消息队列 MQTT 版

快速入门

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

快速入门
创建资源

最近更新时间：2025-01-02 17:05:32

操作场景

在使 SDK 收发消息前，您需要在 MQTT 控制台中创建集群、Topic 等资源，运行客户端时需要配置相关的资源信息。

创建集群

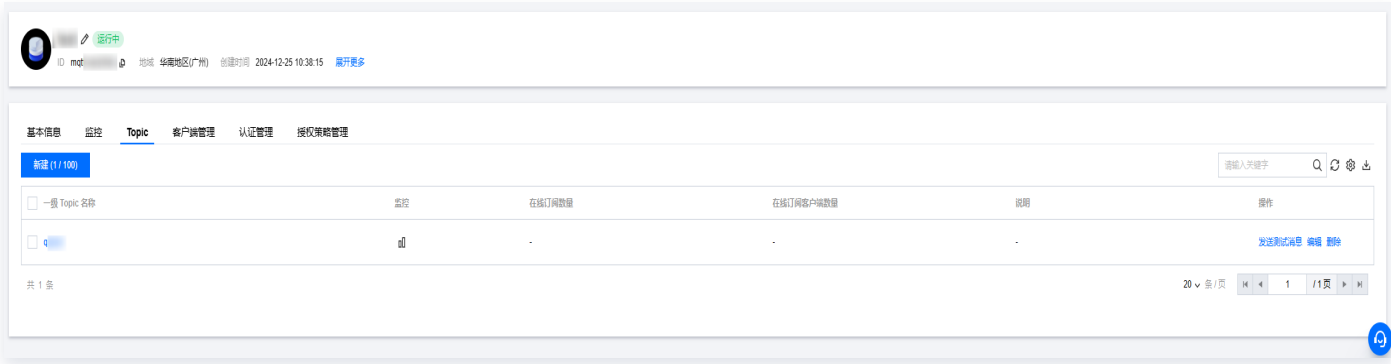
1. 登录 MQTT 控制台 。
2. 在左侧导航栏选择 **资源管理** > **集群管理**，选择好地域后，单击**新建集群**进入集群购买页面。

配置	说明
计费模式	当前支持包年包月和按量计费两种模式。
地域	选择与您的业务最靠近的地域，处于不同地域的云产品内网不通，购买后不能更换，请您谨慎选择。例如，广州地域的云服务器无法通过内网访问上海地域的集群。若需要跨地域内网通信，请查阅 云联网 。
集群规格	当前提供了基础版和专业版两种版本规格，两个版本之间的差异可以查看 产品系列 。
TPS 规格	包含生产消息和消费消息的总和，按照设置不同对单条消息进行折算，详细规则见 购买说明 。
客户端连接数	基础版最高支持 2000 个客户端连接数，不支持额外购买连接数；专业版根据 TPS 规格不同赠送的免费连接数不同，可单独额外购买连接数。
私有网络	授权将新购集群接入点域名绑定至私有网络（VPC）。
公网	开启公网带宽后会新增单独的费用，开通后可在集群管理页关闭。计费价格参见 公网计费说明 。
集群名称	填写集群名称， 长度不能超过64个字符，只能包含数字、字母、“-”和“_”。
标签	标签用于从不同维度对资源分类管理。

3. 选好配置单击**立即购买**，等待3～5分钟，即可在控制台看到完成创建的集群。

创建一级 Topic

1. 在**集群管理**页面，单击刚刚创建的集群ID，进入集群基本信息页面。
2. 在顶部页签选择 **Topic**，单击**新建**，填写好 Topic 名称和说明。
3. 单击**提交**，在 Topic 列表中即可看见创建好的 Topic。



说明：

一级Topic 主要为了帮助客户进行业务的区分和元数据的管理，后续的订阅关系查阅和消息查询都可以根据一级 Topic 来查询。

客户端在发送消息时，需要先新建一级 Topic，否则会出现报错，即如果您需要往 “TopicA/device” 发送消息，您需要先创建一级 Topic “TopicA”，后续可以往类似“TopicA/+”,"TopicA/#" 等多层级 Topic 发送消息。

