Apusic 金蝶天燕

生成时间：2026-07-13

快速使用手册

金蝶Apusic分布式消息队列V2.0.6_for_rocketmq

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 前言

本文档为金蝶Apusic分布式消息队列for RocketMQ（Apusic Distributed Message Queue for RocketMQ，简称：ADMQ for RocketMQ）的快速使用手册，提供产品介绍，安装部署产品的基本步骤及验证方式。

1.1 适用对象

本文档适用于IT信息化业务负责人、研发经理、软件项目经理、软件架构师、运维工程师。

1.2 相关文档

了解更多ADMQ for RocketMQ产品相关的信息，请参阅以下ADMQ for RocketMQ产品手册文档集：

序号	手册文档	说明
1	金蝶Apusic分布式消息队列for RocketMQ 快速使用手册	简单介绍了如何快速上手使用ADMQ for RocketMQ 。
2	金蝶Apusic分布式消息队列for RocketMQ 安装手册	详细介绍如何在各操作系统上安装ADMQ for RocketMQ，以及ADMQ for RocketMQ服务启停等操作。
3	金蝶Apusic分布式消息队列for RocketMQ 消息引擎用户手册	详细介绍 ADMQ for RocketMQ 消息引擎相关功能的使用、配置、管理及配套工具的使用方法。
4	金蝶Apusic分布式消息队列for RocketMQ 管控台用户手册	详细介绍ADMQ for RocketMQ管控台相关功能的使用和操作说明。
5	金蝶Apusic分布式消息队列for RocketMQ 开发手册	详细介绍基于各开发语言进行ADMQ for RocketMQ客户端应用开发的说明。
6	金蝶Apusic分布式消息队列for RocketMQ 迁移手册	详细介绍从RocketMQ迁移到ADMQ for RocketMQ的说明。
7	金蝶Apusic分布式消息队列for RocketMQ 运维手册	详细介绍ADMQ for RocketMQ的监控、运维、安全加固等运维说明。
8	金蝶Apusic分布式消息队列for RocketMQ 性能优化手册	详细介绍ADMQ for RocketMQ性能调优的说明。

1.3 技术支持

ADMQ for RocketMQ产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com

- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

1. 您的姓名
2. 公司信息与联系方式
3. 操作系统及其版本
4. 产品版本号
5. 出现异常及错误的日志、截图等详细信息

2 快速部署

2.1 环境准备

组件	最低要求
CPU	4 核
内存	16 GB
磁盘	200 GB SSD
操作系统	CentOS 7+ / Ubuntu 18.04+

2.2 一键部署

```
mkdir -p /apusic
cd /apusic
# 1.解压安装包
#tar zxvf ADMQ-V2.0.6-RocketMQ-构建日期.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260624.tar.gz -C /apusic/
cd /apusic
# 建议nameserver broker组件文件夹结构分开
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

# 2. 启动 NameServer
bin/rocketmq start nameserver

# 3. 启动 Broker
bin/rocketmq start broker

# 4. 验证状态
bin/mqadmin clusterList -n localhost:9876
```

2.3 管控台部署

```
# 1. 解压管控台
tar zxvf admq-manager-V2.0.6.tar.gz -C /apusic/
cd /apusic/admq-manager-V2.0.6

# 2. 启动管控台
bin/admq-service start
```

3 快速验证

3.1 访问管控台

```
地址：http://<管控台IP>:12305;https://<管控台IP>:12306
默认账号：admq / 11111111
```

3.2 创建测试资源

1. 创建主题
    - 进入「资源管理」-「主题管理」页面
    - 点击「新建」，输入主题名称（如：test-topic）
    - 配置读写队列数（如：3）
    - 确认创建
2. 创建订阅组
    - 进入「资源管理」-「订阅组管理」页面
    - 点击「新建」，输入订阅组名称（如：group）
    - 配置订阅组信息
    - 确认创建
3. 查看消息发送
    - 进入「消息查询」页面
    - 选择主题：test-topic
    - 可查看该 Topic 的消息

