消息队列 RabbitMQ 版

实践教程

腾讯云

文档目录

实践教程
RabbitMQ 客户端实践教程

最近更新时间：2024-06-24 15:31:41

请不要每次发送消息都创建连接/通道

RabbitMQ 中创建连接是一个耗时/耗资源的操作，每个连接会使用至少 100KB 的内存，连接数过多会增大 Broker 的内存压力。应该在程序启动时创建连接，每次发送消息时复用此长连接，提升发送性能和减少服务端内存占用。
通道是一种更加轻量的通信方式，建议尽可能多的使用通道来复用连接。但最好不要跨线程并发使用同一个通道，因为很多 RabbitMQ 客户端的通道实现并不是线程安全的。

为生产者设置合理的发送超时时间

RabbitMQ 不同语言和版本的客户端设置了不同的默认发送时间，部分客户端默认超时时间过长，例如 580 秒或 900 秒。网络异常时，过长的发送超时时间会阻塞发送端线程，甚至进一步造成雪崩效应，建议根据业务场景设置合理的超时时间，3 秒是一个推荐的值。

生产者和消费者使用独立的连接

由于 RabbitMQ 有独特的流控机制，如果生产者和消费者复用同一个物理连接，而消费流量大触发了流控，可能会导致生产者被流控导致发送慢或超时，因此建议生产者和消费者初始化时使用不同的物理连接，避免互相影响。

消费者不建议开启自动消息确认

RabbitMQ 服务端提供了至少投递一次的消费语义，以确保消息正确地触达到下游业务系统。一旦消费端开启了自动确认消息，服务端将消息推送给消费方后，就自动确认删除消息，即使消费端处理消息时出现了异常也不会重试，可能会导致业务上漏处理消息。

消费者幂等处理消息

RabbitMQ 服务端提供了至少投递一次的消费语义，极端场景下有可能重复投递消息，因此建议关键业务处理消息时一定要做幂等处理，即使收到重复消息，也不会产生负面业务影响。
业务幂等处理可以通过在消息中加入唯一业务标识，消费端消费时检查此类标识和消息状态等，根据业务需求处理重复消息，保证即使重复接受消息也不会产生业务上的负面影响。

限制队列长度，避免大量堆积消息

过长的队列（大量消息堆积）会占用大量内存，消耗更多服务端系统资源，不仅在运行时需要更长的时间来进行状态同步，并且会导致服务端 Broker 启动恢复时间大大增加。
较短的队列会提供更快的处理速度和系统性能。
因此需要客户端尽可能提高消费能力、队列维度限制 max-length 等手段来保证队列尽可能短。

使用 Consume 还是 Get 消费消息?

Get 是一种基于轮询的拉模式消费方式，每消费一条消息都需要向 Broker 发送一个请求，如果队列中没有消息，可能会导致大量无效空拉导致资源占用。而 Consume 一次可以接收一批消息，并且由服务端根据实际情况推送消息，在绝大多数情况下都应该使用 Consume 而不是 Get 来消费消息。
如果业务处理必须使用 Get 消费消息，应关注业务层面的 Get 机制，避免持续 Get Empty 空拉（队列中已经无待消费的消息，但消费端一直持续 Get）导致的服务端 CPU 负载异常高。

消费者设置合理的 Prefetch Count

Prefetch 设置是消费端为了提升消费吞吐，预先将消息推送到消费端缓存，降低消费等待和延迟的机制。但如果 Prefetch Count 如果过大或不限制，会导致大量消息缓存在消费端，并且服务端 Broker 也在内存维护 unack 消息的状态，占用大量资源；如果消息持续 unack 状态中，这些消息也无法被其他空闲的消费者消费，表现为消费延迟增加或消费者负载不均衡。
建议根据业务消费速率，将 Prefetch Count 设置在合理范围。

消费者设置合理的异常处理策略

消费端消费消息时，遇到了无法处理的异常，未设置自动确认的消息会触发消息重试。如果异常持续无法正常退出，会触发消息被无限重试，不仅会对 Broker 端造成较高的负载压力，也会导致后续的消息无法被合理消费。

客户端重连机制确认

服务端 Broker 在例如 OOM、容器母机故障等极端场景下会自愈重启，日常业务自行操作集群升配等场景也会触发 Broker 重启，为避免 Broker 重启期间客户端持续连接异常，请确保客户端实现了自动重连机制。

客户端 SDK 请不要关闭 heartbeat 设置

heartbeat 在服务端和客户端分别有一个配置值（服务端为 60 秒），最终生效的 heartbeat 由服务端和客户端协商决定，且不同语言/版本的客户端协商机制不同。客户端设置 heartbeat=0 即关闭心跳检测，会导致服务端无法自动剔除长期无数据交互的无用连接，最终可能引起非预期的连接泄露。