如果当前业务使用时强依赖 Topic 的自动创建能力，并且数量超过当前规格限制可以通过工单 联系我们 。

新建用户

1. 在目标集群详情页，选择**认证管理**，进入**用户名和密码**页签。
2. 单击**新建用户**，填写用户名和说明，设置密码。用户名（username）和密码（password）是 MQTT 提供的最基本的认证方式，后续使用客户端收发消息时需要填写。
   - 用户名：最长为32个字符，支持数字、大小写字母和分隔符（"_","-"）。

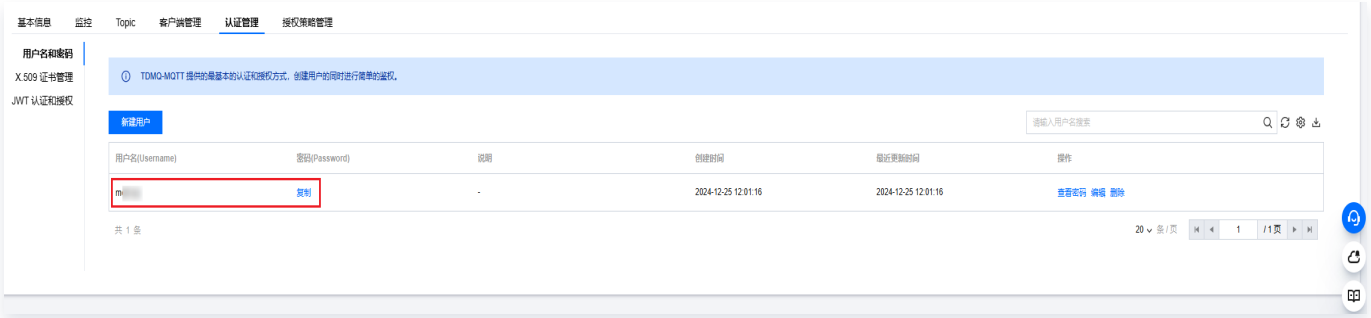
- 设置密码：支持系统自动生成密码或者自定义设置密码。
- 说明（选填）：不得超过128个字符。

