ALIBABA CLOUD

阿里云

消息队列RabbitMQ版

最佳实践

文档版本：20220524

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.开源RabbitMQ迁移上云

1.1. 开源RabbitMQ迁移上云概述

本文介绍将RabbitMQ集群迁移到消息队列RabbitMQ版实例的优势、原理和流程。

迁移优势

将RabbitMQ集群迁移到消息队列RabbitMQ版实例的优势，请参见产品优势。

迁移原理

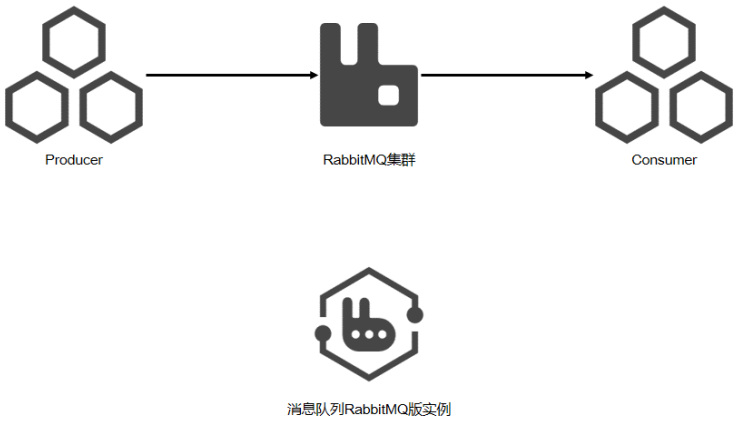
对于消息队列来说，如果要实现集群迁移，只需消费完旧集群的消息即可。由于Producer和Consumer都是集群化的，您可以通过一台一台操作的方式实现上层业务无感知。

迁移流程（双读模式）

通过双读模式将RabbitMQ集群迁移到消息队列RabbitMQ版实例的流程如下：

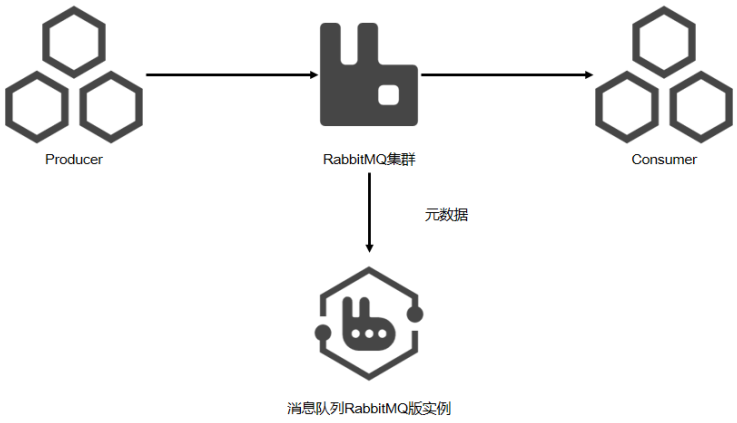
1. 创建消息队列RabbitMQ版实例。

   更多信息，请参见创建实例。

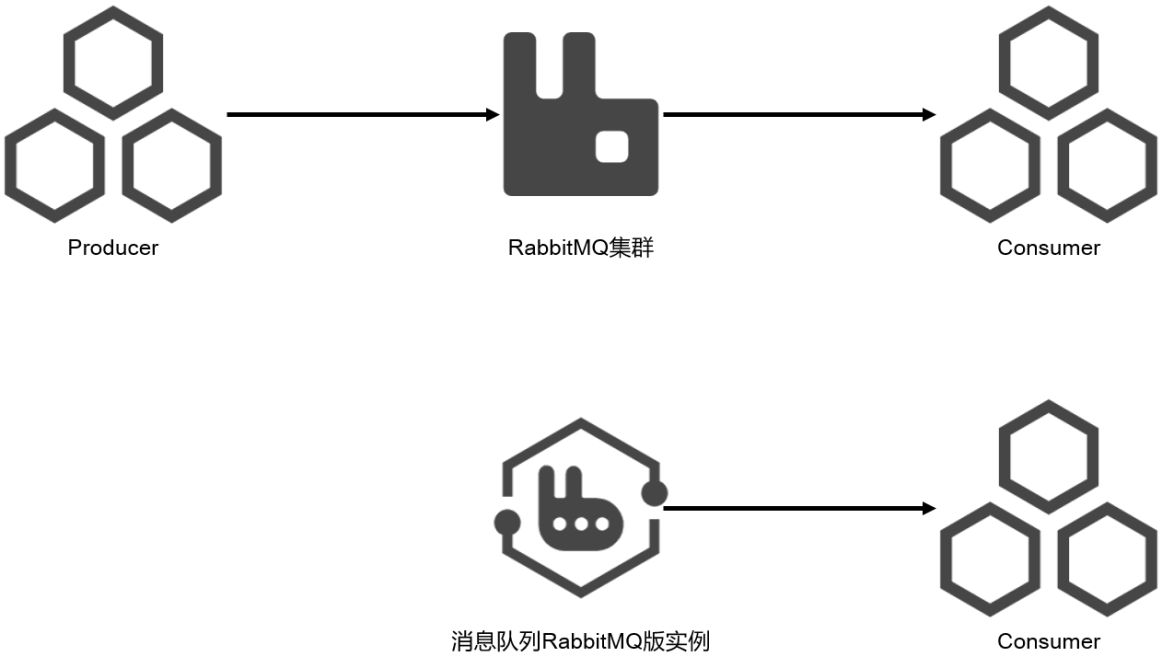


2. 迁移RabbitMQ集群的元数据到消息队列RabbitMQ版实例。

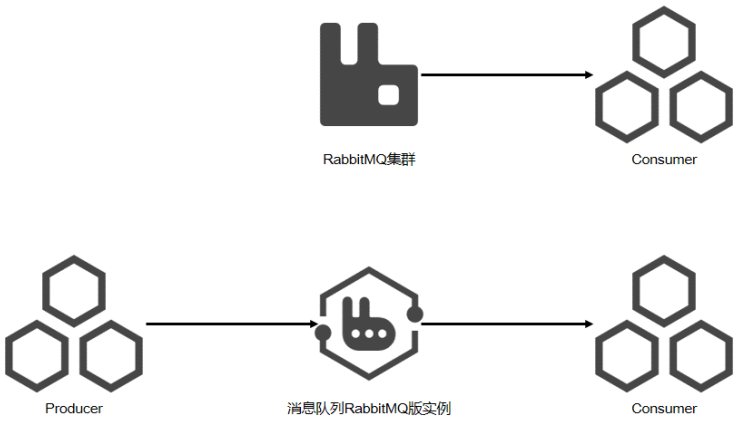
   更多信息，请参见迁移元数据上云。



3. 为消息队列Rabbit MQ版实例开启新的Consumer，准备消费消息队列Rabbit MQ版实例的消息。



4. 为消息队列Rabbit MQ版实例开启新的Producer，下线旧的Producer，并使旧的Consumer继续消费Rabbit MQ集群的消息。



5. 待RabbitMQ集群的消息全部被旧的Consumer消费后，下线旧的Consumer和RabbitMQ集群。

1.2. 开源RabbitMQ元数据迁移上云

本文说明如何将开源RabbitMQ集群元数据迁移到阿里云消息队列RabbitMQ版实例或者实例中的某个Vhost。

前提条件

您已完成以下操作：

- 导出需要迁移的元数据。更多操作，请参见如何导出全部Vhost或者指定Vhost元数据。
- 创建作为元数据迁移目标的消息队列RabbitMQ版实例和Vhost。
  - 元数据包括全部Vhost时，创建消息队列RabbitMQ版实例。更多信息，请参见 创建消息队列RabbitMQ版实例 。
  - 元数据为某个Vhost时，创建消息队列RabbitMQ版实例和Vhost。更多信息，请参见 创建消息队列RabbitMQ版实例、创建Vhost。

背景信息

RabbitMQ集群元数据是指RabbitMQ集群的信息，包括User、Vhost、Queue、Exchange、Binding Key、Permission、Parameter等信息。RabbitMQ集群元数据存储于RabbitMQ集群的内部数据库，在集群的各个节点之间自动复制。集群中的每个节点都有自己的元数据副本。当某个节点的元数据变更时，所有节点的元数据都会同步更新。因此，集群的各个节点的元数据被导出时都是相同的。RabbitMQ集群元数据可以被导出成一份JSON文件，然后被导入另一个RabbitMQ集群，实现RabbitMQ集群元数据备份。

迁移元数据上云是指将开源RabbitMQ集群的元数据迁移到阿里云消息队列RabbitMQ版实例。消息队列RabbitMQ版是阿里云提供的全托管消息队列服务，兼容开源RabbitMQ。您可以将RabbitMQ集群元数据导出，然后导入消息队列RabbitMQ版实例，消息队列RabbitMQ版会根据成功导入的元数据在目标消息队列RabbitMQ版实例中创建对应的Vhost、Queue、Exchange、Binding，实现RabbitMQ集群元数据迁移上云。您可以将全部Vhost信息导入消息队列RabbitMQ版实例，也可以根据需要将某个Vhost信息导入消息队列RabbitMQ版实例中的Vhost。

导入须知

- 如果导入全部Vhost，导入的JSON格式的元数据文件必须包含Vhost信息，即必须包括 `vhosts` 列表，且 `exchanges` 、 `queues` 、 `bindings` 列表中必须包括 `vhost` 。更多信息，请参见sample.json。
- 导入的Vhost、Exchange、Queue、Binding必须符合消息队列RabbitMQ版的数量限制和字符限制。更多信息，请参见使用限制。
- 同一个实例，同一时间只能执行一个任务。当一个任务执行结束后，才能继续为该实例新建任务。