RabbitMQ 消息可靠性实践教程

最近更新时间：2024-06-24 15:31:41

消息持久化

为了确保队列元数据和队列中的消息在 Broker 重启后不丢失，建议将队列设置为 durable、消息设置为 persistent，这样队列在接收到消息后会立即将其持久化到磁盘上。
非持久化的消息也会占用更多的服务端内存资源，极端情况下引起服务端内存负载过高。

发送端 Confirm

Confirm 机制可以确保消息被成功地发送到 Broker。但如果发送消息时不设置 mandatory，无论消息是否成功路由到目标队列，Broker 都会回应 confirm 给发送端。如果设置了 mandatory（延迟交换机不支持设置 mandatory），消息无法路由时 Broker 会将消息原路返回客户端，客户端可以通过实现 **basic.return** 的处理来感知这些无法路由的消息；消息可以成功路由到目标队列时，Broker 才会回应 confirm 给发送端。

消费端 Acknowledgement

消费端 ACK 机制可以确保客户端收到消息，提供了 at least once 级别的消费语义保证，确保消息被正确处理后才能被删除。但这也需要客户端做好幂等，避免重复消费消息引发错误，而且未被 ACK 的消息会堆积在内存中导致客户端和服务端内存占用增加。

开启镜像队列

镜像队列通过将队列数据复制到集群内其他 Broker 上来保证队列的高可用。配置镜像队列策略可能会增加 Broker 的启动时长和资源占用，但可以确保队列在单个 Broker 故障时仍处于可用状态，尽量确保不丢失消息。
配置镜像队列策略时，应避免设置 ha-sync-mode=automatic，该配置会引起服务端 Broker 重启后自动全量同步队列数据（无论队列数据是否被同步过）。如果同步的队列堆积数据过多，最终会导致 Broker 同步数据时间过长、持续占用内存资源等问题，且队列同步完成前队列处于不可用状态，对业务可用性、服务端稳定性都有严重影响。

其他注意事项

最近更新时间：2024-06-24 15:31:41

rabbitmq_delayed_message_exchange 插件实现的延迟消息

1. 当前插件的设计不适用于大量延迟消息（未调度的消息达数十万甚至数百万条）的场景，生产环境请谨慎评估消息量级，避免非预期的长时间延迟、消息丢失等问题。
2. 延时消息在每个节点上只有一个持久化副本，如果节点无法正常运行（例如由于消息堆积导致持续 OOM 后重启且无法恢复），则该节点上的延时消息无法被消费端消费。
3. 延时交换机不支持设置 **mandatory**，生产者无法通过 **basic.return** 事件感知到无法路由的消息，因此发送延时消息前请务必保证对应的交换机、队列、路由关系存在。

综上所述，我们强烈建议您不要使用这一插件，转而使用死信队列来间接实现 延时消息 。如果您在了解了这个插件的若干缺陷后仍然要使用，那我们强烈推荐您将延迟消息数量保持在一个尽可能低的水平，避免触发内存负载高的问题。

网络分区

1. 网络分区是在使用 RabbitMQ 时不得不面对的一个问题。网络分区会带来集群状态的不一致，即使是网络恢复后，RabbitMQ 仍需通过重启 Broker 的方式来重新同步状态。腾讯云 RabbitMQ 目前使用的 autoheal 模式，它会自动决出一个获胜的分区，然后重启非信任分区内的 Broker。
2. 建议客户端做好以下措施，尽可能减小网络分区带来的负面影响：
   - 消息发送方，考虑发送消息时配合使用 **mandatory** 机制并有处理 **basic.return** 的能力，以便及时处理网络分区时可能出现的消息路由失败。
   - 消息消费方，网络分区出现/处理过程中，可能会伴有消息重复，消费端需要做好幂等处理。

告警配置

腾讯云提供了集群/节点等多维度的监控指标，详情请见 监控告警 。
强烈建议重点关注节点 CPU、内存、磁盘利用率、堆积消息数量等几个指标并配置告警，避免服务端持续负载过高影响集群稳定性。

消息轨迹使用限制

消息查询的实现原理概述：腾讯云控制台打开 VHost 的 Trace 插件后，服务端组件会消费对应 RabbitMQ 集群的轨迹消息，通过一系列处理后可实现控制台查询消息轨迹的功能。
由上述原理，消息轨迹依赖于服务组件消费轨迹消息，由于服务组件为底层公共服务，无法保证大流量的 RabbitMQ 集群的轨迹消息可以被及时消费；如轨迹消息堆积，会造成集群内存负载高等问题，影响 RabbitMQ 集群稳定性。
因此，不建议在生产环境尤其整体集群（包括所有 VHost）发送 TPS 超过 1000 的场景下开启 Trace 插件，Trace 插件建议使用在小流量验证/排查问题场景，不建议在生产环境使用。

RabbitMQ 支持 MQTT 协议使用说明

最近更新时间：2024-01-29 15:17:01

方案介绍

MQTT 是目前广泛使用的物联网协议，RabbitMQ 是基于 AMQP 0.9.1 协议实现的广泛使用的开源消息队列产品，RabbitMQ 以插件的形式支持了 MQTT 协议，可以在 RabbitMQ 集群上方便的支持 MQTT 协议，实现对物联网等业务场景的支持。