3. 单击**保存**，完成用户创建，后续在权限管理列表中，可以通过以下任一种方式复制用户名和密码。

> **注意：**
> 密码泄露很可能导致您的数据泄露，请妥善保管您的密码。

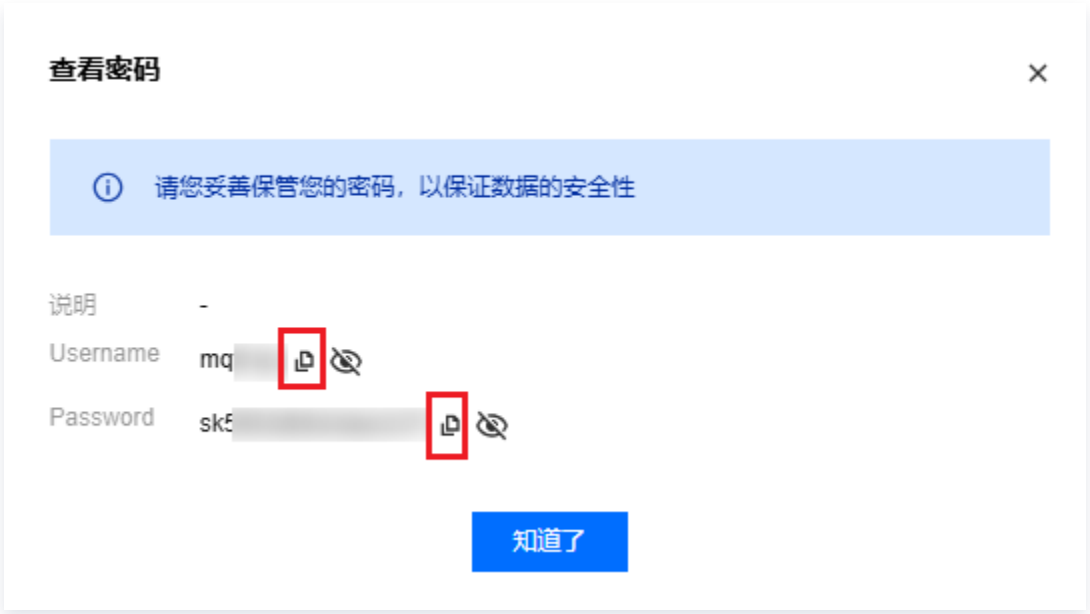
方式一：密钥列复制

在用户名(Username)、密码(Password)列复制。



方式二：操作列查看并复制

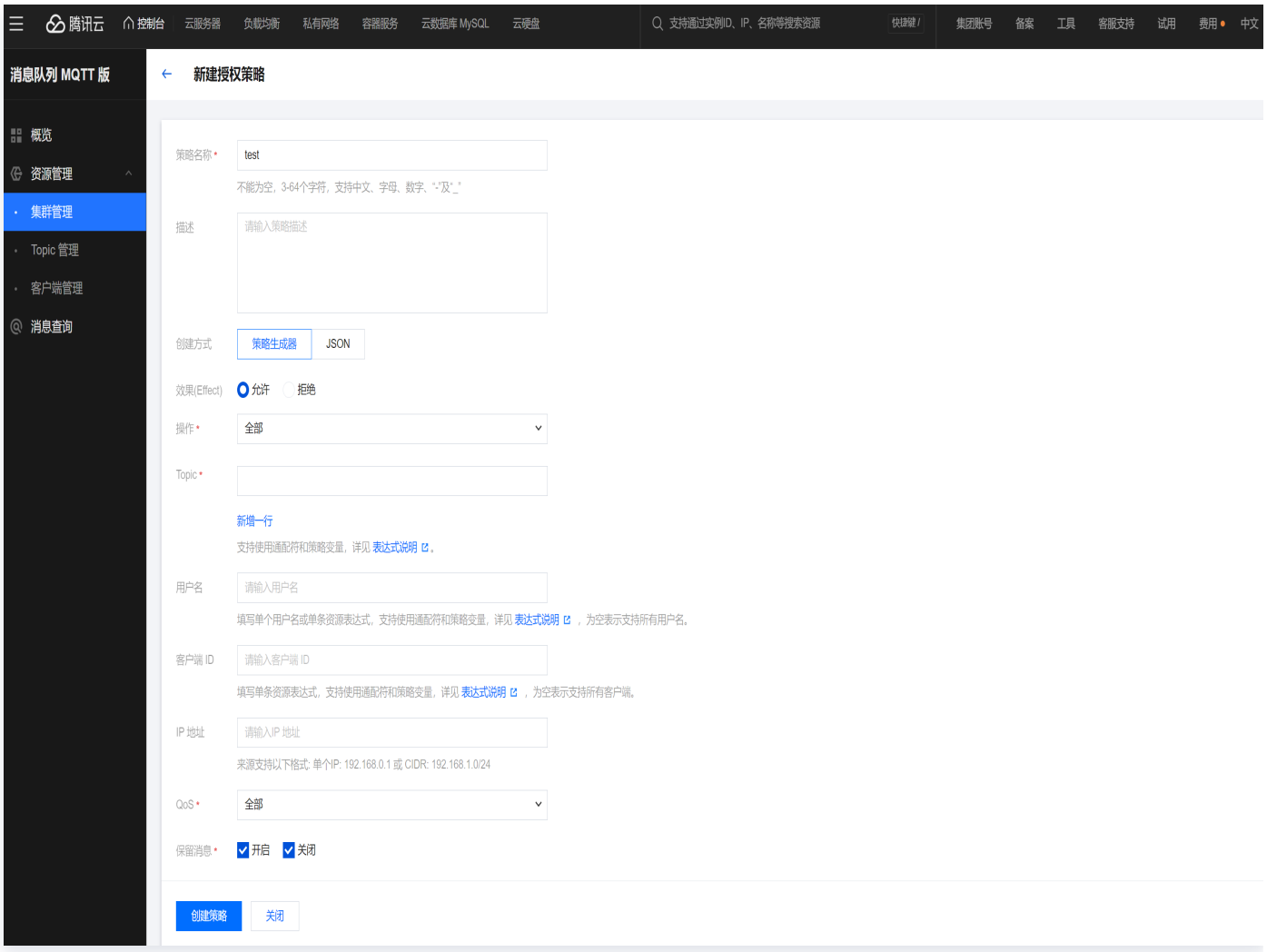
单击操作列的**查看密码**，在查看密码弹框中单击复制图标。



配置权限

创建完成角色后，需要为创建好的角色设置生产消息和消费消息的策略，详情见 数据面授权策略说明 。如果不设置授权策略，默认所有的请求均无权限，请求会被拒绝。

1. 在目标集群详情页，选择**授权策略管理**，单击**新建策略**。
2. 在新建授权策略页面，填写策略相关信息并配置好生产消费权限。授权策略同时支持可视化的策略配置和 JSON 文件配置。数据面授权策略填写详情请参考 数据面授权策略说明 。