元数据兼容性

由于RabbitMQ和消息队列RabbitMQ版在权限管控机制等方面存在差异，部分RabbitMQ集群元数据不支持导入消息队列RabbitMQ版实例。这些RabbitMQ集群元数据在导入消息队列RabbitMQ版实例时会被自动忽略。消息队列RabbitMQ版实例的RabbitMQ集群元数据兼容性如下：

元数据	描述	兼容
rabbit_version	RabbitMQ集群版本。	否
users	RabbitMQ集群用户。	否 **注意** 如果您的RabbitMQ集群使用了用户，您可以通过访问控制RAM为消息队列RabbitMQ版实例实现对应的用户管理模式。更多信息，请参见 创建RAM用户。
vhosts	RabbitMQ集群的Vhost。	是
permissions	RabbitMQ集群的用户管理Vhost的权限。	否 **注意** 如果您的RabbitMQ集群使用了用户管理Vhost的权限，您可以通过访问控制为消息队列RabbitMQ版实例实现对应的用户管理Vhost的权限。更多信息，请参见RAM权限策略。

元数据	描述	兼容
parameters	RabbitMQ集群的运行时参数。	否
global_parameters	RabbitMQ集群的全局运行时参数。	否
policies	RabbitMQ集群的一类Vhost运行时参数，用于为Vhost下的Exchange和Queue设置可选参数。	否 **注意** 如果您的RabbitMQ集群使用了Vhost域运行时参数配置，您可以在 消息队列RabbitMQ版控制台创建对应的Exchange和Queue时，设置相应的可选参数。Exchange和Queue的可选参数设置后不支持修改。如需修改，您需要删除后重新创建。更多信息，请参见创建Exchange和创建Queue。
queues	RabbitMQ集群的Queue。	是
exchanges	RabbitMQ集群的Exchange。	是
bindings	RabbitMQ集群的Exchange和Queue的绑定关系。	是

迁移元数据

1. 登录消息队列RabbitMQ版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在左侧导航栏，单击**迁移上云**。
4. 在**迁移上云**页面，单击**创建任务**。
5. 在**创建任务**面板，设置相关参数，单击**确定**。

参数	描述	示例值
实例	元数据迁移到实例的实例名称。	amqp-cn-7mz2cjgk****
导入模式	将全部Vhost或者指定Vhost的元数据导入消息队列RabbitMQ版实例或者Vhost。 ○ *ALL*：将开源RabbitMQ全部Vhost元数据导入消息队列RabbitMQ版实例。 ○ *Vhost*：将开源RabbitMQ指定Vhost元数据导入消息队列RabbitMQ版实例的Vhost中。	Vhost

参数	描述	示例值
Vhost	迁移元数据至 消息队列RabbitMQ版实例指定的Vhost。**导入模式**选择*Vhost*时，显示该参数。	test-vhost****
元数据	需要迁移的元数据文件。单击**选择文件**，选择本地的元数据文件后，单击**打开**。 **说明** 元数据文件的大小不超过20 MB。	rabbit_mq-amqp-load-test011122063****

在**迁移上云**页面，显示已创建的迁移任务，并可以查看任务执行状态。

关于导入的元数据的详细信息、任务执行失败的详细信息查看，请参见相关操作。

相关操作

通过创建任务迁移的元数据，支持查看元数据的详细信息和任务执行失败时失败原因。

- 查看导入成功的元数据信息
  i. 在**迁移上云**页面，目标任务所在行**目标实例**列，单击实例名称。
  ii. 在左侧导航栏，单击**Vhost 列表**，目标Vhost所在行**操作**列，单击**详情**，查看Vhost详细信息。

  更多信息，请参见查看Vhost连接详情。
- 查看执行失败的任务详情
  i. 在**迁移上云**页面，目标任务所在行**操作**列，单击**详情**。

  您也可以在**迁移上云**页面，目标实例所在行同步元数据数量列，单击同步数量。
  ii. 在**迁移详情**页面，单击Vhost、Exchange、Queue、Binding对应的页签，查看失败原因。

1.3. 开源RabbitMQ迁移上云FAQ

1.3.1. 如何导出全部Vhost或者指定Vhost元数据

在导出开源RabbitMQ集群元数据时，开源RabbitMQ支持导出全部Vhost或某个指定的Vhost元数据。本文介绍如何通过开源RabbitMQ控制台和HTTP API导出全部Vhost或者指定Vhost元数据。

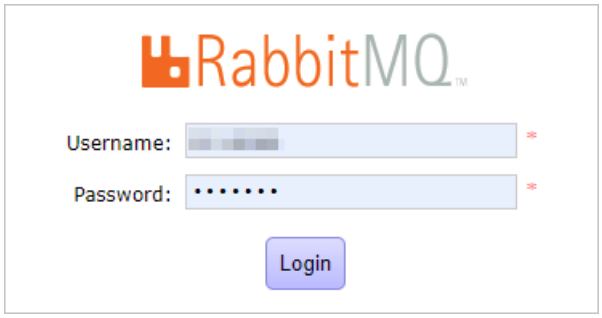
前提条件

开启RabbitMQ管理插件。

开源RabbitMQ控制台导出

1. 在浏览器打开开源RabbitMQ控制台。

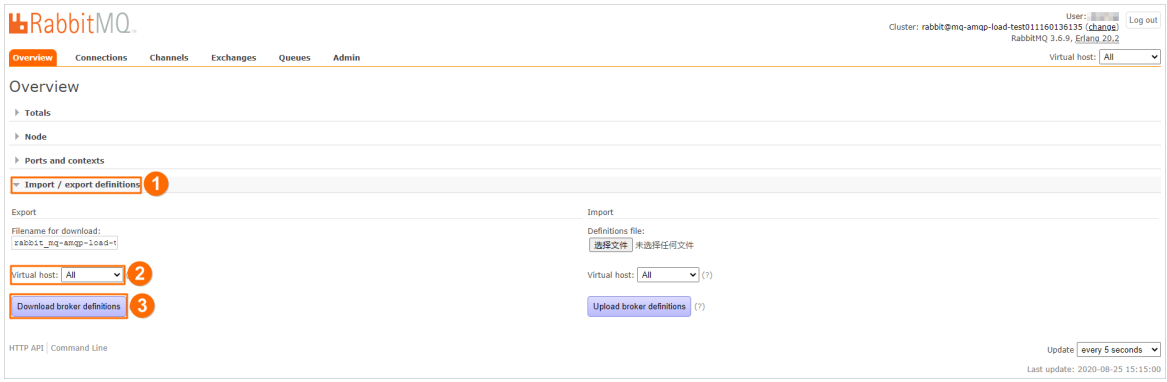
   开源RabbitMQ控制台地址：`http://您的RabbitMQ IP地址:15672/`
2. 在登录页面的Username文本框输入您的用户名，在Password文本框输入您的密码，然后单击Login。



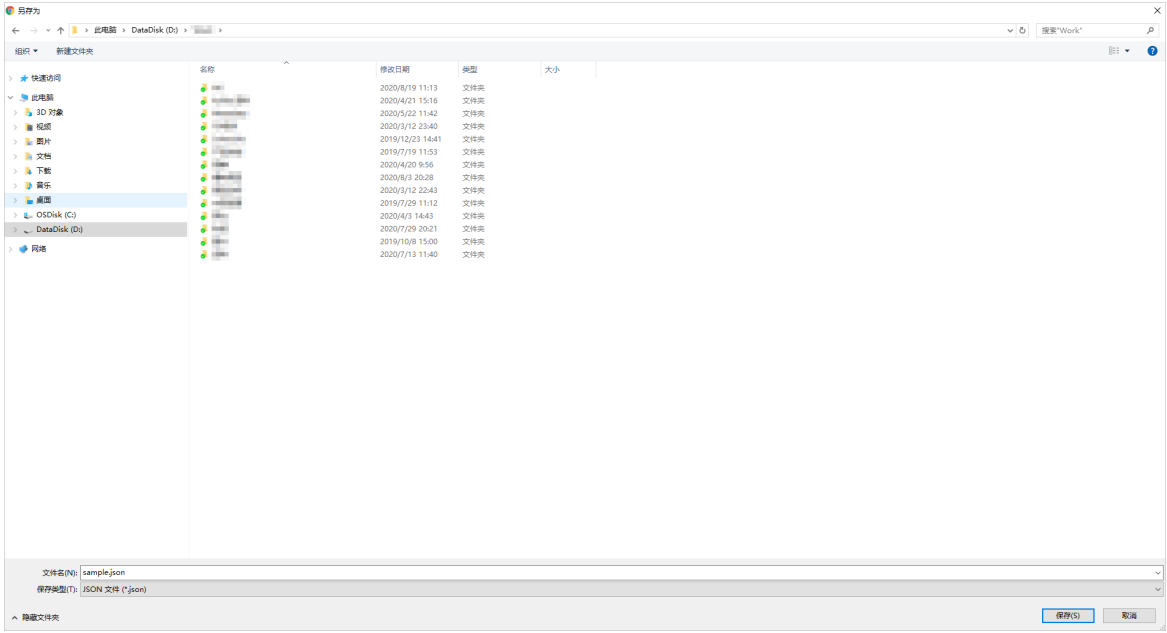
3. 在**Overview**页签下方，单击**Import / export definitions**，从**Export**区域的**Virtual host**列表，选择**All**或者指定的Vhost名称，然后单击**Download broker definitions**。