社区参考文档：

1. RabbitMQ 3.11 之前版本插件支持 MQTT 协议： MQTT Plugin — RabbitMQ
2. RabbitMQ 3.12 版本原生支持 MQTT 协议： Serving Millions of Clients with Native MQTT | RabbitMQ – Blog

操作步骤

步骤1：购买云上或自建RabbitMQ集群

云上直接购买 RabbitMQ 集群， 立即选购 。

或者自己搭建 RabbitMQ 集群，详细参见： Downloading and Installing RabbitMQ — RabbitMQ 。

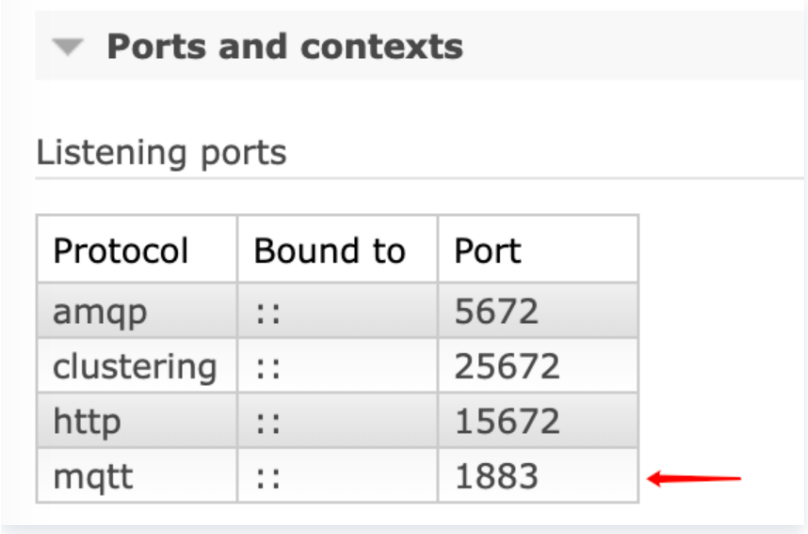
步骤2：开启MQTT插件

通过在集群节点执行以下命令开启MQTT插件：

```
sudo  rabbitmq-plugins enable rabbitmq_mqtt
```

腾讯云的 RabbitMQ 插件管理功能正在开发中，目前可以通过 提交工单 来开启 MQTT 插件和打通网络工作。

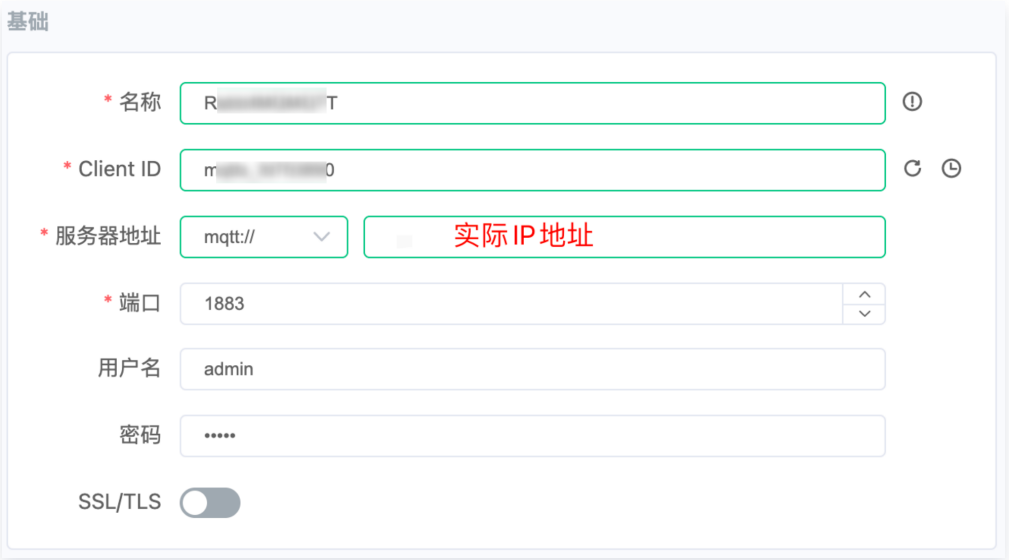
开启 mqtt 插件后，可以在管控台看到新增的1883端口：



步骤3：验证 MQTT 的可用性

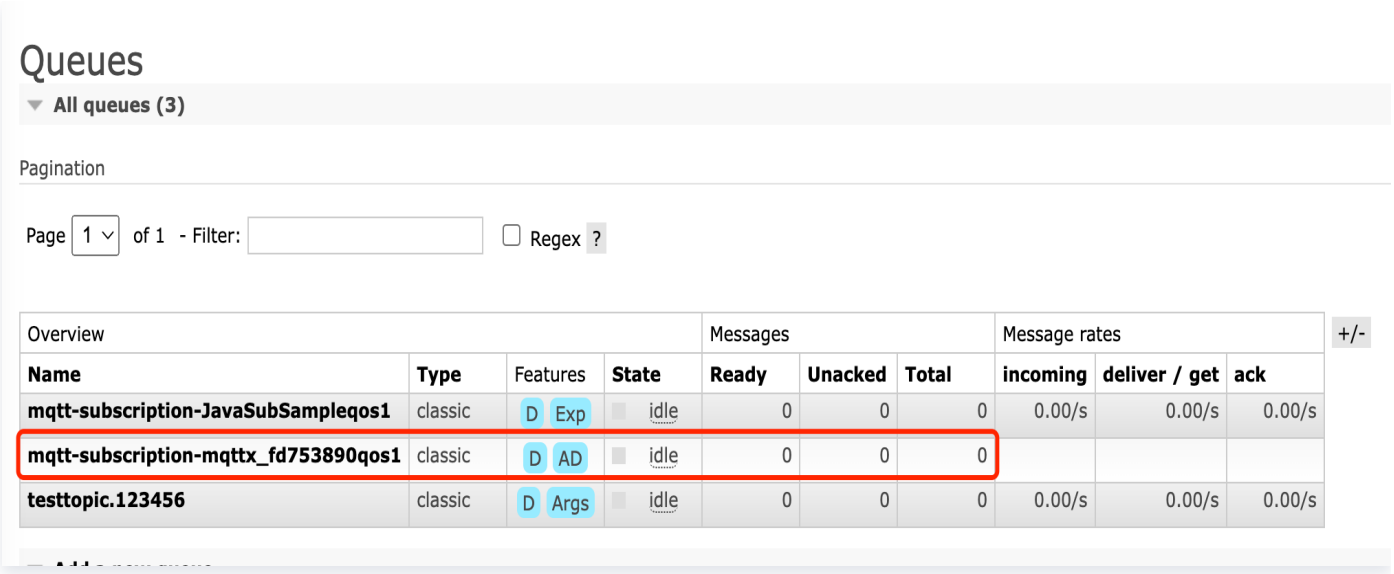
可以下载常用的 mqttx（ MQTTX：全功能 MQTT 客户端工具 ）客户端工具来验证一下：