设置项	说明
策略名称	3-64个字符，支持中文、字母、数字、“-”及“_”。
描述	选填，不得超过128个字符。
创建方式	同时支持可视化的策略配置和 JSON 文件配置。
效果（Effect）	“允许” 或者 “拒绝” 二者选一，如选择 “允许”，则表示满足配置的以下条件时，客户端的操作可以进行，如果选择 “拒绝”，即满足配置的以下条件时，客户端的操作将被拒绝。
操作	授权策略针对的不同的请求，包括连接（Connect），发送消息（Publish）和 订阅消息（Subscribe），支持多选。
Topic	支持使用通配符和策略变量，详见 表达式说明 ，

用户名	选填，填写单个用户名或单条资源表达式，支持使用通配符和策略变量，详见 表达式说明，为空表示支持所有用户名。
客户端 ID	选填，填写单条资源表达式，支持使用通配符和策略变量，详见 表达式说明，为空或者 * 表示支持所有客户端。
IP 地址	选填，仅支持填写单个 IP（如 192.168.0.1）或 CIDR 格式（如 192.168.1.0/24）。
QoS	选择授权策略支持的 QoS 等级。
保留消息	选择授权策略是否支持保留消息（retain message）。

在快速开始阶段，我们可以简单写一条规则如下，表示允许所有的客户端进行访问：

```
{
    "effect": "allow",
    "actions": [
        "connect",
        "pub",
        "sub"
    ],
    "topics": [
        "*"
    ],
    "condition": {
        "ip": "0.0.0.0/0",
        "clientId": "",
        "username": "",
        "qos": [
            0,
            1,
            2
        ],
        "retain": [
            "true",
            "false"
        ]
    }
}
```

使用 SDK 收发消息

最近更新时间：2024-12-27 14:25:13

操作场景

本文以使用 Paho Java Client 为例介绍通过开源终端 SDK 实现消息收发的操作过程，帮助您更好地理解消息收发的完整过程。

前提条件

- 完成资源创建与准备 。
- 安装1.8或以上版本 JDK
- 安装2.5或以上版本 Maven
- 下载 Demo

操作步骤

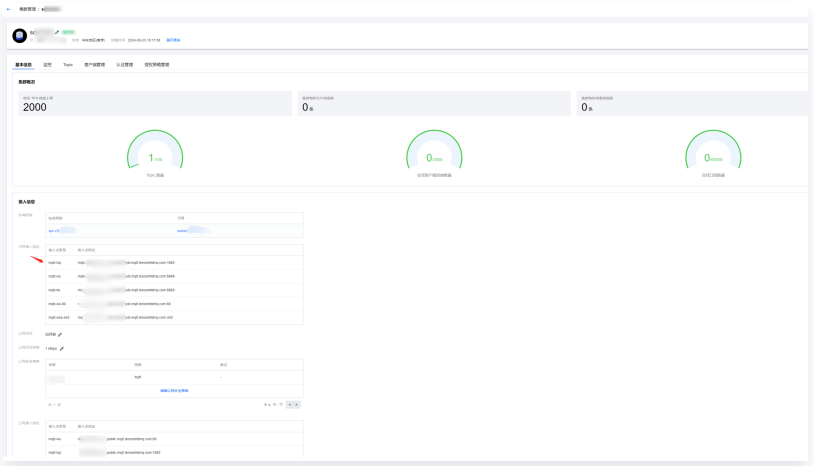
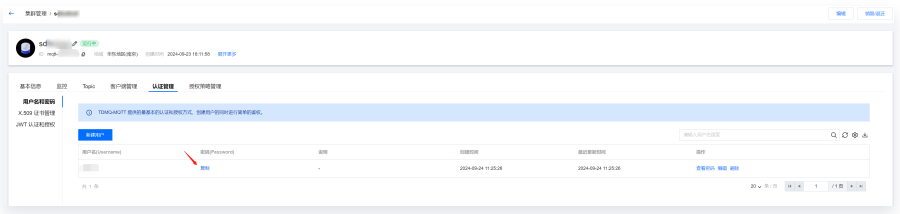
步骤1：安装 Java 依赖库