   Virtual host列表参数说明如下：

   - *All*：表示导出全部Vhost元数据。
   - *Vhost名称*：表示导出指定Vhost元数据。



4. 在**另存为**对话框，选择RabbitMQ集群元数据文件保存路径，然后单击**保存**。



   元数据文件保存路径下显示导出的RabbitMQ集群元数据文件。

开源RabbitMQ HTTP API导出

1. 打开终端。

2. 执行以下命令导出Rabbit MQ集群元数据文件。
   - 导出全部Vhost元数据

     ```
     wget --user <您的RabbitMQ账号> --password <您的RabbitMQ密码> http://<您的RabbitMQ IP地址>:15672/api/definitions -O <您的元数据文件保存路径>
     ```

   - 导出指定Vhost元数据

     ```
     wget --user <您的RabbitMQ账号> --password <您的RabbitMQ密码> http://<您的RabbitMQ IP地址>:15672/api/definitions -O <您的元数据文件保存路径> --vhost <Vhost名称>
     ```

2.消息幂等

如果消息重复消费会影响您的业务处理，请对消息做幂等处理。本文介绍消息幂等的概念、适用场景以及处理方法。

什么是消息幂等

在数学与计算机学中，幂等操作的特点是其任意多次执行所产生的影响均与一次执行的影响相同。在消息领域，幂等是指Consumer重复消费某条消息时，重复消费的结果与消费一次的结果是相同的，并且多次消费并未对业务系统产生任何负面影响。

例如，在支付场景下，Consumer消费扣款消息，对一笔订单执行扣款操作，扣款金额为100元。如果因网络不稳定等原因导致扣款消息重复投递，Consumer重复消费了该扣款消息，但最终的业务结果是只扣款一次，扣费100元，且用户的扣款记录中对应的订单只有一条扣款流水，不会多次扣除费用。那么这次扣款操作是符合要求的，整个消费过程实现了消息幂等。

适用场景

在互联网应用中，尤其在网络不稳定的情况下，消息队列Rabbit MQ版的消息有可能会出现重复。如果消息重复消费会影响您的业务处理，请对消息做幂等处理。消息重复的可能原因如下：

- 发送时消息重复

  当一条消息已被成功发送到服务端并完成持久化，此时出现了网络闪断或者客户端宕机，导致服务端对客户端应答失败。 如果此时Producer意识到消息发送失败并尝试再次发送消息，Consumer后续会收到两条内容相同并且Message ID也相同的消息。

- 投递时消息重复

  消息消费的场景下，消息已投递到Consumer并完成业务处理，当客户端给服务端反馈应答的时候网络闪断。为了保证消息至少被消费一次，消息队列Rabbit MQ版的服务端将在网络恢复后再次尝试投递之前已被处理过的消息，Consumer后续会收到两条内容相同并且Message ID也相同的消息。

- 负载均衡时消息重复（包括但不限于网络抖动、服务端重启以及Consumer应用重启）

  当消息队列Rabbit MQ版的服务端或客户端重启、扩容或缩容时，会触发Rebalance，此时Consumer可能会收到重复消息。

处理方法

以Message ID为幂等键对消息进行幂等处理的步骤如下：

1. 在数据库中创建一张unique key索引为唯一Message ID的表。
2. 在Producer客户端为每条消息设置唯一Message ID。

   设置唯一Message ID的示例代码如下：

   ```
   AMQP.BasicProperties props = new AMQP.BasicProperties.Builder().messageId(UUID.randomUUID().toString()).build();
   channel.basicPublish("${ExchangeName}", "BindingKey", true, props, ("消息发送Body" + i).getBytes(StandardCharsets.UTF_8));
   ```

   了解更多Message ID相关信息，请参见如何设置Message ID。
3. 在Consumer客户端根据唯一Message ID对消息进行幂等处理。

   根据唯一Message ID进行幂等处理的示例代码如下：

```
channel.basicConsume(Producer.QueueName, false, "YourConsumerTag",
    new DefaultConsumer(channel) {
    @Override public void handleDelivery(String consumerTag, Envelope envelope,
                AMQP.BasicProperties properties, byte[] body) throws IOException {
        // 1. 获取业务唯一性索引数据。
        try{
            String messageId = properties.getMessageId();
            // Message ID或者其他作为unique key的信息。
            // 2. 开启数据库事务。
            idempTable.insert(messageId);
            // 3. 对接收到的消息，进行业务逻辑处理。
            // 4. 提交或回滚事务。// 处理成功，则进行ACK，否则不要进行ACK。
            channel.basicAck(envelope.getDeliveryTag(), false);
        }
        catch (数据库主键冲突异常 e){
            // 重复消息，直接确认掉。
            channel.basicAck(envelope.getDeliveryTag(), false);
        }
    }
}
);
```

3.网络异常自动恢复

由于服务端升级、服务端重启、网络抖动等原因，服务端和客户端的网络连接可能会断开。本文介绍如何在客户端设置Connection和Topology自动恢复，使Connection和Topology在网络连接断开后自动恢复，避免断连对您的业务造成影响。

触发原因

触发Connection自动恢复的原因如下：

- Connection的I/O抛出异常。
- Socket读取操作超时。
- 检测到服务端心跳丢失。

恢复方法

> **注意** 4.0.0及以上版本Java客户端默认开启Connection和Topology自动恢复，您无需在代码中设置。

在客户端开启Connection和Topology（Queue、Exchange、Binding、Consumer）自动恢复的方法如下：

- `factory.setAutomaticRecoveryEnabled(boolean)`：用于开启或关闭Connection自动恢复。
- `factory.setNetworkRecoveryInterval(long)`：用于设置重试时间间隔。如果Connection自动恢复异常，设置了Connection自动恢复的客户端将在一段固定时间间隔（默认为5秒）后重试。
- `factory.setTopologyRecoveryEnabled(boolean)`：用于开启Topology自动恢复。Topology包括Queue、Exchange、Binding、Consumer。

示例代码

开启Connection和Topology自动恢复的客户端示例代码如下：

```
ConnectionFactory factory = new ConnectionFactory();
// 设置接入点，在消息队列AMQP版控制台实例详情页面获取。
factory.setHost("xxx.xxx.aliyuncs.com");
// ${instanceId}为实例ID，从阿里云AMQP版控制台实例详情页面获取。
// ${AccessKey}阿里云身份验证，在阿里云管理控制台创建。
// ${SecretKey}阿里云身份验证，在阿里云管理控制台创建。
factory.setCredentialsProvider(new AliyunCredentialsProvider("${AccessKey}", "${SecretKey}"
, "${instanceId}"));
// 设置Vhost名称，请确保已在消息队列AMQP版控制台上创建。
factory.setVirtualHost("${VhostName}");
// 默认端口，非加密端口5672，加密端口5671。
factory.setPort(5672);
// 基于网络环境设置合理的超时时间。
factory.setConnectionTimeout(30 * 1000);
factory.setHandshakeTimeout(30 * 1000);
factory.setShutdownTimeout(0);
// 开启Connection自动恢复。
factory.setAutomaticRecoveryEnabled(true);
// 设置Connection重试时间间隔为10秒。
factory.setNetworkRecoveryInterval(10000);
// 开启Topology自动恢复。
factory.setTopologyRecoveryEnabled(true);
Connection connection = factory.newConnection();
```

恢复限制

Connection自动恢复的限制如下：

- Connection断开需要一定的时间检测。要确保这段时间内发送的消息不丢失，需使用Publisher Confirms实现可靠发送。
- Channel异常导致Connection断开时，不会触发Connection自动恢复。Channel异常通常为应用级别的问题，需要使用方自行处理。
- Connection自动恢复不会使Channel也自动恢复。

4.使用AMQProxy解决PHP等客户端Connection复用问题

AMQProxy是一款开源AMQP代理服务，具备复用AMQP Connection的能力。您可以通过该代理服务使原本只能使用短连接的客户端（例如PHP客户端）使用长连接，从而减少网络资源消耗和消息队列RabbitMQ版资源消耗。

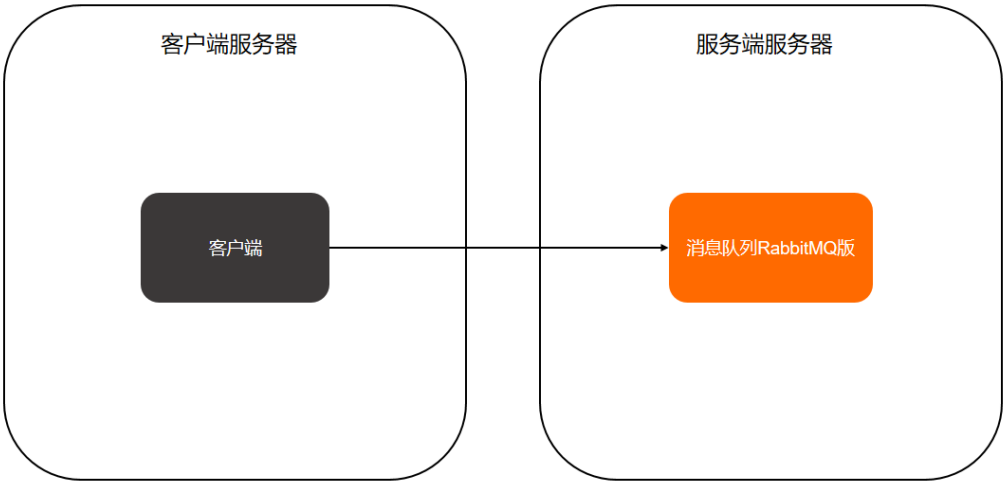
前提条件

如果您要使用SSL连接AMQProxy和消息队列RabbitMQ版，请确保您的客户端服务器已安装OpenSSL。更多信息，请参见Downloads。

> 注意　VPC环境下不支持使用SSL连接消息队列RabbitMQ版。

背景信息

部分语言的客户端，例如PHP客户端，无法使用长连接，会频繁地开启或关闭Connection，消耗大量的网络资源和消息队列RabbitMQ版资源，从而对消息队列RabbitMQ版造成巨大压力。

