.insync.replicas 配置值是否大于当前可用的同步副本数（ISR）。
    - 若为单 Broker 部署，需将 min.insync.replicas 改为 1（默认 1，若手动修改过需调整）。
    - 检查 Follower 节点是否正常同步，排查网络延迟、磁盘 IO 过高导致同步慢的问题。

- Kafka 启动后，JVM 内存溢出（OOM），进程直接崩溃

    - 检查 kafka-server-start.sh 中的 JVM 配置（KAFKA_HEAP_OPTS），默认堆内存可能过大 / 过小。
    - 单机测试环境可调整为 -Xmx512M -Xms512M，生产环境根据服务器内存配置（建议不超过物理内存的 50%）。
    - 检查是否开启了过多的 Topic / 分区，导致内存占用过高，需合理规划分区数。

- 连接 Kafka 时报 SSL handshake failed

    - 若开启了 SSL 认证，检查客户端 / 服务端的 SSL 证书路径、密码配置是否正确。
    - 确认证书未过期，且服务端 listeners 配置为 SSL:// 而非 PLAINTEXT://。
    - 检查客户端是否正确配置了 SSL 相关参数（如 security.protocol=SSL）

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com