

阿里云 专有云企业版 消息队列 RocketMQ 版

MQTT运维指南

产品版本：V3.11.0

文档版本：20200511

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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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
	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。	 禁止： 重置操作将丢失用户配置数据。
	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。
	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。
	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。	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

法律声明.....	I
通用约定.....	I
1 运维概述.....	1
1.1 产品架构.....	1
2 高危操作.....	3
2.1 运维操作的危险级别定义.....	3
2.2 高危操作和文件.....	3
3 运维准备.....	4
4 例行维护.....	5
4.1 组件启停.....	5
4.2 组件巡检监控.....	6
4.3 服务巡检监控.....	6
4.4 日志.....	6
4.5 人工巡检.....	6
4.5.1 Dauth 的巡检.....	6
4.5.2 Diamond Server 的巡检.....	7
4.5.3 MQTT Console 的巡检.....	7
4.5.4 MQTT CS 的巡检.....	7
4.5.5 MQTT DS的巡检.....	8
4.5.6 TLog Console的巡检.....	8
5 故障处理.....	9
5.1 告警处理.....	9
5.1.1 订阅关系过多.....	9
5.1.2 发送消息失败.....	10
5.1.3 离线消息太多.....	10
5.1.4 离线消息匹配延迟.....	10
5.1.5 连接数太多.....	11
5.1.6 消息接收延迟.....	11
5.1.7 消息收发 TPS 过高.....	12
5.2 业务问题处理.....	12
5.2.1 反馈连接不上 MQTT 服务.....	12
5.2.2 反馈连接经常断开.....	13
5.2.3 反馈收不到消息.....	13
6 运维命令.....	15
6.1 概述.....	15
6.2 查看客户端是否在线.....	15
6.3 查看客户端连接轨迹.....	15
6.4 查询某个 topic 在线订阅的客户端.....	16
6.5 查询某个分组下客户端数量.....	16

6.6 查询某条消息收发轨迹.....	17
7 容量评估.....	20
7.1 MQTT CS组件.....	20
7.2 磁盘使用率.....	21
7.3 数据库 组件.....	21
8 日志参考.....	23
9 更新 MQTT 配置.....	24

1 运维概述

本章介绍微消息队列 MQTT 的产品架构和运维架构。

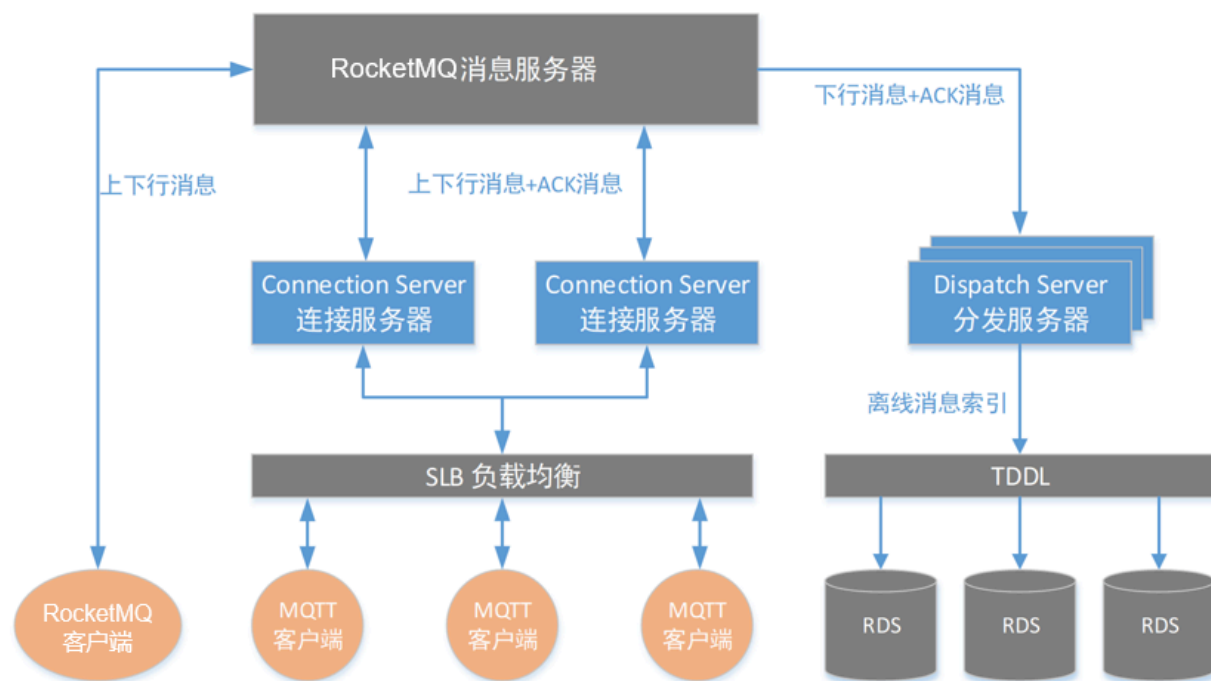
1.1 产品架构

本节内容旨在帮助您在运维过程中了解微消息队列 MQTT 的系统架构和其中包含的各组件。

微消息队列 MQTT 是阿里云推出的一款面向移动互联网以及物联网领域的轻量级消息中间件，针对移动互联网以及物联网 IoT 场景的消息传输特点，支持了包括 MQTT、STOMP、GB-808、新能源国标等主流通信协议。同时，微消息队列 MQTT 在数据传输层支持原生 TCP 长连接、SSL 加密、Websocket 等传输形式，支持包括 C/C++、Java、iOS、Android 等主流开发语言和平台。微消息队列 MQTT 在专有云环境强依赖 RocketMQ 作为存储组件，因此购买部署微消息队列 MQTT 需要先购买部署消息队列 RocketMQ，微消息队列 MQTT 的运维过程中涉及到消息存储相关的问题处理仍然需要参考消息队列 RocketMQ 的运维方式处理。

MQTT 服务主要由 MQTT Connection Server（MQTT CS）、MQTT Dispatch Server（MQTT DS）后端 RocketMQ、Dauth 权限服务、元数据库等部分组成。整体架构如下图所示：