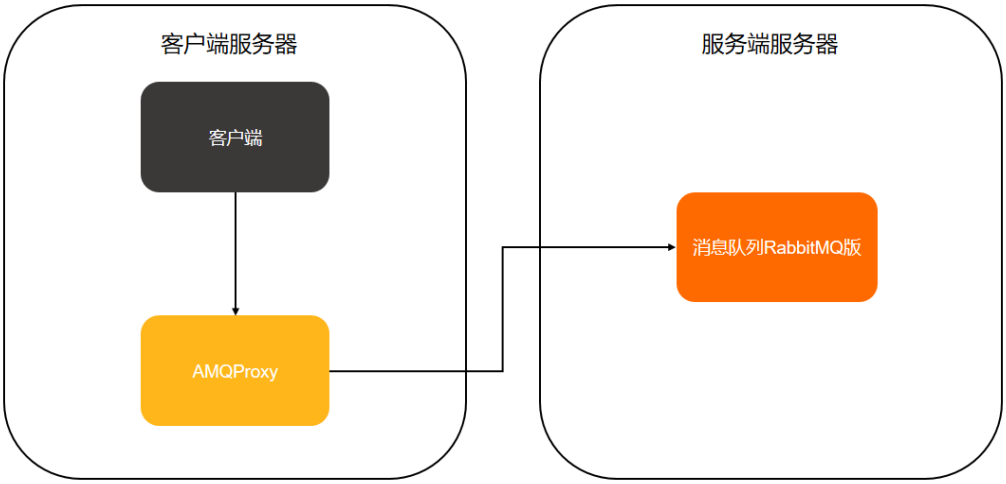


AMQProxy

AMQProxy是Cloud AMQP提供的开源AMQP代理服务。客户端可以通过该代理服务与消息队列RabbitMQ版保持长连接。当您在客户端服务器部署AMQProxy后，客户端和消息队列RabbitMQ版之间的请求都会先发送到AMQProxy，然后由AMQProxy转发到对方。

AMQProxy处理客户端发起的Connection相关请求的逻辑如下：

- 如果客户端发送开启Connection的请求，AMQProxy将根据用户名、密码、Vhost查找当前是否有合适的Connection可以复用，如果有就复用该Connection，如果没有就由AMQProxy代替客户端和消息队列RabbitMQ版开启Connection。
- 如果客户端发送关闭Connection的请求，AMQProxy会直接应答OK，但并不会关闭与消息队列RabbitMQ版的Connection，当该客户端下次再请求开启Connection时，AMQProxy会直接使用该Connection。



更多AMQProxy相关信息，请参见AMQProxy。

部署AMQProxy

1. 执行以下命令下载AMQProxy压缩包。

```
wget https://github.com/cloudamqp/amqproxy/releases/download/v0.4.4/amqproxy-0.4.4-1.linux-x86_64-static.tar.gz
```

> **说明** 本文以AMQProxy 0.4.4为例。更多AMQProxy版本，请参见Releases。

2. 执行以下命令解压AMQProxy压缩包。

```
tar -xzvf amqproxy-0.4.4-1.linux-x86_64-static.tar.gz
```

3. 执行以下命令进入AMQProxy文件夹。

```
cd amqproxy/
```

4. 执行以下命令启动AMQProxy。

```
./amqproxy -l LISTEN_ADDRESS -p LISTEN_PORT AMQP_URL
```

参数	描述
LISTEN_ADDRESS	AMQProxy IP地址。由于是在客户端服务器部署AMQProxy，因此您可以直接使用本机地址*127.0.0.1*。
LISTEN_PORT	AMQProxy监听端口。客户端请求通过该端口发送到AMQProxy。该端口可以为任何可用的端口，例如5673。

参数	描述
AMQP_URL	消息队列RabbitMQ版实例的URL。格式为`{amqp|amqps}://{endpoint}`。 ○ amqp：AMQP协议。不使用SSL连接时使用。 ○ amqps：AMQP/SSL协议。使用SSL连接时使用。 ○ endpoint：消息队列RabbitMQ版实例的接入点。您可以在消息队列RabbitMQ版控制台的**实例详情**页面查看。更多信息，请参见查看实例详情。

示例命令如下：

```
./amqproxy -l 127.0.0.1 -p 5673 amqps://188XXX420.mq-amqp.cn-hangzhou-a.aliyuncs.com
```

返回示例如下：

```
Proxy upstream: 188XXX420.mq-amqp.cn-hangzhou-a.aliyuncs.com:5671 TLS
Proxy listening on 127.0.0.1:5673
0 clients                0 upstreams
```

参数	描述
clients	客户端和AMQProxy的Connection数量。
upstreams	AMQProxy和消息队列RabbitMQ版实例的Connection数量。

5. 在客户端代码中将Host和端口修改为AMQProxy IP地址和监听端口。

```
factory.setHost("127.0.0.1");
factory.setPort(5673);
```

5.基于日志服务最佳实践

5.1. 基于日志服务的日志说明

消息队列RabbitMQ版的日志管理功能将消息队列RabbitMQ版实例的消息操作日志推送到日志服务，通过SLS分析语句快速查询并统计TPS流量图表。本文介绍SLS中日志字段和请求方法。

字段说明

查询的日志字段说明如下表所示。

日志字段说明

参数	描述
Action	操作对应的请求方法。取值和描述请参见请求方法。
Queue	订阅或者消息对应的Queue。描述如下： • Action为*PushMessage*、*BasicGet*和*DeleteMessage*时，为订阅的Queue。 • Action为*BasicReject*时，为被拒绝的消息对应的Queue。 • Action为*BasicNack*时，为Nack消息对应的Queue。
Property	消息的属性。取值和描述如下： • *consumerTag*：用于标识Queue的订阅者。 • *deliveryTag*：服务端标识某个Channel上的唯一消息。 **说明** 仅Action为*PushMessage*、*BasicGet*、*DeleteMessage*或*SendDlqMessage*时，记录该字段。
ResourceName	资源名称。 **说明** Action为*ConnectionOpen*、*ConnectionClose*、*ChannelOpen*或*ChannelClose*时，不记录该字段。
Vhost	Vhost名称。您可以在消息队列RabbitMQ版控制台的**Vhost列表**页面查看。

参数	描述
ReqUid	账号ID。可以是阿里云账号（主账号）或RAM用户（子账号）。 **说明** Action为*SendDlqMessage*时，不记录该字段。
RemoteAddress	发起该操作的客户端地址。 **说明** Action为*SendDlqMessage*时，不记录该字段。
InstanceId	本次执行的实例ID。
Info	表示当前API调用失败时的报错信息。
ConnectionId	服务端用于唯一标识Connection。 **说明** Action为*SendDlqMessage*时，不记录该字段。
Code	200表示成功调用，其他为异常。关于异常描述，请参见Info字段描述。
ChannelId	客户端生成的Channel ID，用于标识当前Connection下的唯一Channel。 **说明** Action为*ConnectionOpen*和*ConnectionClose*时，为*null*。

请求方法

日志服务Action的请求方法如下表所示。

请求方法

请求方法	说明
ConnectionOpen	开启连接。
ConnectionClose	关闭连接。
ChannelOpen	开启Channel。
ChannelClose	关闭Channel。

请求方法	说明
QueueDeclare	创建Queue。
QueueDelete	删除Queue。
ExchangeDeclare	创建Exchange。
ExchangeDelete	删除Exchange。
ExchangeBind	绑定路由到Exchange。
ExchangeUnBind	解除源Exchange到目标Exchange的绑定。
QueueBind	绑定路由到Queue。
QueueUnbind	解除源Exchange到目标Queue的x0005绑定。
SendMessage	生产者生产消息。
PushMessage	服务端推送消息。
BasicGet	客户端拉取消息。
BasicAck	ACK消息。
BasicConsume	订阅Queue。
BasicReject	Reject消息。
BasicRecover	Recover消息。
BasicNack	Nack消息。
BasicQos	设置Consumer的流控。
QueuePurge	清空Queue中所有消息。
DeleteMessage	客户端调用*BasicAck*，服务端确认消息被成功删除。
SendDlqMessage	发送死信消息。

5.2. 通过原始日志快速分析和定位问题

本文介绍基于消息队列RabbitMQ版当前日志快速查询和分析问题的方法。当您遇到消息不符合预期、消费异常、消息堆积时，通过该方法可以帮助您高效识别出异常、保证业务正常运行。