1. 新建连接，填写地址和端口。用户名和密码选用 RabbitMQ 的用户名和密码即可。

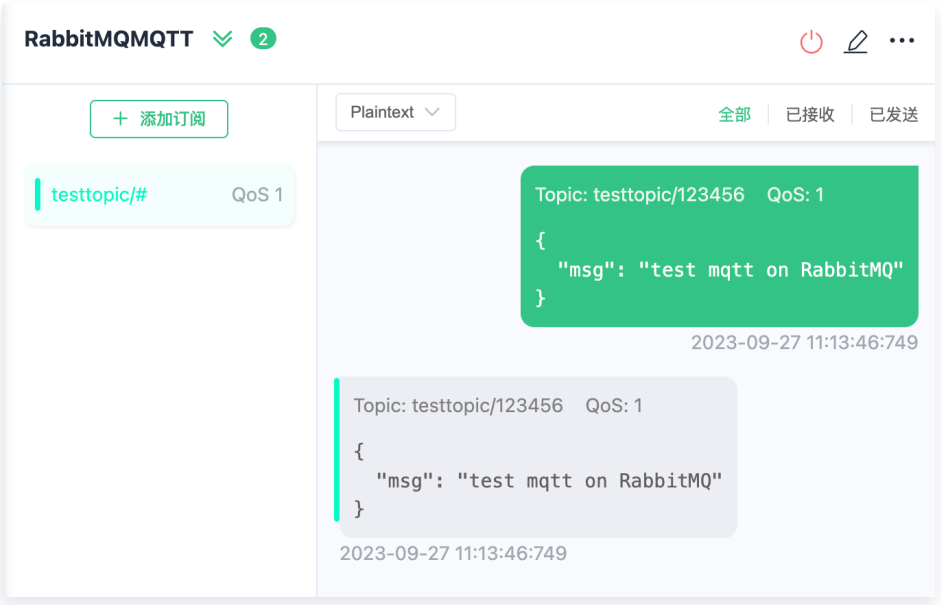


2. 新建订阅，订阅 testtopic/# 主题的消息。

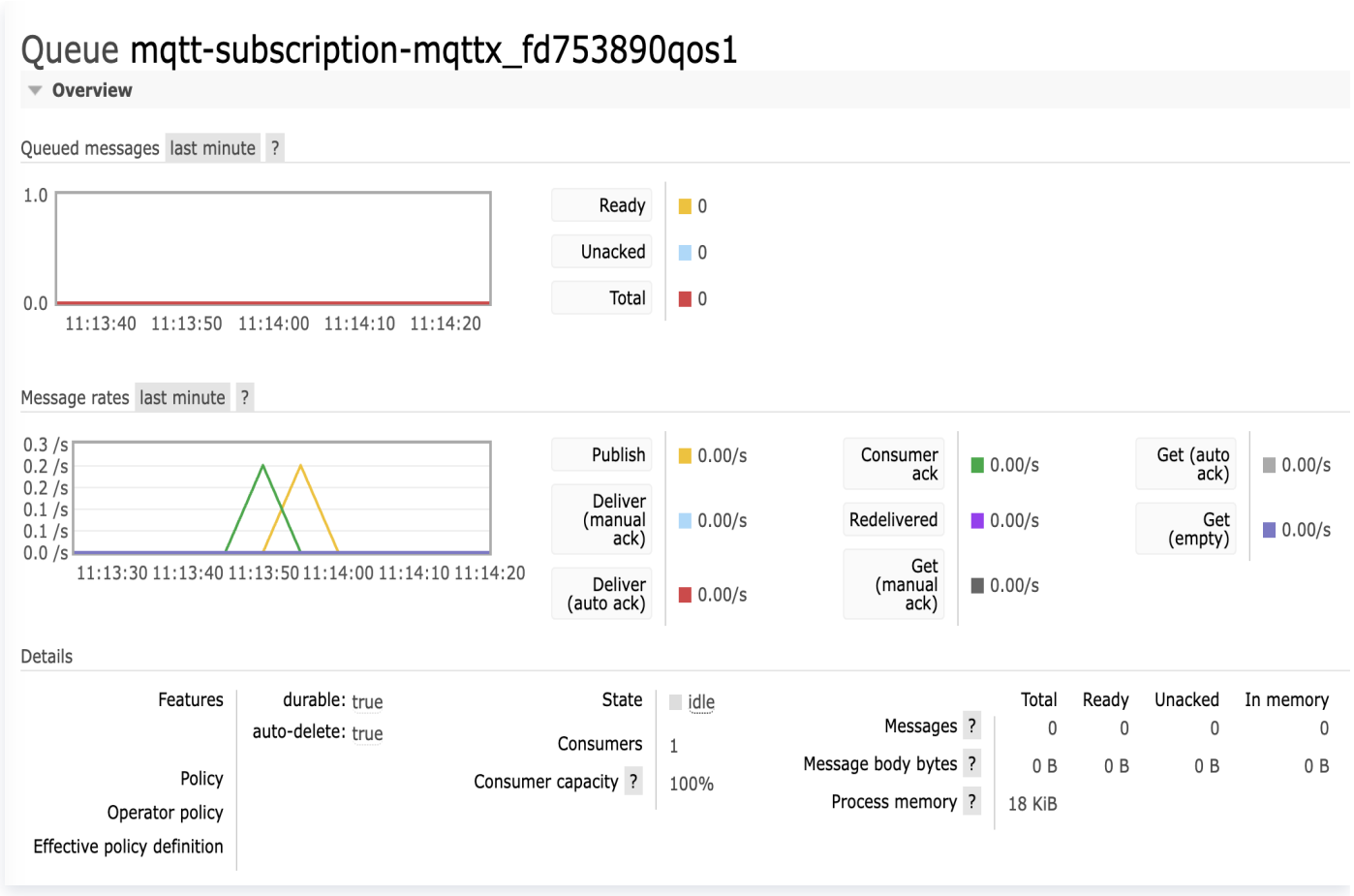
3. 检查 RabbitMQ 的队列情况，可以看到每个订阅会在 RabbitMQ 新增一个队列。



4. 收发消息验证，发送 testtopic/123456消息，立即可以通过订阅收到消息。



5. 查看 RabbitMQ 监控，可以看到刚才的队列有一次收发消息的监控。

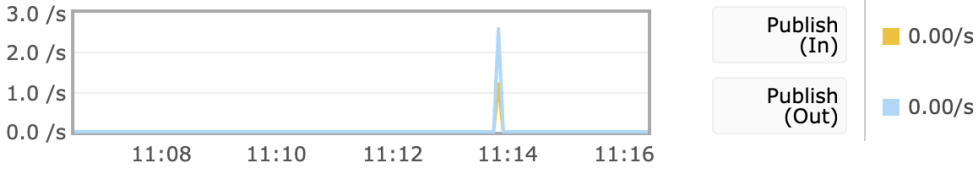


6. 验证 MQTT 上行消息和 RabbitMQ 的消息互通性，MQTT 发送消息也可以路由到普通队列，由 RabbitMQ 下游应用来消费这个消息。

Exchange: amq.topic

Overview

Message rates last ten minutes ?