各模块和组件的介绍如下：



MQTT CS

MQTT CS 组件是 MQTT 的核心网关节点，负责承担海量的 MQTT 客户端长连接和主要的消息收发处理逻辑。

MQTT CS 组件是一组无状态的服务节点，运行在物理机独占 docker 容器中，对外暴露了 SLB VIP 和域名，MQTT 客户端通过域名和 VIP 随机和一个后端节点建立 TCP 长连接，后续通过应用层心跳保活维持该长连接。

MQTT CS 除了承担长连接之外，MQTT 客户端发送消息到 CS 节点，CS 节点负责将消息转存到后端的 RocketMQ 存储，同时监听 RocketMQ 的消息通知，将下行消息拉取到网关节点，根据 MQTT 的订阅关系匹配消息推送给 MQTT 消费客户端。

MQTT DS

MQTT DS 组件是 MQTT 服务的后台服务角色，对 MQTT 客户端不暴露，负责转发来自 RocketMQ 的下行推送消息以及匹配生成离线消息。

MQTT DS 组件也是一组无状态的服务节点，运行在管控共享的 docker 容器中，对外不暴露服务。

MQTT DS 会从 RocketMQ 拉取需要推送的消息，转换成内部推送事件传递到 CS。

MQTT DS 同时负责匹配生成离线消息，当 MQTT 客户端使用了持久化订阅方式订阅消息后，客户端如果不在线期间产生了消息，DS 会匹配生成离线消息存储到数据库，等待客户端再次上线后，由 DS 捞取数据库的数据再次生成推送消息传递到 CS 节点进行重试推送。

MQTT Console

MQTT Console 和 RocketMQ Console 共享资源，同进程插件方式部署，当用户购买了 MQTT 后会自动开启 MQTT 相关功能。因此控制台相关运维文档参考 RocketMQ。

RocketMQ 消息存储

MQTT 使用了 RocketMQ 作为消息存储模块，该部分参考 RocketMQ 的运维文档。

外部依赖服务

MQTT 同时依赖了很多外部服务，例如 Dauth、Tlog，该部分的功能和依赖同 RocketMQ 用法完全一致，请参考对应的运维文档维护。

2 高危操作

2.1 运维操作的危险级别定义

所有运维操作按业务场景和操作的危险程度被分为三级：G1、G2、G3。

- G1** L1/L2 驻场人员可以根据文档完成操作，不需要做变更申请，该操作按文档执行完全安全，该操作不会对业务产生影响。
- G2** L1/L2 驻场人员可以根据文档指引，需要做变更申请，向产品侧咨询确认方可执行操作。该操作不会对业务产生影响。
- G3** L1/L2 驻场人员可以根据文档指引，需要做变更申请，向产品侧以及客户侧咨询确认方可执行操作（提交工单）。该操作有可能对业务产生影响。

2.2 高危操作和文件

本文将列出您在运维时需格外注意的高危运维操作和不能随意清理的高危文件。

高危操作

高危操作即 **G3** 级别的运维操作，进行该类操作时请按照相应的指导进行，详情请参见[运维操作的危险级别定义](#)。

高危文件

高危文件指不能被随意清理的文件，包括：

- 从文件产生到现在都没有动态增长的文件，原则上不能清理。如需要清理（如一些陈旧的包，升级时遗留下来不再使用的版本），需要与产品客户进行确认。
- 配置文件不能清理。所有 conf 目录下的内容都是受保护的。
- 应用程序包目录下的文件不能清理。
- 当前正在使用的日志文件不能清理。例如日志文件有mw.log, mw-2018-07-22.log，这种场景下，当前生效的文件 mw.log 不能被清理。

文件名称	用途	说明
/home/admin/proxy	MQTT 服务安装包	该安装包不允许移动或者删除
/home/admin/svn_conf	MQTT 服务配置文件	所有配置文件不允许移动或者删除

3 运维准备

本文介绍运维时需要的登录入口、账号、权限和工具。

表 3-1: 运维准备说明

名称	作用	说明
SSH远程登录工具（如 MobaXterm 或 PuTTY）	登录组件所在的机器。	登录使用的账号需要具有相应的操作权限。但不建议直接使用 root 或 admin 账号。
账号	用于登录控制台或具体机器	- 控制台的登录账号和密码请向 微消息队列 MQTT 技术支持人员获取。 - 登录微消息队列 MQTT 机器组件所在机器使用的账号需要具有相应的操作权限。但不建议直接使用 root 或 admin 账号。

4 例行维护

微消息队列 MQTT 的例行维护包含服务巡检监控和人工巡检两部分。

人工巡检

人工巡检即需要驻场运维同学登陆 MQTT 的各个服务节点按照指定的操作步骤检查系统各项指标是否正常。

服务巡检监控

服务巡检监控是指对微消息队列 MQTT 各系统组件的状态进行监测，包括 MQTT CS、MQTT DS 和 Console 进行运行时巡检和监控。

服务巡检监控是由 MQTT 的健康检查组件 Housekeeping 提供。Housekeeping 组件每分钟会对各监控项进行一次全面扫描诊断，发现异常状况会记录到本地日志，如有业务报警通道会通过钉钉群告警；同时，Housekeeping 每天还会定时发送系统巡检报告，方便您每天监控 MQTT 系统的健康状况。关于服务巡检监控的详细信息，请参见[服务巡检监控](#)。

4.1 组件启停

组件启停是通过人工命令行的形式。

1. 获取容器主机名称（操作高危级别：G1）

（以下使用`container\_id`表示）。针对不同的组件，执行相应的命令。命令会列出该组件的所有容器，每一行代表一个容器，每行的首个列输出是`container\_id`。

组件	命令
MQTT CS	<code>docker ps -a grep mqtt grep cs</code>
MQTT DS	<code>docker ps -a grep mqtt grep ds</code>
Console	<code>docker ps -a grep onsconsole</code>
Housekeeping	<code>docker ps -a grep mq grep housekeep</code>