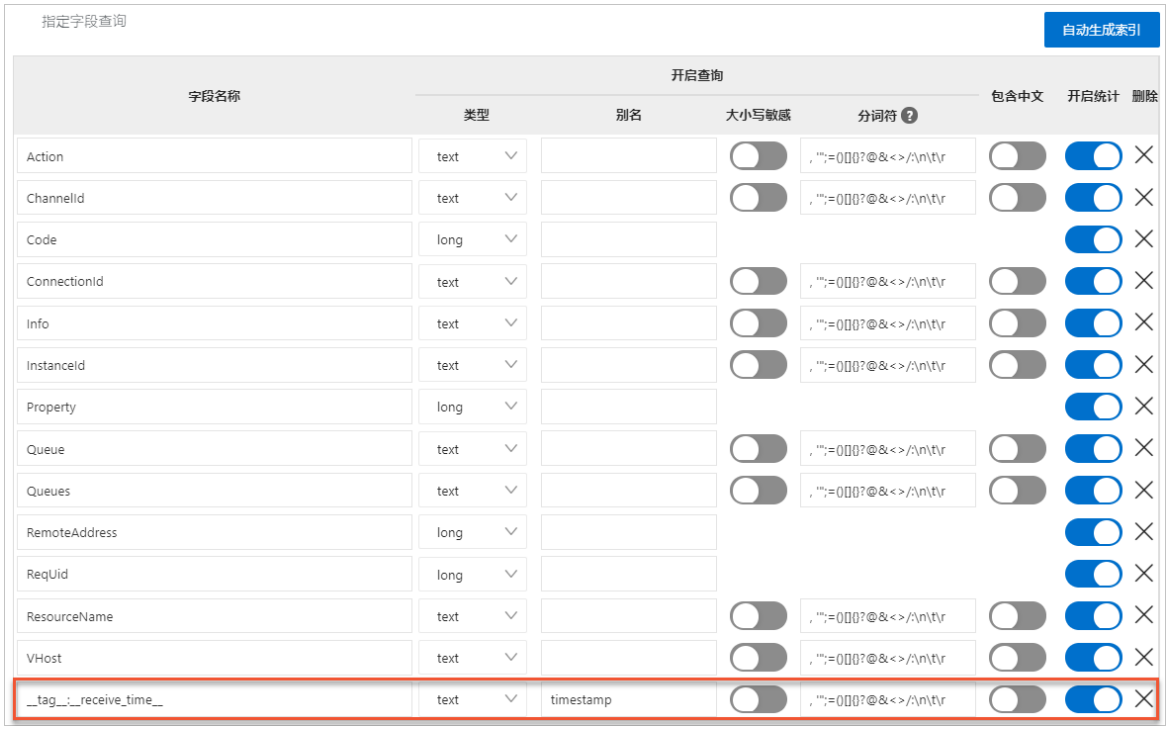
前提条件

推送消息队列RabbitMQ版实例的消息操作日志到日志服务。具体操作，请参见配置消息日志。

操作步骤

1. 登录日志服务控制台，配置日志字段索引。

具体操作，请参见配置索引。除自动生成的普通字段之外，还需手动添加__tag__:__receive_time__字段，并设置别名为*timestamp*，如下图所示。



2. 设置查询的时间段，在搜索框输入SLS查询语句，查询消息日志。

   根据以下消息内容，查询对应的消息日志：

   - 查询消息轨迹
   - 查询Queue的消费情况
   - 死信消息查询

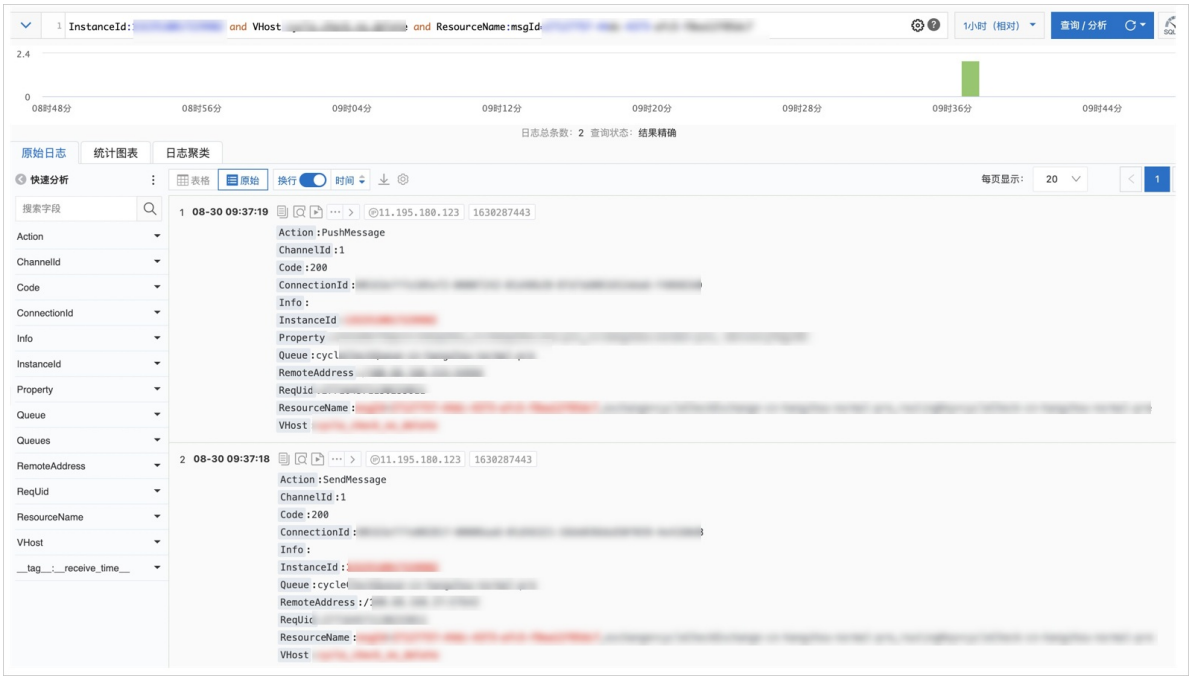
查询消息轨迹

在日志搜索框中输入查询语句，查询消息轨迹。

- 通过Message ID查询

  在搜索框输入SLS查询语句，查询实例ID和Vhost下的msgId的日志。

```
InstanceId:amqp-cn-i7m29o3s**** and VHost:cycle**** and ResourceName:msgId=27127757-44dc-4373-afc5-f8ea12f****
```



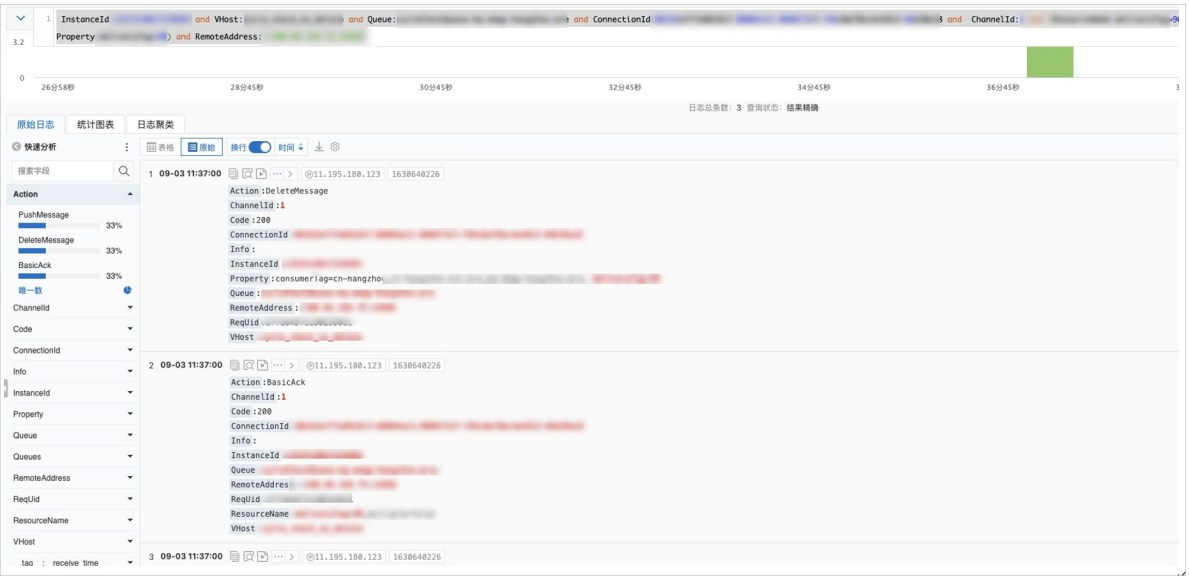
客户端调用 `BasicConsume` 订阅，可以查询到消息的发送和服务端的推送轨迹，`SendMessage` 对应客户端调用 `BasicPublish` 、`PushMessage` 对应服务端推送消息给客户端；如果客户端采用的 `BasicGet` 拉取消息，则Action为 `BasicGet` 而不是 `PushMessage` 。

> **说明**
>
> - 如果日志结果中 `SendMessage` 只有一条，但是 `PushMessage(BasicGet)` 对应多条，可能是客户端没有在首条 `PushMessage(BasicGet)` 之后的60s内调用 `BasicAck` 确认消息，导致服务端以为客户端没有消费成功而重新推送该条消息。
> - 如果日志结果中除了 `SendMessage` 和 `PushMessage` 之外，还存在 `SendDlqMessage` 日志，`SendDlqMessage` 日志表示当前消息转入死信。

- 查询消息是否被消费

```
InstanceId:amqp-cn-i7m29o3s**** and VHost:cycle**** and Queue:cycleCheckQueue**** and ConnectionId:00163efffe08281f-00004e11-0009732f-799c0af9bc4e4913-96b3**** and  ChannelId:1 and (ResourceName:deliveryTag=90 or Property:deliveryTag=90) and RemoteAddress:"/192.168.XX.XX:XXXX"
```

其中，ConnectionId、ChannelId、deliveryTag、RemoteAddress都是通过 `PushMessage` 日志填写。RemoteAddress是客户端的IP地址，ConnectionId是当前Connection的唯一标识，ChannelId是当前Channel的唯一标识，`deliveryTag` 是服务端对当前消息的唯一标识；`DeleteMessage` 日志则是客户端成功消费消息的标记。