3.3 发送和接收消息

3.3.1 使用命令行工具

```
# 发送消息
bin/mqadmin sendMessage -n localhost:9876 -t test-topic -p "Hello ADMQ for RocketMQ!"

# 查看消息
bin/mqadmin queryMsgByKey -n localhost:9876 -t test-topic -k test-key
```

3.3.2 使用管控台发送消息

1. 进入「资源管理」-「Topic 管理」页面
2. 选择 test-topic，点击「发送消息」
3. 填写消息内容、Tag、Key 等信息
4. 点击发送

3.3.3 使用开源SDK实现消息收发（Java SDK为例）

3.3.3.1 概览

目前RocketMQ兼容开源特性，根据底层通信协议的差异主要支持两个系列的客户端SDK，分别是Remoting协议和gRPC协议 关于 Remoting 协议 SDK 和 gRPC 协议 SDK 的对比参考如下：

对比项	Remoting 协议SDK	gRPC 协议SDK
多语言支持	Java为主，其他语言为第三方仓库实现	Java/C/C++/.NET/Go/Rust
接口范围	Producer PushConsumer PullConsumer LitePullConsumer Admin	Producer PushConsumer(仅Java) SimpleConsumer
兼容版本	兼容4.x、5.x版本服务端	仅支持5.x 版本服务端

3.3.3.2 生产消息

创建消息生产者

```
// 实例化消息生产者Producer
DefaultMQProducer producer = new DefaultMQProducer(
      namespace,
      groupName,
      new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)) //
ACL权限
);
// 设置NameServer的地址
producer.setNamesrvAddr(nameserver);
// 启动Producer实例
producer.start();
```

发送消息 发送消息由多种方式，同步发送、异步发送、单向发送等。

- 同步消息

```
 for (int i = 0; i < 10; i++) {
 // 创建消息实例，设置topic和消息内容
 Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " +
i).getBytes(RemotingHelper.DEFAULT_CHARSET));
 // 发送消息
 SendResult sendResult = producer.send(msg);
 System.out.printf("%s%n", sendResult);
 }
```

- 异步消息

```
// 设置发送失败后不重试
producer.setRetryTimesWhenSendAsyncFailed(0);
// 设置发送消息的数量
int messageCount = 10;
final CountDownLatch countDownLatch = new CountDownLatch(messageCount);
for (int i = 0; i < messageCount; i++) {
try {
        final int index = i;
        // 创建消息实体，设置topic和消息内容
        Message msg = new Message(topic_name, "TAG", ("Hello rocketMq " +
index).getBytes(RemotingHelper.DEFAULT_CHARSET));
        producer.send(msg, new SendCallback() {
                @Override
                public void onSuccess(SendResult sendResult) {
                        // 消息发送成功逻辑
                        countDownLatch.countDown();
                        System.out.printf("%-10d OK %s %n", index,
sendResult.getMsgId());
                }

                @Override
                public void onException(Throwable e) {
                        // 消息发送失败逻辑
                        countDownLatch.countDown();
                        System.out.printf("%-10d Exception %s %n", index,
e);
                        e.printStackTrace();
                }
```

```
        });
 } catch (Exception e) {
        e.printStackTrace();
 }
}
countDownLatch.await(5, TimeUnit.SECONDS);
```

- 单向发送

```
for (int i = 0; i < 10; i++) {
 // 创建消息实例，设置topic和消息内容
 Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " +
i).getBytes(RemotingHelper.DEFAULT_CHARSET));
 // 发送单向消息
 producer.sendOneway(msg);
}
```

3.3.3.3 消费消息（Remoting协议为例）

创建消费者 支持 push 和 pull 两种消费模式

- push消费者

```
 // 实例化消费者
DefaultMQPushConsumer pushConsumer = new DefaultMQPushConsumer(
 namespace,
 groupName,
 new AclClientRPCHook(new SessionCredentials(accessKey, secretKey))); //ACL
权限
// 设置NameServer的地址
pushConsumer.setNamesrvAddr(nameserver);
```