2. 启动组件（操作高危级别：G3）

执行命令`docker start <container\_id>`启动容器。

3. 停止组件（操作高危级别：G3）

执行命令`docker stop <container\_id> -t 300`停止容器

4.2 组件巡检监控

MQTT 的组件巡检监控主要有以下内容：

- 巡检：周期性地拨测 URL 或端口等来判断 MQTT 的组件运行是否正常。目前支持的巡检方式有：HTTP、TCP、PING 和 JDBC。

4.3 服务巡检监控

MQTT 的服务巡检是通过 Housekeeping 组件执行，Housekeeping 组件和 MQTT 的巡检组件共享。

Housekeeping 会周期性地巡检 MQTT 的各个监控项，并把告警信息输出到 Housekeeping 容器中的位置 /home/admin/logs/aliware-mq-hkagent/warning.log。



说明：

当前的监控项都是固定配置的，不支持变更

4.4 日志

日志是 MQTT 运维的重要方式之一，主要用于监控及故障处理。所以需要对日志进行例行的维护。

MQTT 的日志主要包括基础监控日志、服务监控日志、容器监控日志和 JVM 监控日志。详细信息请参考[日志参考](#)。

4.5 人工巡检

对于某些逻辑比较复杂的巡检，通过 Butler 难以支持。所以微消息队列 MQTT 还提供了通过命令行的方式进行巡检。

人工巡检使用的命令为 Linux 系统通用的命令，具体参数需要根据实际环境设置。

4.5.1 Dauth 的巡检

Dauth 是一个部署在 Tomcat 内部的 Web 系统，因此巡检考虑如下几方面：

- Tomcat 进程是否存在，8080 端口是否已开通。
- Nginx 进程是否存在，80 端口是否能够正常转发到 8080。
- /home/admin/dauth/logs/*.log 中是否存在明显的异常。
- /home/admin/dauth/logs 中是否存在需要清理的无用日志文件。

- 执行 `curl 'localhost:8080'|grep Unsupported` 命令，若返回结果中有 **Unsupported** 字段，说明服务正常。
- /home/admin 对应磁盘是否有足够的容量。
- Dauth 也是 Diamond 的客户端，因此需要巡检如下方面：
 - /home/admin/logs/diamond-client/diamond-client.log 中是否有明显异常。
 - 执行如下 `curl 'http://jmenv.tbsite.net:8080/diamond-server/diamond-unit-spas'` 命令，返回 diamond spas 单元的服务器列表，ping 每台机器，确保延迟在 10 ms 以下。

4.5.2 Diamond Server 的巡检

- 磁盘空间检查，避免磁盘满导致 Diamond 服务异常：**df -lh**。
- 确保系统内存，CPU 在正常范围：**top**。
- 确保地址服务器正常连接，且不能延迟太大（内网 10 ms 以下）：**ping jmenv.tbsite.net -c 3**。
- 确保每个 Diamond Server 彼此能相互连接，网络延迟不能太大，执行一下脚本，循环 ping 所有的 Diamond Server。
- 确保 diamond 服务压力在可控范围内，单机 QPS 极限 4000 左右，建议单机 QPS 不超过 500，load 少于 CPU 核数，CPU 不超过 20%：**tsar --nginx --load --cpu -l -i3**。

4.5.3 MQTT Console 的巡检

MQTT Console 是一个部署在 Tomcat 内部的 Web 系统，MQTT Console 和 MQ Console 是同一个容器，巡检考虑如下几方面：

- Tomcat 进程是否存在，7001 端口是否开通：**netstat -an |grep 7001**
- Nginx 进程是否存在，80 端口是否能够正常转发到 7001。
- /home/admin/logs/ons-api/*.log 中是否存在明显的异常。
- /home/admin/ons-api/logs 中是否需要清理无用的日志文件。
- /home/admin 对应磁盘是否有足够的容量。

4.5.4 MQTT CS 的巡检

- 执行命令 **ps aux|grep MqttStartup**，查看进程信息。
- 执行命令 **netstat -an|grep 1883**，检查端口 1883 是否正常打开。
- 查看 MQTT CS 相关的日志，路径为 /home/admin/logs/onsproxy/。
- 检查磁盘容量：**df -h**。
- 客户端连接之后，查看 /home/admin/logs/onsproxy/device.log 文件是否有连接记录产生。
- 客户端发送消息后，查看 /home/admin/logs/onsproxy/msg.log 文件看是否有发送记录产生。

- 客户端订阅后，如果发消息到订阅的 topic，查看/home/admin/logs/onsproxy/msg.log文件看是否有推送记录产生。

**注意：**

巡检单个 CS 节点时，测试客户端要连接这个 CS 节点的 IP 地址，不要连接 VIP，否则连接会随机飘到某一个 CS 节点上，不一定是当前节点。

4.5.5 MQTT DS的巡检

- MQTT DS 运行时是一个 Java 进程，通过 PS 命令查看进程状态是否正常运行：**ps aux|grep DispatcherAndAsyncProcessCenterServer**
- 执行命令`netstat -an|grep 9999`，检查端口 9999 是否正常打开。
- 查看 MQTT DS 相关的日志，路径为/home/admin/logs/onsproxy/。
- 检查磁盘容量：**df -h**。

4.5.6 TLog Console的巡检

- 检查Tomcat进程：**ps aux|grep tomcat**。
- 检查磁盘容量：**df -h**
- 检查访问是否正常：**curl -v 'http://localhost'**。

5 故障处理

在使用 MQTT 的过程中，难免会遇到故障。本章介绍您在运维过程中可能遇到的典型故障及如何处理这些故障。

故障分类

MQTT 目前的故障主要分为以下三类：

- 组件可用性故障
- 服务可用性故障
- 业务连续性故障

故障定位

您可以通过巡检、监控、日志及告警发现故障并进行定位。

5.1 告警处理

在微消息队列 MQTT 运维过程中，巡检系统会对 MQTT 各组件的状态及指标进行巡检和监控。而巡检和监控异常，会触发告警。本节主要介绍如何巡检及监控异常告警。

5.1.1 订阅关系过多

巡检系统通过钉钉群或者本地巡检日志报出 MQTT 订阅关系数量超限的信息，此时需要检查用户使用的持久化订阅关系的数量，人工清理不合理的订阅关系。

可能的原因

1. 用户 MQTT 程序每次上线连接时使用了随机的 clientId，但是却使用了 QoS1，CleanSession=false 的初始化参数订阅了消息，此种情况需要尽快修改为 true。
2. 用户 MQTT 程序每次上线连接时订阅的 topic 都不一样，但是却使用了 QoS1，CleanSession=false 的初始化参数订阅了消息，此种情况需要尽快修改为 true。

