



快速入門

金蝶Apusic分散式訊息佇列V2.0.6_for_rocketmq

版權所有 © 深圳市金蝶天燕雲計算股份有限公司2026。保留所有權利。

版權聲明

本文件所涉及的軟件著作權、版權等知識產權已依法進行了註冊，由金蝶天燕雲計算股份有限公司合法擁有。受《中華人民共和國著作權法》《計算機軟件保護條例》《知識產權保護條例》和相關國際版權條約、法律、法規以及其它知識產權法律和條約的保護。未經授權許可，不得非法使用。

免責聲明

本文件包含的版權資訊由金蝶天燕雲計算股份有限公司合法擁有，受法律的保護，金蝶天燕雲計算股份有限公司對本文件可能涉及到的非金蝶天燕雲計算股份有限公司的資訊不承擔任何責任。在法律允許的範圍內，您可以查閱並僅能夠在《中華人民共和國著作權法》規定的合法範圍內複製和打印本文件。任何單位和個人未經金蝶天燕雲計算股份有限公司書面授權許可，不得使用、修改、再發佈本文件的任何部分和內容，否則將被視為侵權，金蝶天燕雲計算股份有限公司有依法追究其責任的權利。

本文件如有更新，不另行通知。對本文件中的問題您可向金蝶天燕雲計算股份有限公司告知或查詢。未經本公司明確授予的任何權利均予保留。

商標聲明

 是深圳市金蝶天燕雲計算股份有限公司向中華人民共和國國家商標局申請註冊的註冊商標，註冊商標專用權由金蝶天燕合法擁有，受法律保護。未經金蝶天燕的書面許可，任何單位及個人不得以任何方式或理由對該商標的任何部分進行使用、複製、修改、傳播、抄錄或與其它產品捆綁使用銷售。凡侵犯金蝶天燕商標權的，金蝶天燕將依法追究其法律責任。本文件提及的其他所有商標或註冊商標，由各自的所有人擁有。

目錄

- 1 前言
 - 1.1 適用對象
 - 1.2 相關文檔
 - 1.3 技術支援
- 2 快速部署
 - 2.1 環境準備
 - 2.2 一鍵部署
 - 2.3 管控台部署
- 3 快速驗證
 - 3.1 訪問管控台
 - 3.2 建立測試資源
 - 3.3 傳送和接收訊息
 - 3.3.1 使用命令列工具
 - 3.3.2 使用管控台傳送訊息
 - 3.3.3 使用開源SDK實現訊息收發（Java SDK為例）
 - 3.3.3.1 概覽
 - 3.3.3.2 生產訊息
 - 3.3.3.3 消費訊息（Remoting協定為例）
- 4 客戶端快速接入
 - 4.1 Java 客戶端示例
 - 4.2 Go 客戶端示例

1 前言

本文檔為金蝶Apusic分散式訊息佇列for RocketMQ（Apusic Distributed Message Queue for RocketMQ，簡稱：ADMQ for RocketMQ）的快速使用手冊，提供產品介紹，安裝部署產品的基本步驟及驗證方式。

1.1 適用對象

本文檔適用於IT資訊化業務負責人、研發經理、軟體專案經理、軟體架構師、運維工程師。

1.2 相關文檔

了解更多ADMQ for RocketMQ產品相關的資訊，請參閱以下ADMQ for RocketMQ產品手冊文件集：

序號	手冊文檔	說明
1	金蝶Apusic分散式訊息佇列for RocketMQ 快速使用手冊	簡單介紹了如何快速上手使用ADMQ for RocketMQ。
2	金蝶Apusic分散式訊息佇列for RocketMQ 安裝手冊	詳細介紹如何在各作業系統上安裝ADMQ for RocketMQ，以及ADMQ for RocketMQ服務啟停等操作。
3	金蝶Apusic分散式訊息佇列for RocketMQ 訊息引擎使用者手冊	詳細介紹 ADMQ for RocketMQ 訊息引擎相關功能的使用、配置、管理及配套工具的使用方法。
4	金蝶Apusic分散式訊息佇列for RocketMQ 管控台使用者手冊	詳細介紹ADMQ for RocketMQ管控台相關功能的使用和操作說明。
5	金蝶Apusic分散式訊息佇列for RocketMQ 開發手冊	詳細介紹基於各開發語言進行ADMQ for RocketMQ客戶端應用開發的說明。
6	金蝶Apusic分散式訊息佇列for RocketMQ 遷移手冊	詳細介紹從RocketMQ遷移到ADMQ for RocketMQ的說明。
7	金蝶Apusic分散式訊息佇列for RocketMQ 運維手冊	詳細介紹ADMQ for RocketMQ的監控、運維、安全加固等運維說明。
8	金蝶Apusic分散式訊息佇列for RocketMQ 效能最佳化手冊	詳細介紹ADMQ for RocketMQ效能調優的說明。