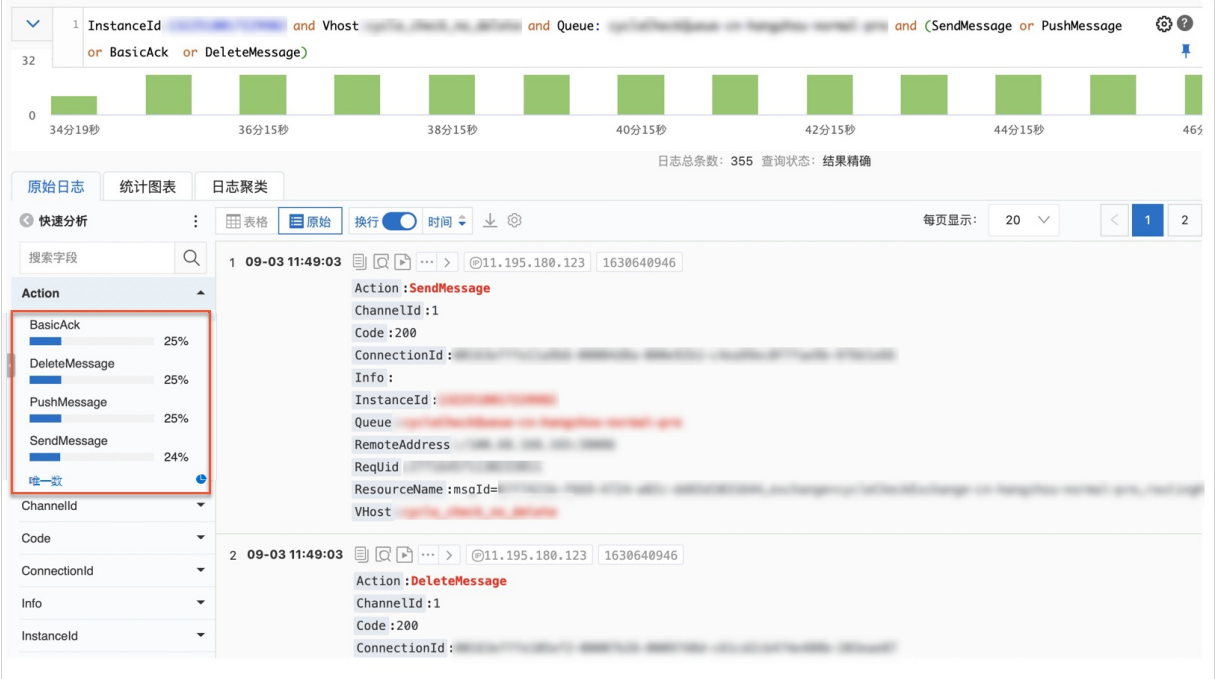


> **说明**
>
> - 如果客户端调用 `BasicConsume` 时设置 `autoAck=false` ，则需要客户端调用 `BasicAck` 服务端才能确认消息被客户端正常消费并删除，而且 `BasicAck` 必须在 `PushMessage` 时间戳之后的60s内调用，超过60s服务端则认为客户端消费失败，并会重新推送该消息到客户端。
> - 客户端已经调用 `BasicAck` ，但是查询消息是否被消费的时候，发现日志中并没有 `DeleteMessage` 日志，说明当前消息并没有成功消费，可以观察 `PushMessage` 和 `BasicAck` 时间差，大于60s则表示当前 `BasicAck` 是无效调用。
> - 如果只查询到 `PushMessage` 和 `DeleteMessage` ，没有查到 `BasicAck` ，可能是 `BasicAck(deliveryTag, multiple=true)` ，一次性Ack了 `deliveryTag` 之前的所有消息。

查询Queue的消费情况

在日志搜索框中输入查询语句，查询 `SendMessage` 、 `PushMessage(BasicGet)` 、 `BasicAck` 、 `DeleteMessage` 。当四者的百分比相同的时候，表示客户端的发送与消费速度基本持平，当前Queue无消息堆积或是有极少消息堆积。

```
InstanceId:amqp-cn-i7m29o3s**** and Vhost:cycle**** and Queue: cycleCheckQueue**** and (SendMessage or PushMessage or BasicAck  or DeleteMessage)
```



如果查询的日志出现以下几种情况，请根据可能的原因分析：

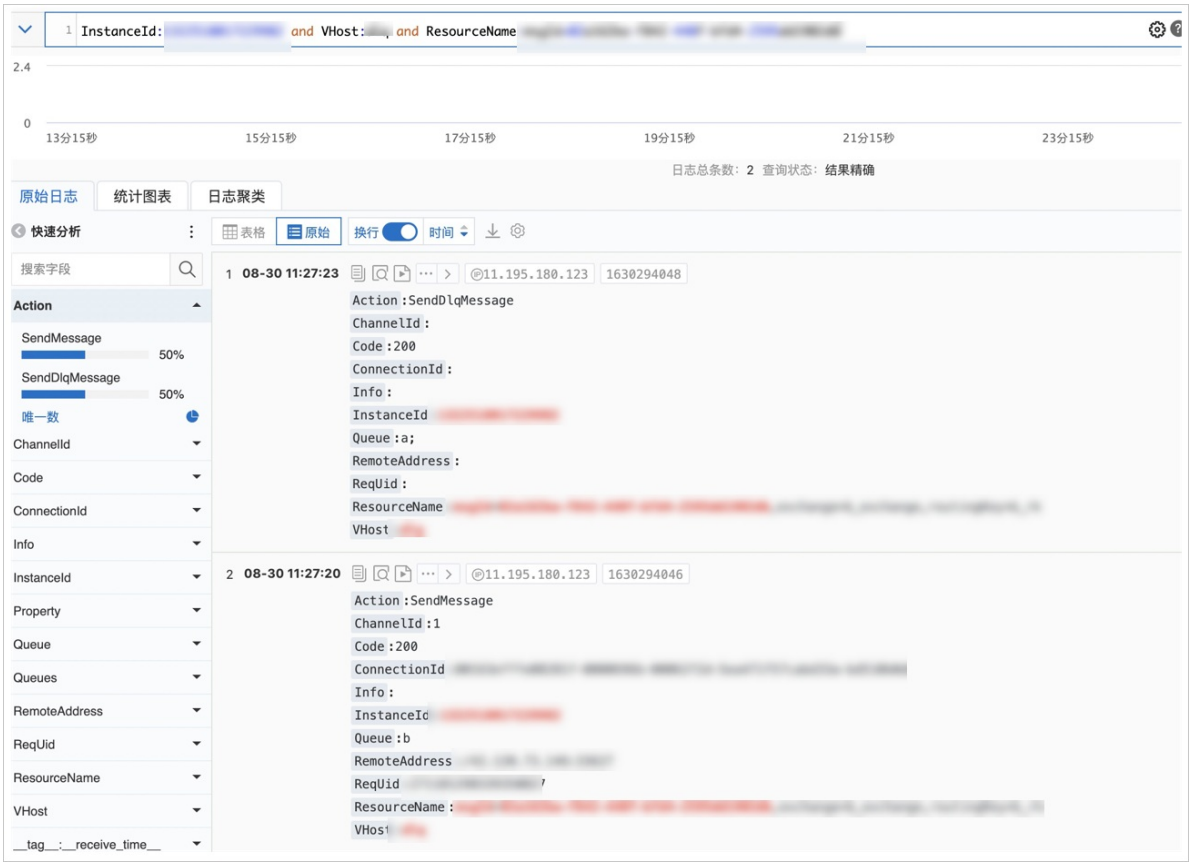
- 日志搜索结果中没有 `DeleteMessage` ，可能是客户端调用 `BasicConsume` 或者是 `BasicGet` 的时候，设置了 `autoAck=true` ，也可能是客户端所有的 `BasicAck` 都是在 `PushMessage` 日志60s之后的无效调用。
- 日志搜索结果中发现 `PushMessage` 的百分比相比 `SendMessage` 较少，可能是客户端的订阅者过少，建议多开Connection并创建新的Consumer。
- 日志搜索结果中发现 `SendMessage` 和 `PushMessage` 的日志百分比相当，但是 `DeleteMessage` 的百分比则相对较少，可能是客户端存在大量 `BasicAck` 都是在 `PushMessage` 日志60s之后的无效调用。
- 日志搜索结果中发现 `SendMessage` 、 `PushMessage` 、 `DeleteMessage` 日志百分比大致相同，但是 `BasicAck` 日志相对较少，考虑是否代码中使用了 `BasicAck(multiple=true)` 。

死信消息查询

日志搜索框输入查询语句，查询对应消息。

- 消息TTL到期进入死信队列

```
InstanceId:amqp-cn-i7m29o3s**** and VHost:dlq**** and ResourceName:msgId=02a162ba-f842-440f-bfd4-2595dd19****
```



`SendMessage` 对应用户调用 `BasicPublish` 发送消息，`SendDlqMessage` 对应该消息ttl过期后发送到对应的死信队列中。