对系统的影响

- 单集群支持的订阅关系数量有上限，参考集群规格，超出后服务端匹配离线消息速度会受影响。
- 服务端会定期清理旧的数据，因此出现报警后需要提醒用户是否存在上述几种可能的错误用法。
- 如果持续报警说明服务端定期清理速度跟不上用户产生订阅关系的速度，需要人工及时清理，清理数据联系产品方审核变更。此操作为 **G2** 级别。

5.1.2 发送消息失败

巡检系统会定期模拟用户行为，使用 MQTT 客户端发送消息，如果发送失败会通过钉钉和本地日志报警。

可能的原因

1. MQTT CS 组件正在运维或者服务存在异常
2. MQTT 后端 RocketMQ 正在运维或者服务存在异常掉

对系统的影响

如果持续报警无法自行恢复，则会影响业务应用。

- 登陆 MQTT CS 容器节点，查看/home/admin/logs/onsproxy/mqtt_system.log
- 搜索 mqtt_system.log 中的异常信息，提交产品开发确认原因。
- 检查 RocketMQ 巡检是否有异常，以便排除是否受 RocketMQ 的影响。

5.1.3 离线消息太多

巡检系统通过钉钉群或者本地巡检日志报出 MQTT 离线消息数量超限的信息，此时需要检查用户使用的离线消息的数量，并人工清理不合理的离线消息。

可能的原因

- 用户 MQTT 程序每次上线连接时使用了随机的 clientId，但是却使用了 QoS1，CleanSession=false 的初始化参数订阅了消息，此种情况需要尽快修改为 true。
- 用户 MQTT 程序每次上线连接时订阅的 topic 都不一样，但是却使用了 QoS1，CleanSession=false 的初始化参数订阅了消息，此种情况需要尽快修改为 true。

对系统的影响

单集群支持的离线消息数量有上限，参考集群规格，超出后会影响离线消息生成和读取速度。

- 服务端会定期清理旧的数据，因此出现报警后需要提醒用户是否存在上述几种可能的错误用法。
- 如果持续报警说明服务端定期清理速度跟不上用户产生离线消息的速度，需要人工及时清理，清理数据联系产品方审核变更。此操作为 G2 级别。

5.1.4 离线消息匹配延迟

巡检系统会监控 MQTT 匹配离线消息的延迟和堆积数，如果出现延迟过大或者堆积过多都会通过钉钉群或者本地巡检日志报警。

离线消息匹配延迟的监控包括：

- 离线消息匹配延迟时间：代表目前生成离线消息的时间戳，也代表用户上线读取离线消息的延迟。

- 离线消息匹配堆积量：代表当前还有多少消息没有匹配出离线消息。

可能的原因

- 数据库存在异常
- 用户存在太多订阅关系

对系统的影响

影响离线消息生成速度，如果持续增加需要人工介入处理。

1. 登陆 MQTT 的数据库，查看 mqtt_sub 表的记录数是否太大。
2. 查看 MQTT 的数据库监控，是否存在性能异常。

5.1.5 连接数太多

MQTT 巡检会监控所有 MQTT CS 节点当前的连接数，当超过预定的报警阈值后会报警，需要和用户评估当前的用量和后期增长规划。

可能的原因

- 报警阈值太低
- 用户程序异常，制造了大量僵尸连接。
- 用户业务增长。

对系统的影响

单机连接数超限会影响客户端连接稳定性，如果报警阈值合理，则需要确认上述原因，是否需要查出业务异常代码或者服务端进行扩容。此操作为 **G3** 级别。

5.1.6 消息接收延迟

巡检系统会定期模拟用户行为，使用 MQTT 客户端接收消息，如果消息在指定的时间窗口内都没有收到，则会报警提示消息接收可能存在延迟。

可能的原因

- MQTT CS 组件正在运维或者服务存在异常
- MQTT 后端 RocketMQ 正在运维或者服务存在异常

对系统的影响

如果持续报警无法自行恢复，则会影响业务应用。

- 登陆 MQTT CS 容器节点，查看/home/admin/logs/onsproxy/proxy_stat.log
- 搜索proxy_stat.log中的 **FAST_DELAY** 字段，提取出相关日志给产品开发，确认可能的原因。
- 检查 RocketMQ 巡检是否有异常，以便排除是否受 RocketMQ 的影响。

5.1.7 消息收发 TPS 过高

巡检系统会监控 MQTT CS 节点的上行和下行 TPS，如果消息 TPS 超过了预定的报警阈值，则通过钉钉群或者本地巡检日志报出该问题。

可能的原因

- 报警阈值设置太低
- 用户存在突发流量
- 用户业务量上涨

对系统的影响

MQTT 服务单机能够提供的消息 TPS 是有上限的，如果单机超过容量则会出现消息收发延迟，影响业务。

1. 如果是报警阈值不合理，调整报警阈值。
2. 如果是用户存在突发流量或者本身业务量上涨，则需要进行扩容。此操作为 **G3** 级别。

5.2 业务问题处理

5.2.1 反馈连接不上 MQTT 服务

本文介绍反馈连接不上 MQTT 服务。

现象

用户 MQTT 客户端程序连接服务端不成功，connect 方法卡住没有返回，或者返回超时，或者调用后即抛异常退出。

可能的原因

1. 客户端到 MQTT 服务端网络不通。
2. 客户端参数错误导致权限验证不通过。

处理方案

1. 如果此时巡检系统没有报警，则基本判定为用户程序问题或者环境问题。
2. 如果 connect 卡住或者没有返回，则需要检查用户所在的网络和 MQTT 服务端的网络连通性。
3. 如果 connect 返回异常，则一般是权限验证不通过，需要登上 MQTT CS 节点根据 clientId 查询设备轨迹日志，确认异常原因，一般是签名错误，AK、SK 不匹配造成。具体方法如下。