- pull消费者

```
// 实例化消费者
DefaultLitePullConsumer pullConsumer = new DefaultLitePullConsumer(
namespace,
 groupName,
 new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)));
// 设置NameServer的地址
pullConsumer.setNamesrvAddr(nameserver);
```

```
// 设置从第一个偏移量开始消费
pullConsumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET);
```

订阅消息 根据消费模式不同，订阅方式也有所区别。

- push订阅

```
// 订阅topic
pushConsumer.subscribe(topic_name, "*");
// 注册回调实现类来处理从broker拉取回来的消息
pushConsumer.registerMessageListener((MessageListenerConcurrently) (msgs,
context) -> {
 // 消息处理逻辑
 System.out.printf("%s Receive New Messages: %s %n",
Thread.currentThread().getName(), msgs);
 // 标记该消息已经被成功消费，根据消费情况，返回处理状态
 return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
});
// 启动消费者实例
pushConsumer.start();
```

- pull订阅

```
 // 订阅topic
pullConsumer.subscribe(topic_name, "*");
// 启动消费者实例
pullConsumer.start();
try {
 System.out.printf("Consumer Started.%n");
 while (true) {
         // 拉取消息
         List<MessageExt> messageExts = pullConsumer.poll();
         System.out.printf("%s%n", messageExts);
 }
} finally {
      pullConsumer.shutdown();
}
```

4 客户端快速接入

4.1 Java 客户端示例

```
import org.apache.rocketmq.client.producer.DefaultMQProducer;
import org.apache.rocketmq.client.consumer.DefaultMQPushConsumer;
import
org.apache.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext;
import
org.apache.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus;
import
org.apache.rocketmq.client.consumer.listener.MessageListenerConcurrently;
import org.apache.rocketmq.common.message.Message;
import org.apache.rocketmq.common.message.MessageExt;
import java.util.List;

public class QuickStart {
    public static void main(String[] args) throws Exception {
        // 生产者
        DefaultMQProducer producer = new DefaultMQProducer("test-group");
        producer.setNamesrvAddr("localhost:9876");
        producer.start();

        // 发送消息
        Message msg = new Message("test-topic", "tag-a", "test-key",
            "Hello ADMQ for RocketMQ!".getBytes("UTF-8"));
        producer.send(msg);
        System.out.println("Message sent");
        producer.shutdown();

        // 消费者
        DefaultMQPushConsumer consumer = new DefaultMQPushConsumer("test-
group");
        consumer.setNamesrvAddr("localhost:9876");
        consumer.subscribe("test-topic", "*");
        consumer.registerMessageListener(new MessageListenerConcurrently()
{
```

```
            @Override
            public ConsumeConcurrentlyStatus
consumeMessage(List<MessageExt> msgs,
                    ConsumeConcurrentlyContext context) {
                for (MessageExt msg : msgs) {
                    System.out.println("Received: " + new
String(msg.getBody()));
                }
                return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
            }
        });
        consumer.start();
        System.out.println("Consumer started");
    }
}
```

4.2 Go 客户端示例

```
package main

import (
    "context"
    "fmt"
    "github.com/apache/rocketmq-clients/golang/v2"
)

func main() {
    // 生产者
    producer, err := golang.NewProducer(
        golang.WithEndpoints("localhost:9876"),
        golang.WithConsumerGroup("test-group"),
    )
    if err != nil {
        panic(err)
    }

    err = producer.Start()
```

```
    if err != nil {
        panic(err)
    }
    defer producer.Shutdown()

    msg := golang.NewMessage("test-topic", []byte("Hello ADMQ for
RocketMQ!"))
    resp, err := producer.Send(context.TODO(), msg)
    if err != nil {
        panic(err)
    }
    fmt.Printf("Message sent: %s\n", resp[0].MsgId)
}
```

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