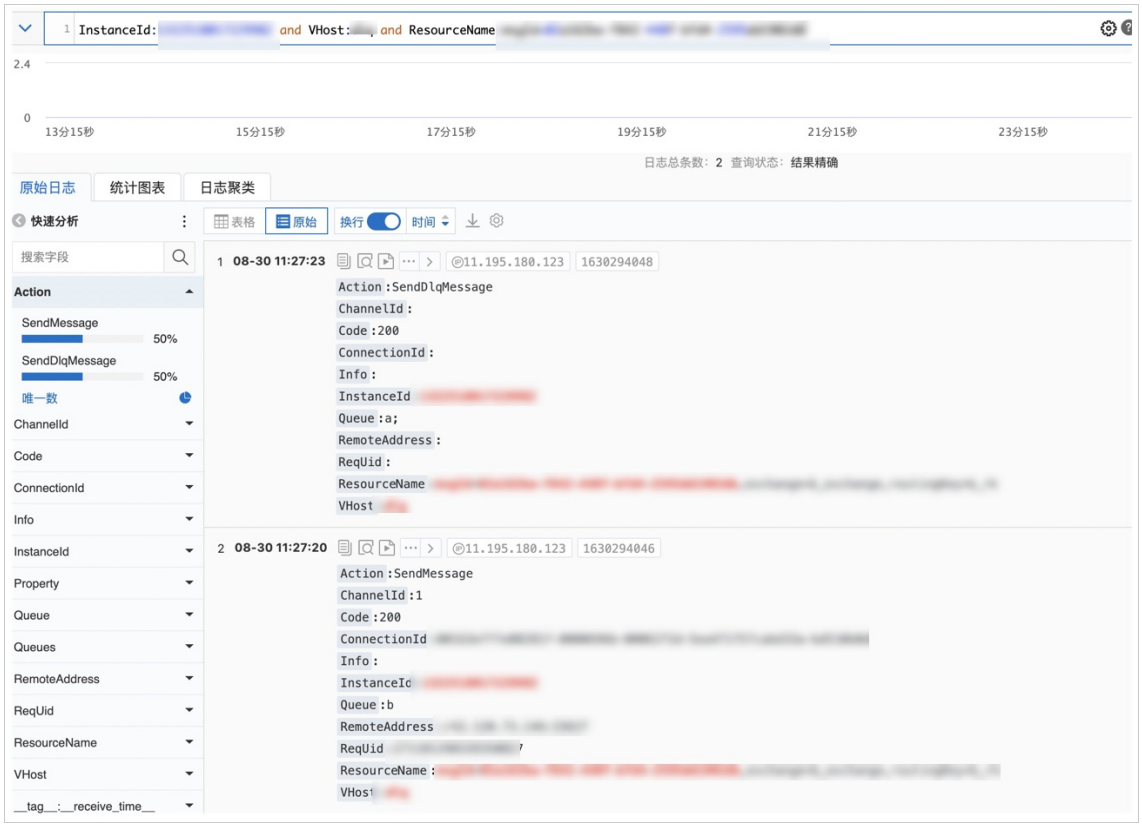
> **说明** 如果用户配置了死信队列才会有该条日志。

- Queue的TTL到期进入死信队列
  i. 设置Queue的死信属性以及TTL。

```
Map<String, Object> argument = new HashMap<>();
argument.put("x-dead-letter-exchange", [exchangeName]);
argument.put("x-dead-letter-routing-key", [routingKey]);
argument.put("x-message-ttl", [ttl]);
channel.queueDeclare([queueName], true, false, false, argument);
```

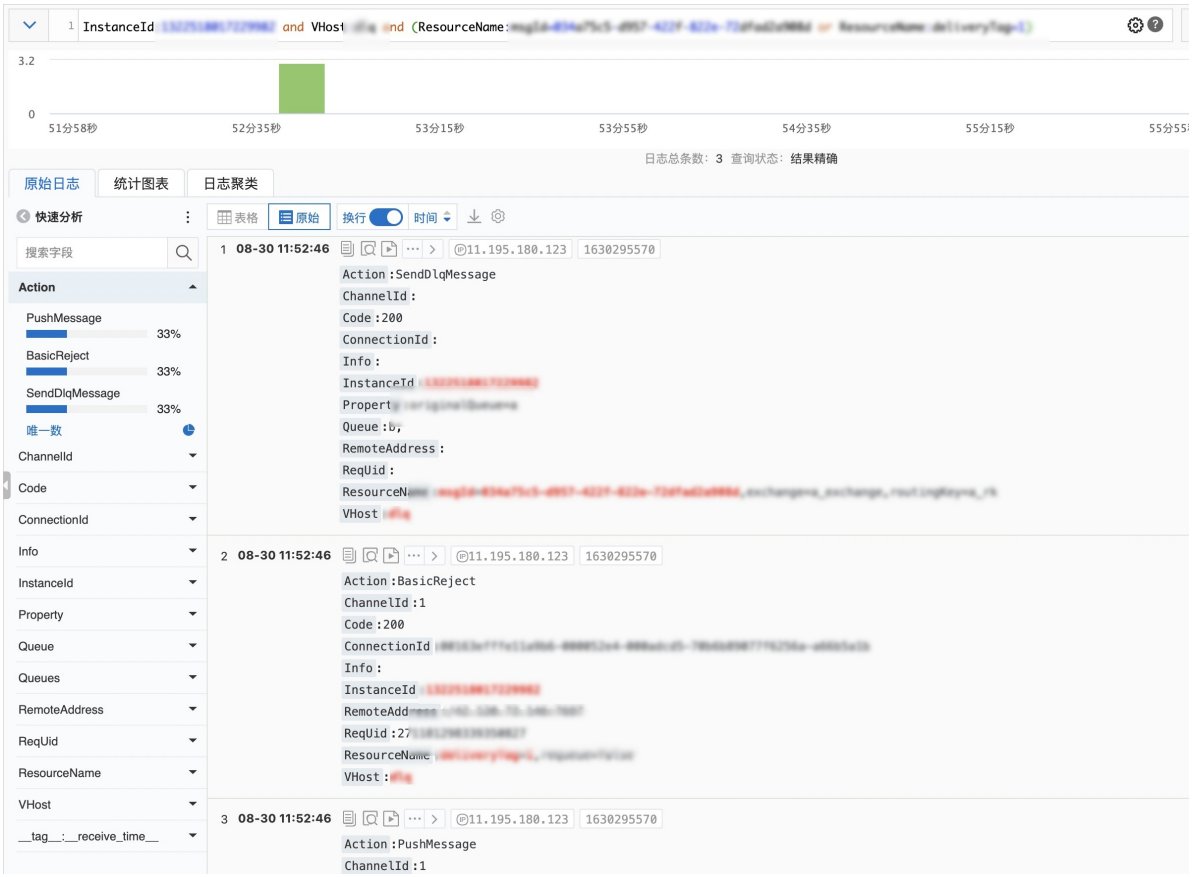
  ii. 日志搜索框输入如下，可以得到 `SendMessage` 和 `SendDlqMessage` 日志。

```
InstanceId:amqp-cn-i7m29o3s**** and VHost:dlq**** and ResourceName:msgId=034a75c5-d957-422f-822e-72dfad2a****
```



- 调用 `BasicReject` 、 `BasicNack` 时， `requeue=false`

```
InstanceId:amqp-cn-i7m29o3s**** and VHost:dlq**** and (ResourceName:msgId=034a75c5-d957-422f-822e-72dfad2a**** or ResourceName:deliveryTag=1)
```



其中 `PushMessage` 日志表示服务端推送消息到客户端，客户端在收到消息后调用 `BasicReject(requeue=false)` ，对应 `SendDlqMessage` 日志表示消息被路由到死信队列中。

5.3. 查询TPS统计图表

本文介绍基于 消息队列Rabbit MQ版当前日志快速查询TPS统计图表的方法。当您遇到TPS流量超限时，通过该方法可以及时查询秒级的TPS统计图表，帮助您高效识别出异常，保证业务正常运行。

前提条件

登录消息队列Rabbit MQ版控制台 ，将 消息队列Rabbit MQ版实例的消息操作日志推送到日志服务。具体操作，请参见 配置消息日志。

背景信息

当前云监控提供的图表是分钟级统计数据的平均值，无法展示秒级的TPS统计数据。消息队列Rabbit MQ版的TPS统计了每秒Client主动发起的AMQP协议方法请求数量。