查看客户端异常日志

登录 MQTT CS 容器节点，根据 clientId 搜索 /home/admin/logs/onsproxy/onsproxy.log，查看是否有权限异常。再结合 /home/admin/logs/onsproxy/authManager.log 即可判断原因，示例如下：

```
cat onsproxy.log | grep GID_yueliao_mqtt@001117
2019-08-07 20:23:49 ERROR worker-io-10-thread-45 post-cn-45918471z07n6GID_yueliao_mqtt@001117*Signature|LTAlWtbaboZAugEG |post-cn-45918471z07n6GID_yueliao_mqtt@001117 clientIp:100.101.197.121 NOT_AUTHORIZED
2019-08-07 20:25:00 ERROR worker-io-10-thread-104 post-cn-45918471z07n6GID_yueliao_mqtt@001117*Signature|LTAlWtbaboZAugEG |post-cn-45918471z07n6GID_yueliao_mqtt@001117 clientIp:100.101.197.17 NOT_AUTHORIZED
2019-08-07 20:25:41 ERROR worker-io-10-thread-52 post-cn-45918471z07n6GID_yueliao_mqtt@001117*Signature|LTAlWtbaboZAugEG |post-cn-45918471z07n6GID_yueliao_mqtt@001117 clientIp:100.101.198.225 NOT_AUTHORIZED
2019-08-07 20:26:12 ERROR worker-io-10-thread-126 post-cn-45918471z07n6GID_yueliao_mqtt@001117*Signature|LTAlWtbaboZAugEG |post-cn-45918471z07n6GID_yueliao_mqtt@001117 clientIp:100.101.198.147 NOT_AUTHORIZED
2019-08-07 20:26:22 ERROR worker-io-10-thread-17 post-cn-45918471z07n6GID_yueliao_mqtt@001117*Signature|LTAlWtbaboZAugEG |post-cn-45918471z07n6GID_yueliao_mqtt@001117 clientIp:100.101.199.17 NOT_AUTHORIZED
```

5.2.2 反馈连接经常断开

现象

用户反馈 MQTT 客户端连接经常被断开，或者是只要发送消息订阅消息就会被断开

可能的原因

1. MQTT 客户端使用了相同的 clientId 起了多个程序或者连接，相互影响。
2. MQTT 客户端订阅或者发送的 Topic 没有权限导致权限异常。

处理方案

1. 登录 MQTT CS 容器节点，根据 clientId 搜索 /home/admin/logs/onsproxy/device.log，查看 clientId 的连接轨迹，如果出现了 clientIDConflict 等关键字，则说明是客户端使用了相同的 clientId 起了多个程序或者连接导致的相互影响。
2. 如果出现了 Auth Failed 关键字，则说明是客户端操作的 topic 没有权限，需要检查该 Topic 是否在控制台创建，或者是否已经授权。

```
[admin@ons-mqtt-cs0121901010 /home/admin/logs/onsproxy]
cat device.log | grep GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f --color
2019-08-07 20:17:05.951 [GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f]##connect|##CONNECTION_ACCEPTED,clearSession=true
2019-08-07 20:17:05.981 [GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f]##close|##Auth Failed, Sub auth failed,clientId:GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789fProxyAuthData(channel="ALIVIN", authData=AuthData(us
erName="LTAlT50e2qkCVm", password="HaE2pCH5WpKvPD7NgvUTbABxI="), requestCode=11, resource="DML_PRO_MEITUAN_MQTT/mtOrder/5213_1012", token="{}", actionCodes=[4], clientId="GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f", owner
Id=null) failed
2019-08-07 20:18:16.336 [GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f]##connect|##CONNECTION_ACCEPTED,clearSession=true
2019-08-07 20:18:16.357 [GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f]##close|##Auth Failed, Sub auth failed,clientId:GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789fProxyAuthData(channel="ALIVIN", authData=AuthData(us
erName="LTAlT50e2qkCVm", password="HaE2pCH5WpKvPD7NgvUTbABxI="), requestCode=11, resource="DML_PRO_MEITUAN_MQTT/mtOrder/5213_1012", token="{}", actionCodes=[4], clientId="GID_MEITUANPRO000a6ed872b-f258-4faa-af0d-4942ede7789f", owner
Id=null) failed
```

5.2.3 反馈收不到消息

现象

MQTT 客户端连接成功，且订阅完成，本地程序无异常，但是收不到下推的消息。

可能的原因

1. 服务端消息存储异常。
2. 客户端订阅不成功。
3. 业务上没有发送消息或者消息发送的 topic 和订阅的不匹配。

处理方案

1. 首先检查此时的巡检监控，是否有消息收发异常，如果没有报警则基本判定为业务逻辑问题。

2. 登陆 MQTT CS 容器节点查询设备轨迹日志，确认客户端订阅的 topic 是否正确。设备轨迹查询方法参考运维命令章节。
3. 如果订阅是成功的，则需要查询是否有消息发送成功，且消息发送的 topic 和订阅的 topic 是否匹配。消息发送的轨迹查询参考消息轨迹查询章节。

6 运维命令

后端运维命令是提供给有经验的运维工程师使用的，请优先选择使用控制台执行相关的命令。

6.1 概述

微消息队列 MQTT 提供了一些运维管控 HTTP API 接口，用于排查日常问题，实际使用时可以登录到所有的 MQTT 机器上依次执行，或者使用批量脚本批量执行 curl 命令发出 HTTP 请求，即可汇总得到查询结果。



注意：

由于所有服务节点的 API 接口仅汇总当前节点的状态，因此如果要查询完整的状态需要将所有机器都执行一遍才能汇总集群内的完整情况。

举例：根据 clientId 查询一个设备的在线连接，该设备当前连接到容器节点 1，此时查询容器节点 2 则没有数据，需要同时查询所有的容器节点才能获取准确数据。

6.2 查看客户端是否在线

本节介绍如何查看客户端是否在线。

背景信息

登陆所有 MQTT CS 容器节点，依次执行以下命令获取指定 clientId 的在线连接数据。

```
curl -s "http://127.0.0.1:9999/csChannel/csGetDeviceRunningInfo?clientId=XXX&namespace=XXX"
```