1.3 技術支援

ADMQ for RocketMQ產品提供全面的技術支援服務，您可以透過以下方式獲得技術支援：

- 網址：www.apusic.com
- 電話：400-855-5800
- 郵箱：support@apusic.com

- 金蝶雲社群: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

您在取得技術支援時，請提供如下資訊：

1. 您的姓名
2. 公司資訊與聯絡方式
3. 作業系統及其版本
4. 產品版本號
5. 出現異常及錯誤的日誌、截圖等詳細資訊

2 快速部署

2.1 環境準備

元件	最低要求
CPU	4 核
記憶體	16 GB
磁碟	200 GB SSD
作業系統	CentOS 7+ / Ubuntu 18.04+

2.2 一鍵部署

```
mkdir -p /apusic
cd /apusic
# 1. 解壓安裝包
#tar zxvf ADMQ-V2.0.6-RocketMQ-構建日期.tar.gz -C /apusic/
tar zxvf ADMQ-V2.0.6.534-RocketMQ-20260624.tar.gz -C /apusic/
cd /apusic
# 建議nameserver broker元件資料夾結構分開
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. 啟動 NameServer
bin/rocketmq start nameserver

# 3. 啟動 Broker
bin/rocketmq start broker

# 4. 驗證狀態
bin/mqadmin clusterList -n localhost:9876
```

2.3 管控台部署

1. 解壓管控台

```
tar zxvf admq-manager-V2.0.6.tar.gz -C /apusic/
```

```
cd /apusic/admq-manager-V2.0.6
```

2. 啟動管控台

```
bin/admq-service start
```

3 快速驗證

3.1 訪問管控台

地址: `http://<管控台IP>:12305`; `https://<管控台IP>:12306`
預設帳號: `admq / 11111111`

3.2 建立測試資源

1. 建立主題

- 進入「資源管理」-「主題管理」頁面
- 點選「新建」，輸入主題名稱（如：test-topic）
- 配置讀寫佇列數（如：3）
- 確認建立

2. 建立訂閱組

- 進入「資源管理」-「訂閱組管理」頁面
- 點選「新建」，輸入訂閱組名稱（如：group）
- 配置訂閱組資訊
- 確認建立

3. 檢視訊息傳送

- 進入「訊息查詢」頁面
- 選擇主題：test-topic
- 可檢視該 Topic 的訊息

3.3 傳送和接收訊息

3.3.1 使用命令列工具

傳送訊息

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

檢視訊息

```
bin/mqadmin queryMsgByKey -n localhost:9876 -t test-topic -k test-key
```

3.3.2 使用管控台傳送訊息

1. 進入「資源管理」-「Topic 管理」頁面
2. 選擇 test-topic，點選「傳送訊息」
3. 填寫訊息內容、Tag、Key 等資訊
4. 點選傳送

3.3.3 使用開源SDK實現訊息收發（Java SDK為例）

3.3.3.1 概覽

目前RocketMQ相容開源特性，根據底層通訊協議的差異主要支援兩個系列的客戶端SDK，分別是Remoting協定和gRPC協定 關於 Remoting 協定 SDK 和 gRPC 協定 SDK 的對比參考如下：

對比項	Remoting 協定SDK	gRPC 協定SDK
多語言支援	Java為主，其他語言為第三方倉庫實現	Java/C/C++/.NET/Go/Rust
介面範圍	Producer PushConsumer PullConsumer LitePullConsumer Admin	Producer PushConsumer(僅Java) SimpleConsumer
相容版本	相容4.x、5.x版本服務端	僅支援5.x 版本服務端

3.3.3.2 生產訊息

建立訊息生產者

```
// 實例化訊息生產者Producer
DefaultMQProducer producer = new DefaultMQProducer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)) //
    ACL權限
);
// 設定NameServer的地址
producer.setNamesrvAddr(nameserver);
// 啟動Producer實例
producer.start();
```

傳送訊息 傳送訊息由多種方式，同步傳送、非同步傳送、單向傳送等。

- 同步訊息

```

for (int i = 0; i < 10; i++) {
    // 建立訊息實例，設定topic和訊息內容
    Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " +
i).getBytes(RemotingHelper.DEFAULT_CHARSET));
    // 傳送訊息
    SendResult sendResult = producer.send(msg);
    System.out.printf("%s\n", sendResult);
}