TPS统计的AMQP协议请求方法如下：

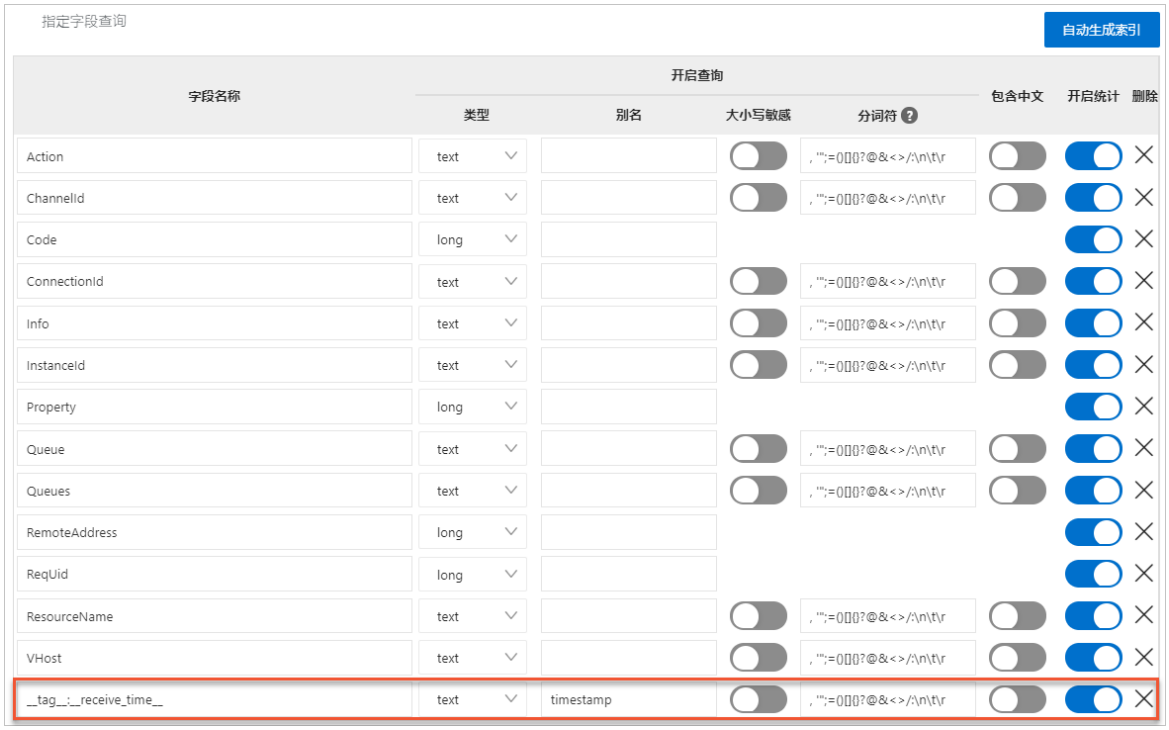
- ConnectionOpen、ChannelOpen
- QueueDeclare、QueueDelete、QueueBind、QueueUnbind
- ExchangeDeclare、ExchangeDelete
- ExchangeBind、ExchangeUnBind
- SendMessage、BasicConsume、BasicGet、BasicAck、BasicReject、BasicNack、BasicRecover

关于请求方法的详细描述，请参见请求方法。

操作步骤

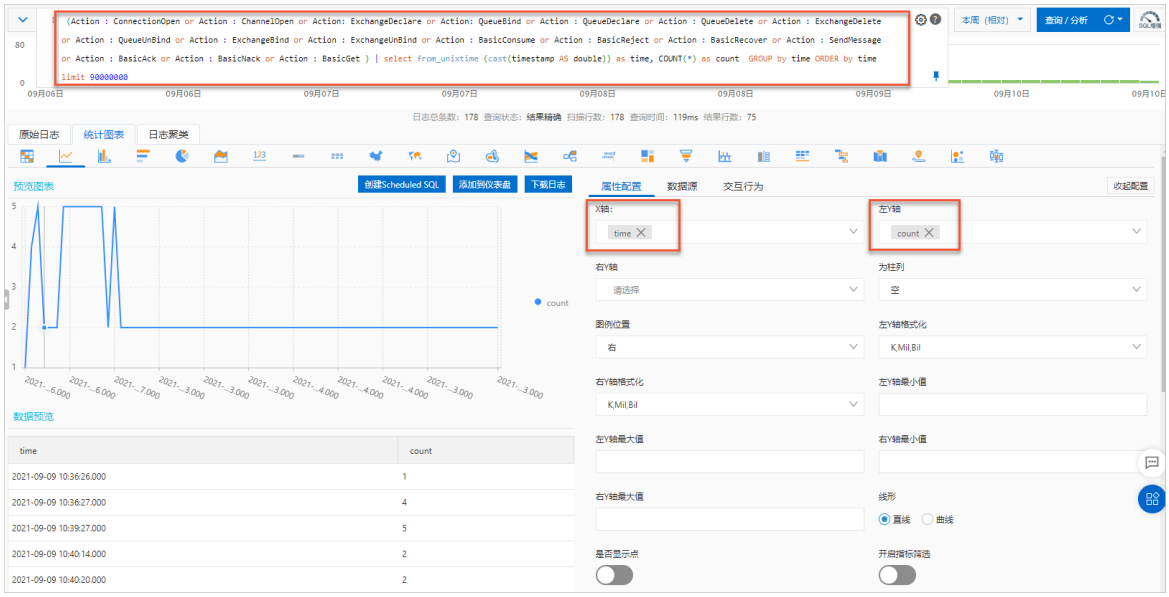
1. 登录 日志服务控制台 ，配置日志字段索引。

   具体操作，请参见配置索引。除自动生成的普通字段之外，还需手动添加 __tag__:__receive_time__ 字段，并设置别名为*timestamp*，如下图所示。



2. 设置查询的时间段，配置统计图表的属性，在搜索框输入SLS分析语句，查询TPS统计图表。具体操作，请参见 查询和分析日志。

```
(Action : ConnectionOpen or Action : ChannelOpen or Action: ExchangeDeclare or Action:
QueueBind or Action : QueueDeclare or Action : QueueDelete or Action : ExchangeDelete o
r Action : QueueUnBind or Action : ExchangeBind or Action : ExchangeUnBind or Action :
BasicConsume or Action : BasicReject or Action : BasicRecover or Action : SendMessage o
r Action : BasicAck or Action : BasicNack or Action : BasicGet ) | select from_unixtime
(cast(timestamp AS double)) as time, COUNT(*) as count GROUP by time ORDER by time limi
t 90000000
```



说明

- 其中*BasicNack(multiple=false)*，计TPS=1，*BasicNack(multiple=true)*，计TPS=N，因此通过SLS日志配置统计出来的TPS值会小于实际发起的请求量。
- 查询TPS流量图时，如果客户端的流量比较大，建议将查询的时间范围限制在1小时或是更小的范围，然后在SQL语句后面加上 `limit 90000000` ，或者 limit 取值尽可能大。

6.SpringBoot客户端使用最佳实践

本文介绍如何在SpringBoot客户端使用5671端口收发消息。

示例代码

创建消息队列RabbitMQ版配置文件application.properties，根据代码提示信息，设置相关参数。

```
spring.rabbitmq.host=              #设置接入点，在消息队列RabbitMQ版控制台实例详情页面获取。
spring.rabbitmq.port=5671          #消息队列RabbitMQ版加密端口。
spring.rabbitmq.username=          #用户名，在消息队列RabbitMQ版控制台用户名密码管理页面查看。
spring.rabbitmq.password=          #密码，在消息消息队列RabbitMQ版控制台用户名密码管理页面查看。
spring.rabbitmq.dynamic=false      #不创建AmqpAdmin bean。
spring.rabbitmq.listener.simple.acknowledge-mode=       #指定Acknowledge的模式。
# SSL相关配置
spring.rabbitmq.ssl.enabled=true     #启用SSL支持。
spring.rabbitmq.ssl.algorithm=TLSv1.2  #SSL使用的算法。
spring.rabbitmq.ssl.validate-server-certificate=true      #启用服务器端证书验证。
spring.rabbitmq.ssl.verify-hostname=false    #不启用主机校验。
```

7.实例限流最佳实践

消息队列Rabbit MQ版会对单实例的TPS流量峰值进行限流，本文介绍消息队列Rabbit MQ版实例的限流规则、限流后的行为以及限流最佳实践等。

限流后行为

当消息队列Rabbit MQ版实例的TPS流量峰值超过您所购买实例的TPS规格上限时，消息队列Rabbit MQ版实例会被限流。

限流后的行为如下：

- 消息队列Rabbit MQ版服务端会返回错误码信息。
- 消息队列Rabbit MQ版服务端**关闭当前请求的Channel**。代码中可以捕获异常重新开启Channel。

错误码信息：

- 错误码：reply-code=530
- 错误信息：reply-text=denied for too many requests

Java客户端错误堆栈示例：

```
Caused by: com.rabbitmq.client.ShutdownSignalException: channel error; protocol method: #me
thod<channel.close>
(reply-code=530, reply-text=denied for too many requests, ReqId:5FB4C999314635F952FCBFF6, E
rrorHelp[dstQueue=XXX_test_queue,
srcExchange=Producer.ExchangeName,bindingKey=XXX_test_bk, http://mrw.so/6rNqO8], class-id=5
0, method-id=20)
    at com.rabbitmq.client.impl.ChannelN.asyncShutdown(ChannelN.java:516)
    at com.rabbitmq.client.impl.ChannelN.processAsync(ChannelN.java:346)
    at com.rabbitmq.client.impl.AMQChannel.handleCompleteInboundCommand(AMQChannel.java:182
)
    at com.rabbitmq.client.impl.AMQChannel.handleFrame(AMQChannel.java:114)
    at com.rabbitmq.client.impl.AMQConnection.readFrame(AMQConnection.java:672)
    at com.rabbitmq.client.impl.AMQConnection.access$300(AMQConnection.java:48)
    at com.rabbitmq.client.impl.AMQConnection$MainLoop.run(AMQConnection.java:599)
    at java.lang.Thread.run(Thread.java:748)
```

实例秒级TPS峰值查询

基于消息队列Rabbit MQ版当前日志快速查询TPS统计图表的方法，请参见查询TPS统计图表。

通过查询实例实际使用的秒级TPS峰值，您可以了解业务的流量波动情况和流量峰值，判断实例规格是否满足业务需求。

实例TPS计算规则

以下接口调用时，会被计算进TPS流量中，即调用一次接口，计算为一次TPS。

- ConnectionOpen、ChannelOpen
- QueueDeclare、QueueDelete、QueueBind、QueueUnbind
- ExchangeDeclare、ExchangeDelete
- ExchangeBind、ExchangeUnBind
- SendMessage、BasicConsume、BasicGet、BasicAck、BasicReject、BasicNack、BasicRecover

> **说明**
>
> 延时消息是消息队列RabbitMQ版的高级特性消息，此类消息将按照普通消息的5倍倍率进行计算。
>
> 示例：发布延时消息10次/秒，计算TPS时，按照10×5=50次/秒计算。