其中 XXX 更换为实际查询的设备的 clientId 和实例 Id。

查询结果为 JSON 数据，可以得到当前客户端是否在线，如果在线同时展示客户端当前的订阅关系数据。

6.3 查看客户端连接轨迹

本文介绍如何查看客户端连接轨迹。

客户端连接轨迹是根据客户端 clientId 和时间范围搜索客户端的连接上下线行为，用于判断是否有异常。查询时根据 clientId 搜索所有 MQTT CS 容器的轨迹日志。

1. 首先根据时间范围确认日志文件，最新的日志文件位于 /home/admin/logs/onsproxy/device.log 历史日志文件位于 /home/admin/logs/onsproxy/otherdays/device.xx.log.gz
2. 确认文件范围后根据 clientId 搜索对应文件即可。

图 6-1: 设备轨迹文件示例

```
$fgrep GID-YHBC@@_1020464_035947_shop_client device.log
2019-08-06 20:04:12.182 |##GID-YHBC@@_1020464_035947_shop_client|##connect|##CONNECTION_ACCEPTED, cleanSession=false
2019-08-06 20:04:12.228 |##GID-YHBC@@_1020464_035947_shop_client|##subscribe|##msg_yhbc/1020464_shop_h5/, AT_MOST_ONCE;
2019-08-06 20:04:17.087 |##GID-YHBC@@_1020464_035947_shop_client|##close|##be closed
```

如上图所示，根据 clientId 即可搜索得到一个设备的执行过程，首先 connect 成功，然后 subscribe 订阅了 topic，最后设备端主动断连。



注意：

如果根据 clientId 查询轨迹，需要将所有 MQTT CS 容器都查询一次才能得到完整的结果。

6.4 查询某个 topic 在线订阅的客户端

登陆所有 MQTT CS 容器节点，依次执行以下命令即可根据 topic 获取订阅该 topic 的所有客户端在线数据。

```
curl -s "http://127.0.0.1:9999/csTopic/csGetClientInfoByTopic?firstTopic=XXX&secondTopic=XXX&namespace=XXX"
```

其中 XXX 更换为实际查询的一级 topic，子级 topic 和实例 Id，如果订阅关系没有子级 topic 则 secondTopic 参数可以不传。

查询结果为 JSON 数据，可以得到当前符合订阅的在线客户端的数量。

6.5 查询某个分组下客户端数量

登陆所有 MQTT CS 容器节点，依次执行以下命令获取指定 GroupId 的在线客户端连接数据。

```
curl -s "http://127.0.0.1:9999/csChannel/csGetClientInfoByGroupId?groupId=XXX&namespace=XXX"
```

其中 XXX 更换为实际查询的设备的 groupId 和实例 Id。

查询结果为 JSON 数据，可以得到当前在线的客户端数量。

6.6 查询某条消息收发轨迹

消息收发轨迹是根据客户端 `clientId` 或者 `msgId` 查询对应的消息的收发轨迹，用于确认消息是否收发正常。查询时根据 `clientId` 搜索所有 MQTT CS 容器的轨迹日志。

查询轨迹日志文件的步骤如下：

1. 根据时间范围，确认日志文件



说明：

- 最新的日志文件位于 `/home/admin/logs/onsproxy/msg.log`。
- 历史日志文件位于 `/home/admin/logs/onsproxy/otherdays/msg.xx.log.gz`。

2. 根据查询条件，搜索对应文件。



注意：

如果查询消息轨迹，需要将所有 MQTT CS 容器都查询一次才能得到完整的结果。

MQTT 客户端发消息 MQTT客户端收消息场景

该场景查询需要以下信息：

- 发送方的 `clientId`
- 发送的时间范围。

1. 根据 clientId 和时间范围查询符合条件的 msgId 集合。

```
cat trace.log | grep clientId | grep pub_mqtt
```

```

kcat msg.log | grep GID_HK_hzyhbc@@11.193.54.7_S_V2471 | grep pub_mqtt
2019-08-06 18:47:51.077 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86023D2E1|##pub_mqtt|##success
2019-08-06 18:47:51.084 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86029D2E2|##pub_mqtt|##success
2019-08-06 18:47:51.101 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC8603CD2E5|##pub_mqtt|##success
2019-08-06 18:47:51.124 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86051D2EA|##pub_mqtt|##success
2019-08-06 18:47:51.129 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86058D2EF|##pub_mqtt|##success
2019-08-06 18:47:51.148 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC8606BD2F5|##pub_mqtt|##success
2019-08-06 19:04:31.267 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A323C2E6|##pub_mqtt|##success
2019-08-06 19:04:31.278 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A32DC2EA|##pub_mqtt|##success
2019-08-06 19:04:31.304 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A345C2F0|##pub_mqtt|##success
2019-08-06 19:04:31.335 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A366C2F2|##pub_mqtt|##success
2019-08-06 19:04:31.354 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A378C2F4|##pub_mqtt|##success
2019-08-06 19:04:31.374 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A38CC2F7|##pub_mqtt|##success
2019-08-06 19:21:11.391 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E5DEB0E8|##pub_mqtt|##success
2019-08-06 19:21:11.399 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E5E6B0E9|##pub_mqtt|##success
2019-08-06 19:21:11.421 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E5FCB0EA|##pub_mqtt|##success
2019-08-06 19:21:11.450 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E61AB0EF|##pub_mqtt|##success
2019-08-06 19:21:11.461 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E623B0F0|##pub_mqtt|##success
2019-08-06 19:21:11.483 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E63AB0F2|##pub_mqtt|##success
2019-08-06 19:37:51.384 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628169C06|##pub_mqtt|##success
2019-08-06 19:37:51.392 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF6281F9C07|##pub_mqtt|##success
2019-08-06 19:37:51.415 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628349C08|##pub_mqtt|##success
2019-08-06 19:37:51.444 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628539C0C|##pub_mqtt|##success
2019-08-06 19:37:51.453 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF6285C9C0F|##pub_mqtt|##success
2019-08-06 19:37:51.475 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628709C12|##pub_mqtt|##success
2019-08-06 19:54:31.440 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056A8E7E8D|##pub_mqtt|##success
2019-08-06 19:54:31.449 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056A967E8E|##pub_mqtt|##success
2019-08-06 19:54:31.467 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056AAA7E92|##pub_mqtt|##success
2019-08-06 19:54:31.493 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056AC47E95|##pub_mqtt|##success
2019-08-06 19:54:31.500 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056ACC7E97|##pub_mqtt|##success
2019-08-06 19:54:31.519 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056ADE7E98|##pub_mqtt|##success
2019-08-06 20:11:11.687 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14ADC45453|##pub_mqtt|##success
2019-08-06 20:11:11.695 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14ADC5458|##pub_mqtt|##success
2019-08-06 20:11:11.715 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14ADE2545B|##pub_mqtt|##success
2019-08-06 20:11:11.777 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14AE1F54AA|##pub_mqtt|##success
2019-08-06 20:11:11.786 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14AE2754AC|##pub_mqtt|##success
2019-08-06 20:11:11.805 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14AE3B54B2|##pub_mqtt|##success