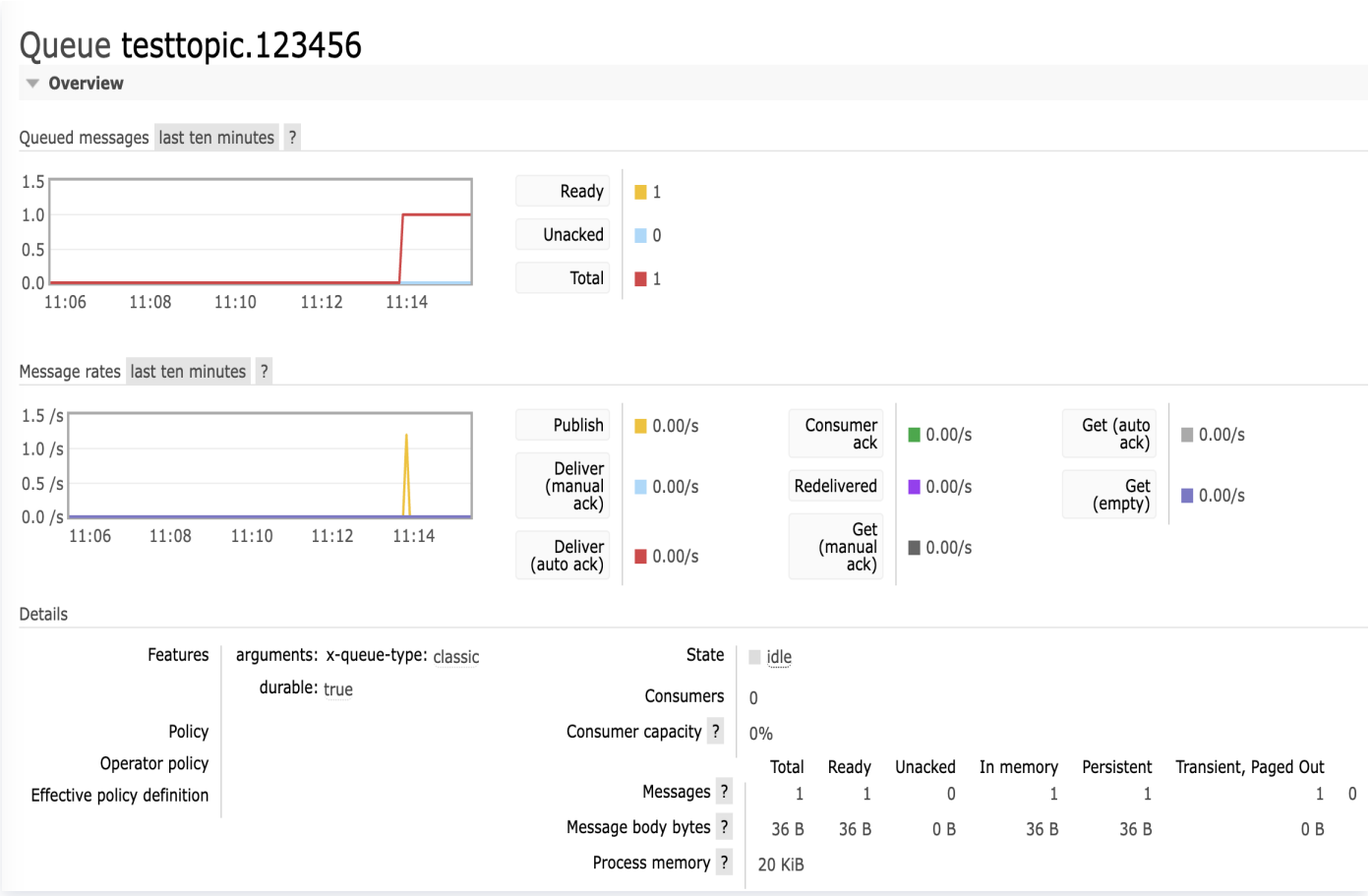
Details

Type	topic
Features	durable: true
Policy	

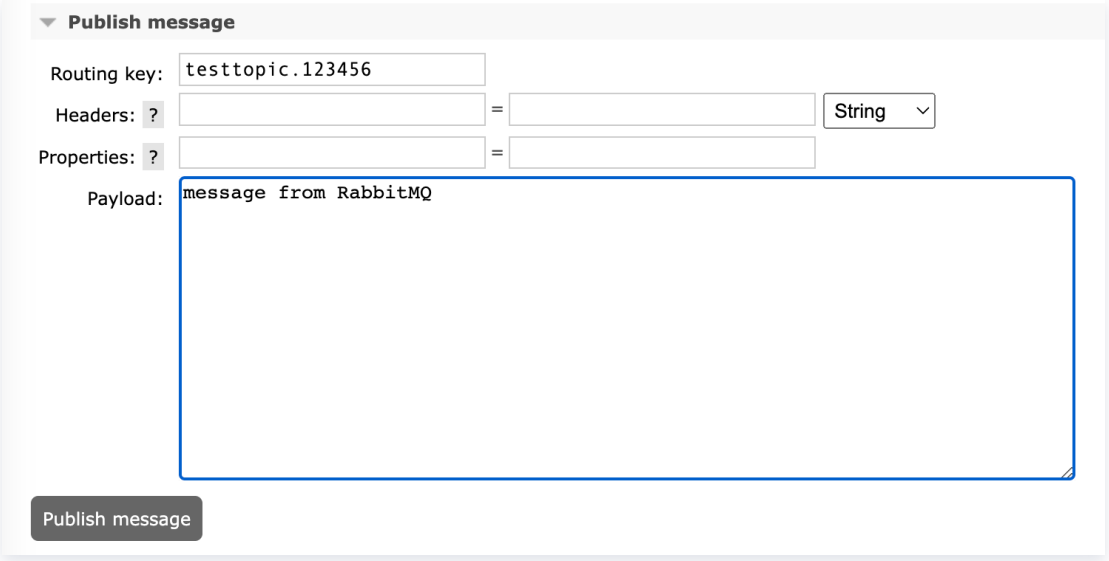
Bindings

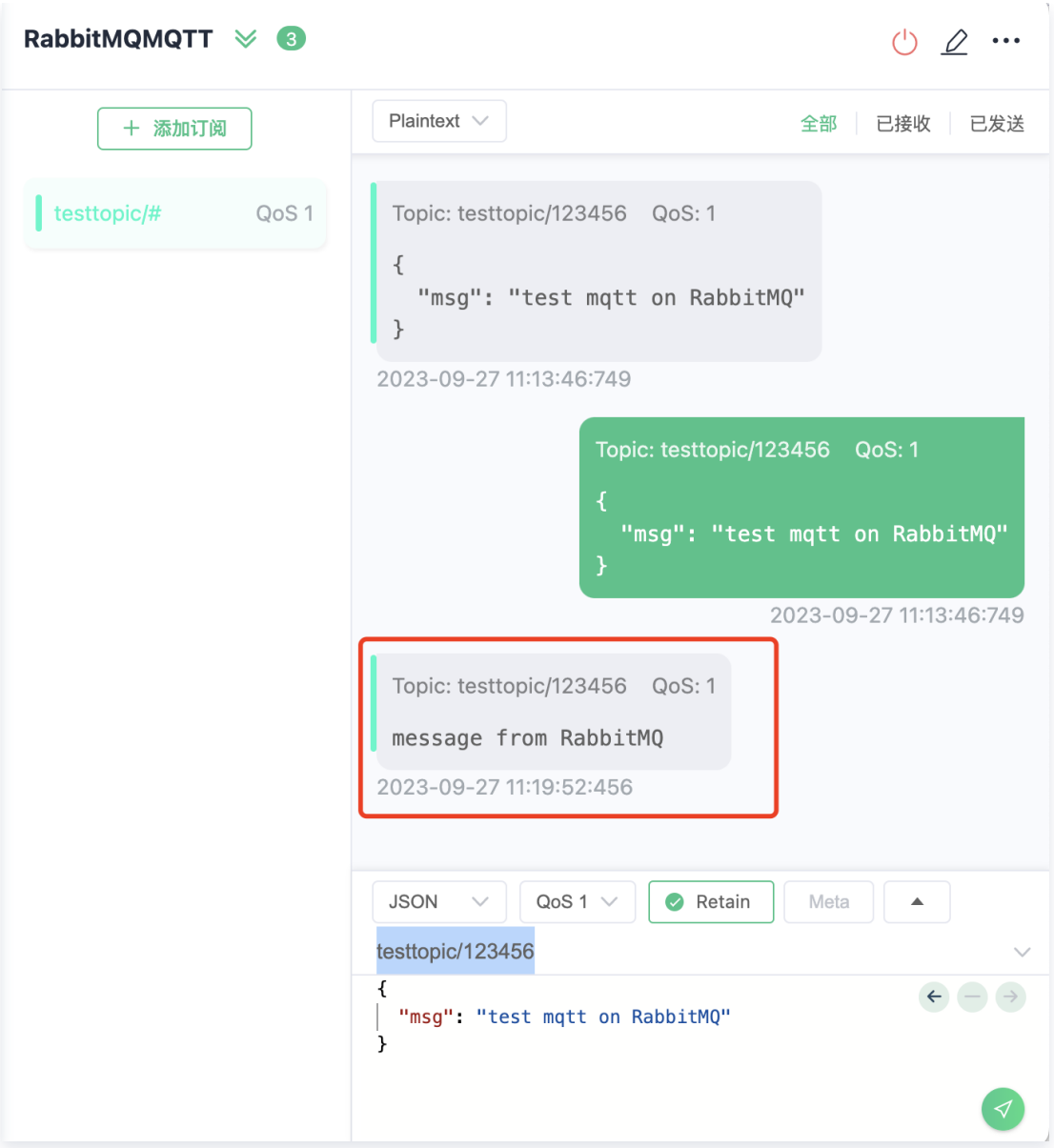
This exchange

To	Routing key	Arguments	
mqtt-subscription-JavaSubSampleqos1	testtopic.123456		Unbind
mqtt-subscription-mqttx_fd753890qos1	testtopic.#		Unbind
testtopic.123456	testtopic.123456		Unbind