```

- 非同步訊息

```

// 設定傳送失敗後不重試
producer.setRetryTimesWhenSendAsyncFailed(0);
// 設定傳送訊息的數量
int messageCount = 10;
final CountDownLatch countDownLatch = new CountDownLatch(messageCount);
for (int i = 0; i < messageCount; i++) {
    try {
        final int index = i;
        // 建立訊息實體，設定topic和訊息內容
        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) {
                // 訊息傳送成功邏輯
                countDownLatch.countDown();
                System.out.printf("%-10d OK %s %n", index,
sendResult.getMsgId());
            }

            @Override
            public void onException(Throwable e) {
                // 訊息傳送失敗邏輯
                countDownLatch.countDown();
                System.out.printf("%-10d Exception %s %n", index,
e);
                e.printStackTrace();
            }
        });
    }
}

```

```

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

```

- 單向傳送

```

for (int i = 0; i < 10; i++) {
    // 建立訊息實例, 設定topic和訊息內容
    Message msg = new Message(topic_name, "TAG", ("Hello RocketMQ " +
i).getBytes(RemotingHelper.DEFAULT_CHARSET));
    // 傳送單向訊息
    producer.sendOneway(msg);
}

```

3.3.3.3 消費訊息 (Remoting協定為例)

建立消費者 支援 push 和 pull 兩種消費模式

- push消費者

```

// 實例化消費者
DefaultMQPushConsumer pushConsumer = new DefaultMQPushConsumer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey))); //ACL
權限
// 設定NameServer的地址
pushConsumer.setNamesrvAddr(nameserver);

```

- pull消費者

```

// 實例化消費者
DefaultLitePullConsumer pullConsumer = new DefaultLitePullConsumer(
    namespace,
    groupName,
    new AclClientRPCHook(new SessionCredentials(accessKey, secretKey)));
// 設定NameServer的地址
pullConsumer.setNamesrvAddr(nameserver);

```

```
// 設定從第一個偏移量開始消費
```

```
pullConsumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET);
```

訂閱訊息 根據消費模式不同，訂閱方式也有所區別。

- push訂閱

```
// 訂閱topic
```

```
pushConsumer.subscribe(topic_name, "*");
```

```
// 註冊回撥實現類來處理從broker拉取回來的訊息
```

```
pushConsumer.registerMessageListener((MessageListenerConcurrently) (msgs, context) -> {
```

```
    // 訊息處理邏輯
```

```
    System.out.printf("%s Receive New Messages: %s %n",
```

```
Thread.currentThread().getName(), msgs);
```

```
    // 標記該訊息已經被成功消費，根據消費情況，返回處理狀態
```

```
    return ConsumeConcurrentlyStatus.CONSUME_SUCCESS;
```

```
});
```

```
// 啟動消費者實例
```

```
pushConsumer.start();
```

- pull訂閱

```
// 訂閱topic
```

```
pullConsumer.subscribe(topic_name, "*");
```

```
// 啟動消費者實例
```

```
pullConsumer.start();
```

```
try {
```

```
    System.out.printf("Consumer Started.%n");
```

```
    while (true) {
```

```
        // 拉取訊息
```

```
        List<MessageExt> messageExts = pullConsumer.poll();
```

```
        System.out.printf("%s%n", messageExts);
```

```
    }
```

```
} finally {
```

```
    pullConsumer.shutdown();
```

```
}
```

4 客戶端快速接入

4.1 Java 客戶端示例

```
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 {
        // 生產者
        DefaultMQProducer producer = new DefaultMQProducer("test-group");
        producer.setNamesrvAddr("localhost:9876");
        producer.start();

        // 傳送訊息
        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();

        // 消費者
        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 客戶端示例

```

package main

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

func main() {
    // 生產者
    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)
}
```

全國統一服務熱線
4008-555-800

金蝶天燕雲計算股份有限公司（簡稱“金蝶天燕”）成立於2000年，前身為“金蝶中間件公司”，是金蝶集團旗下基礎軟件平台提供商，雲計算國家標準制定企業，國家信創產業核心軟件企業。金蝶天燕是國家863重點研發計劃與核高基重大專項承接企業，也是“兩網一站四庫十二金”國家重點工程的基礎平台提供商，產品廣泛應用於政府、軍工、金融、能源等關鍵行業，累計服務客戶總數超過10萬家。



雲計算國家標準制定企業
金蝶集團旗下基礎軟件企業
資訊技術應用創新核心企業
官網：www.apusic.com