```

2. 得到 msgId 集合后到 RocketMQ 控制台根据 msgId 查询消息，确认哪条消息是实际关心的消息，筛选出 msgId。
3. 根据 msgId 再搜索 msg.log，得到推送成功的客户端日志。

```
cat msg.log | grep xxxx. | grep sub
```

```

kcat msg.log | grep 0BC13607473F4AA298B71DC86023D2E1 | grep sub
2019-08-06 18:47:51.081 |##GID_HK_hzyhbc@@11.193.54.7_R_V2,|##0BC13607473F4AA298B71DC86023D2E1|##sub|##

```

MQTT 客户端发消息 RocketMQ 客户端收消息场景

该场景查询需要以下信息：

- 发送方的 clientId
- 发送的时间范围。

1. 根据 clientId 和时间范围查询符合条件的 msgId 集合。

cat trace.log | grep clientId | grep pub_mqtt

```
!cat msg.log | grep GID_HK_hzyhbc@@11.193.54.7_S_V2471 | grep pub_mqtt
2019-08-06 18:47:51.077 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86023D2E1|##pub_mqtt|##success
2019-08-06 18:47:51.084 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86029D2E2|##pub_mqtt|##success
2019-08-06 18:47:51.101 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC8603CD2E5|##pub_mqtt|##success
2019-08-06 18:47:51.124 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86051D2EA|##pub_mqtt|##success
2019-08-06 18:47:51.129 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC86058D2EF|##pub_mqtt|##success
2019-08-06 18:47:51.148 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DC8606BD2F5|##pub_mqtt|##success
2019-08-06 19:04:31.267 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A323C2E6|##pub_mqtt|##success
2019-08-06 19:04:31.278 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A32D2EA|##pub_mqtt|##success
2019-08-06 19:04:31.304 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A345C2F0|##pub_mqtt|##success
2019-08-06 19:04:31.335 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A366C2F2|##pub_mqtt|##success
2019-08-06 19:04:31.354 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A378C2F4|##pub_mqtt|##success
2019-08-06 19:04:31.374 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DD7A38C2F7|##pub_mqtt|##success
2019-08-06 19:21:11.391 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E5DEB0E8|##pub_mqtt|##success
2019-08-06 19:21:11.399 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E5E6B0E9|##pub_mqtt|##success
2019-08-06 19:21:11.421 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E5FCB0EA|##pub_mqtt|##success
2019-08-06 19:21:11.450 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E61AB0EF|##pub_mqtt|##success
2019-08-06 19:21:11.461 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E623B0F0|##pub_mqtt|##success
2019-08-06 19:21:11.483 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DE6E63AB0F2|##pub_mqtt|##success
2019-08-06 19:37:51.384 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628169C06|##pub_mqtt|##success
2019-08-06 19:37:51.392 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF6281F9C07|##pub_mqtt|##success
2019-08-06 19:37:51.415 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628349C08|##pub_mqtt|##success
2019-08-06 19:37:51.444 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628539C0C|##pub_mqtt|##success
2019-08-06 19:37:51.453 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF6285C9C0F|##pub_mqtt|##success
2019-08-06 19:37:51.475 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71DF628709C12|##pub_mqtt|##success
2019-08-06 19:54:31.440 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056A8E7E8D|##pub_mqtt|##success
2019-08-06 19:54:31.449 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056A967E8E|##pub_mqtt|##success
2019-08-06 19:54:31.467 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056AAE7E92|##pub_mqtt|##success
2019-08-06 19:54:31.493 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056AC47E95|##pub_mqtt|##success
2019-08-06 19:54:31.500 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056ACC7E97|##pub_mqtt|##success
2019-08-06 19:54:31.519 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E056ADE7E98|##pub_mqtt|##success
2019-08-06 20:11:11.687 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14ADC45453|##pub_mqtt|##success
2019-08-06 20:11:11.695 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14ADC5458|##pub_mqtt|##success
2019-08-06 20:11:11.715 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14ADE2545B|##pub_mqtt|##success
2019-08-06 20:11:11.777 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14AE1F54AA|##pub_mqtt|##success
2019-08-06 20:11:11.786 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14AE2754AC|##pub_mqtt|##success
2019-08-06 20:11:11.805 |##GID_HK_hzyhbc@@11.193.54.7_S_V2471|##0BC13607473F4AA298B71E14AE3B54B2|##pub_mqtt|##success
```

2. 得到 msgId 集合后到 RocketMQ 控制台根据 msgId 查询消息，确认哪条消息是实际关心的消息，筛选出 msgId。使用 RocketMQ 的消息轨迹功能确认该消息是否被 RocketMQ 的某个订阅组处理成功。