7. 验证 RabbitMQ 和 MQTT 下行消息的互通性，可以 RabbitMQ 发送消息，MQTT 订阅到消息。

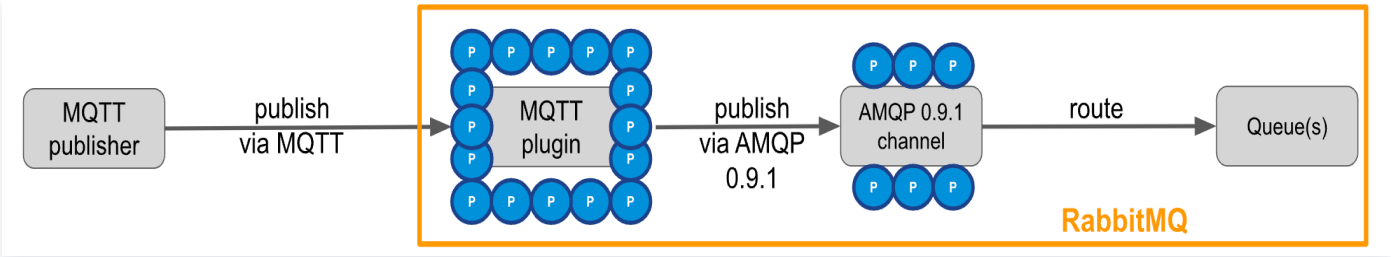




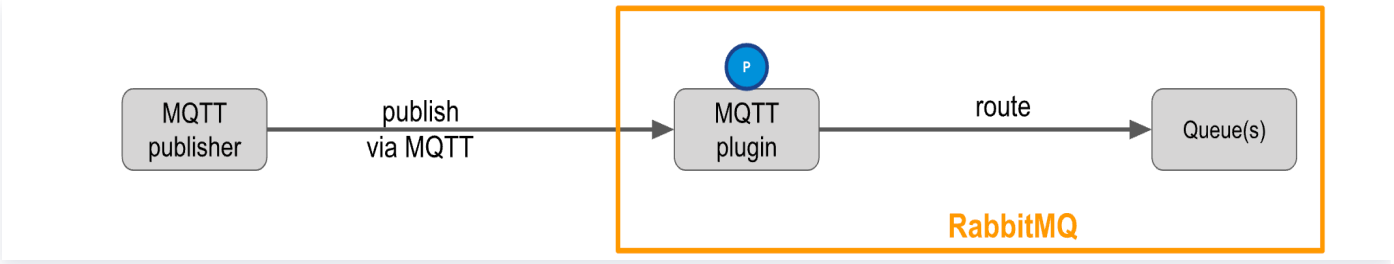
8. 验证总结。通过以上验证可以确认，RabbitMQ 的 MQTT 插件，可以支持正常的 MQTT 消息收发，也可以支持 MQTT 上行消息到应用，也可以支持应用发送 MQTT 消息下行消息到订阅端，并且有完善的监控。

实现原理

- 3.11 版本之前的实现原理，通过 MQTT 消息转为 AMQP 协议，实现 MQTT 的消息收发。



- 3.12 之后版本的实现原理，不再经过 AMQP 协议转换，理论上有更好的性能。



注意事项

- 我们推荐使用现在的稳定版本3.8.x 或3.11.x等稳定版本，没有已知 Bug。
  - 因为3.12.x版本是新发布的版本，MQTT 实现有重新实现刚发布初始版本，我们在验证过程中，发现监控不准，RabbitMQ 互通偶尔异常等问题。因此**不推荐生产环境使用3.12版本**。
- 目前支持主流的 MQTT v3 和v3.1版本，暂不支持v5版本，RabbitMQ 预计3.13版本才会支持 。
- MQTT 协议的 topic 使用"/"分割 topic， AMQP 协议的 Topic(Routingkey)使用 "."分割 topic，所以在协议转换的时候会自动转换，应用使用的时候要注意这个差别。
- 不推荐 MQTT 使用匿名连接或“no login credentials”,因为 AMQP 协议会自动转换为默认用户 guest 或 mqtt.default_user，不方便做权限管控。
- 关于订阅持久性，注意 MQTT 和 AMQP 队列持久性的映射。
  - Transient clients that use transient (non-persistent) messages
  - Stateful clients that use durable subscriptions (non-clean sessions, QoS1)优先使用镜像队列，不要使用 Quorum Queues 特性，因为 Quorum 要求至少三节点，并且新特性稳定性待验证，暂不推荐使用。
- 优先使用镜像队列，不要使用 Quorum Queues 特性，因为 Quorum 要求至少三节点，并且新特性稳定性待验证，暂不推荐使用。