RocketMQ 客户端发消息 MQTT 客户端收消息场景

该场景查询需要以下信息：

- RocketMQ 客户端发送成功后得到的 msgId。

根据指定的 msgId 直接搜索消费的轨迹日志即可判断 MQTT 客户端是否消费该消息。

cat msg.log | grep xxxxx. | grep sub

```
!cat msg.log | grep 0BC13607473F4AA298B71DC86023D2E1 | grep sub
!019-08-06 18:47:51.081 |##GID_HK_hzyhbc@@11.193.54.7_R_V2,|##0BC13607473F4AA298B71DC86023D2E1|##sub|##
```

7 容量评估

7.1 MQTT CS组件

MQTT CS 是维持客户端长连接以及进行消息收发的核心组件，CS 组件的规模和业务规模强相关，评估 CS 组件的容量主要从两个指标，连接数和 TPS。

连接数指标

- **当前水位统计：**

进入 mq-mqtt-cs 容器执行 `ss -s`

- **基准和警戒水位：**

当单机连接数超过规格半数以上时，请提工单申请处理扩容。

TPS 指标

- **当前水位监控：**

进入 mq-mqtt-cs 容器，查看运行统计日志/home/admin/logs/onsproxy/commercial_stat.log，该日志按照分钟汇总所有用户的 TPS 总和，按照最近的 1 分钟关键字搜索 TPS 字段，确认 TPS。

- **基准和警戒水位：**

目前 MQTT CS 节点单机上行和下行 TPS 总和规格半数以上时，请提工单申请处理扩容。

附录：计算 TPS 的方法

```
[admin@onsmqttcs011193099233.cm12 /home/admin/logs/onsproxy]
$date
Mon Aug 5 20:43:02 CST 2019

[admin@onsmqttcs011193099233.cm12 /home/admin/logs/onsproxy]
$cat commercial_stat.log | grep "2019-08-05 20:42" | grep TPS
```

第一步，看当前时间

第二步搜索前一分钟的 TPS 汇总

```
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1615860008861890|12,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1159520133865134|4,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1752368399225346|6,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1030521292969044|4,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,post-cn-45913ndtd02%1215616266703710|6,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1331574342750496|110,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1237256220760583|2,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,post-HK-bjpost01%1080056302921134|592,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1301225312146762|422,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1684967696495902|17667,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1735694812357719|3721,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1420787220309929|2,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1080056302921134|624,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,post-cn-v0h10cjix05%1293505463152644|152,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1651344197670905|45,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,50068491|7,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,post-cn-v0h10k0kf0h%1184846914103586|877,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,31168686|2,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,1721706406530105|2,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,post-cn-v0h11yjer02%1157922787518210|2,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,31882848|4,0
2019-08-05 20:42:00|1|COMM_STAT_TPS,bjpost01,31662787|16,0
```

取|后面的第一个数字

根据当前时间，搜索前一分钟的日志，再搜索关键字 TPS 即可得到前一分钟所有用户的 TPS 汇总数据，在取方框中指出的 "|" 后的数字，就是这个用户一分钟的消息收发数量，所有用户的加和即可得到单机分钟消息收发总数。

7.2 磁盘使用率

MQTT 服务是无状态网关服务，所以本地不记录业务数据状态，磁盘使用仅限日志文件，因此当出现磁盘空间不足时可能需要人工介入清理日志文件，相关日志文件描述参考[日志参考](#)。

7.3 数据库 组件

MQTT 使用数据库存储持久化订阅关系以及客户端的离线消息，正常情况下服务端会对历史的没有更新的非活跃订阅关系和离线消息进行滚动删除，但如果线上出现异常业务场景需要巡检数据库各表的容量，对超限的数据进行清理，确保单表容量。

订阅关系：

- 当前水位统计：

登录 MQTT 数据库，执行命令：

```
select count(*) from mqtt_sub;
```

离线消息：

- **当前水位统计：**

登录 MQTT 数据库，执行命令：

```
select count(*) from msg_push_status;
```

8 日志参考

在运维过程中，您可以通过日志了解 MQTT 各组件的状态或定位故障。本文将罗列出 MQTT 组件的所有日志的路径，并提供相关说明。

组件	文件	说明
mqtt-cs	/home/admin/logs/onsproxy/onsproxy.log	MQTT 连接节点所有交互日志 - 包含客户端触发的各种异常日志 - 一般用于确认客户端是否有连接的痕迹和错误日志 - 搜索关键字为您的 clientId 或者 topic
mqtt-cs	/home/admin/logs/onsproxy/authManager.log	MQTT 连接节点鉴权日志 - 包含客户端触发的权限异常日志 - 一般用于排查客户端是否使用了错误的 topic 等权限错误 - 搜索关键字为用户使用的 clientId 或者 topic
mqtt-cs	/home/admin/logs/onsproxy/mqtt_system.log	MQTT 连接节点后端日志 - 包含 MQTT 服务内部的调用异常日志 - 用于确认是否服务端内部出现异常
mqtt-cs	/home/admin/logs/onsproxy/device.log	MQTT 客户端连接日志 - 用于查询一个客户端的所有行为日志 - 搜索关键字为用户使用的 clientId
mqtt-cs	/home/admin/logs/onsproxy/msg.log	MQTT 客户端消息日志 - 用于查询一个客户端所有的消息收发日志 - 搜索关键字为用户使用的 clientId

9 更新 MQTT 配置

MQTT CS 组件和 DS 组件内置管控 HTTP API，修改配置需要调用内置的管控 API 实现，通常需要登陆到对应的容器内部，通过 curl 命令调用 API 接口实现参数的动态修改，此操作为 **G3** 级别。

修改 MQTT CS 的配置

1. 登录所有的 MQTT CS 容器。
2. 在每个容器内部都执行如下命令：

```
curl -s "http://127.0.0.1:9999/csConfig/csUpdateMqttConfig?key={KEY}&value={VALUE}"
```

其中 **{KEY}** 更换为实际需要修改的配置项名称，**{VALUE}** 更换为实际调整的参数值。

修改 MQTT DS 的配置

1. 登录所有的 MQTT DS 容器。
2. 在每个容器内部都执行如下命令：

```
curl -s "http://127.0.0.1:9999/dsConfig/dsUpdateMqttConfig?key={KEY}&value={VALUE}"
```

其中 **{KEY}** 更换为实际需要修改的配置项名称，**{VALUE}** 更换为实际调整的参数值。