在 Java 项目中引入相关依赖，以 Maven 工程为例，在 pom.xml 添加以下依赖：

```
 <dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.2.5</version>
</dependency>
<dependency>
    <groupId>org.bouncycastle</groupId>
    <artifactId>bcpkix-jdk15on</artifactId>
    <version>1.70</version>
</dependency>
```

步骤2：参数和配置说明

参数	说明
BROKER_ADDR	broker 连接地址，可以从 控制台 目标集群**基本信息** > **接入信息**处复制。位置如下图所示。格式：mqtt-xxx-gz.mqtt.qcloud.tencenttdmq.com:1883

	
USERNAME	连接用户名，请从控制台集群认证管理页处复制。
PASSWORD	连接用户名匹配的密码，请从控制台集群认证管理页处复制。 
FIRST_TOPIC	MQTT 第一级 Topic，请从控制台集群 Topic 页处复制。

步骤3：消息发布

示例代码如下：

```
String server = "tcp://" + BROKER_ADDR;

// 请使用全局唯一的 clientId
String clientId = MqttClient.generateClientId();

// 如果启用持久session，请使用文件等非易失性存储介质
client = new MqttClient(server, clientId, new MemoryPersistence());

client.setCallback(new MqttCallback() {

    @Override
    public void connectionLost(Throwable cause) {
        System.out.println(MessageFormat.format("connection lost, cause:
{0}", cause));
```

```
    }

    @Override
    public void messageArrived(String topic, MqttMessage message) throws
Exception {
        // 处理业务逻辑
        System.out.println(MessageFormat.format("received message from
topic: {0}, payload: {1}", topic, message.toString()));
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken token) {
        try {
            System.out.println(MessageFormat.format("delivered message
to topics: {0}", Arrays.asList(token.getTopics())));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

// 设置请求超时时间，避免无限等待
client.setTimeToWait(3000);

MqttConnectOptions options = new MqttConnectOptions();
options.setUserName(USERNAME);
options.setPassword(PASSWORD.toCharArray());
options.setCleanSession(false);
options.setConnectionTimeout(3000);

// 连接 broker
client.connect(options);
if (!client.isConnected()) {
    System.out.println("Failed to connect to broker: " + server);
    return;
} else {
    System.out.println("Connected to broker: " + server);
}

// 向 topic 发布消息
for (int i = 0; i < 10; i++) {
    MqttMessage msg = new
MqttMessage("payload".getBytes(StandardCharsets.UTF_8));
```

```
    msg.setQos(1);
    msg.setRetained(false);
    client.publish(FIRST_TOPIC + "/second/first", msg);
}

// 断开与 broker 的连接
client.disconnect();

// 关闭 client
client.close();
```

步骤4：消息订阅

示例代码如下：

```
String server = "tcp://" + BROKER_ADDR;

// 请使用全局唯一的 clientId
String clientId = MqttClient.generateClientId();

// 如果启用持久session，请使用文件等非易失性存储介质
client = new MqttClient(server, clientId, new MemoryPersistence());

client.setCallback(new MqttCallback() {

    @Override
    public void connectionLost(Throwable cause) {
        System.out.println(MessageFormat.format("connection lost, cause:
{0}", cause));
    }

    @Override
    public void messageArrived(String topic, MqttMessage message) throws
Exception {
        // 处理业务逻辑
        System.out.println(MessageFormat.format("received message from
topic: {0}, payload: {1}", topic, message.toString()));
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken token) {
        try {
            System.out.println(MessageFormat.format("delivered message
```

```
to topics: {0}", Arrays.asList(token.getTopics())));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

// 设置请求超时时间，避免无限等待
client.setTimeToWait(3000);

MqttConnectOptions options = new MqttConnectOptions();
options.setUserName(USERNAME);
options.setPassword(PASSWORD.toCharArray());
options.setCleanSession(false);
options.setConnectionTimeout(3000);

// 连接 broker
client.connect(options);
if (!client.isConnected()) {
    System.out.println("Failed to connect to broker: " + server);
    return;
} else {
    System.out.println("Connected to broker: " + server);
}

// 订阅 topicFilter
String topicFilter = FIRST_TOPIC + "/second/first";
client.subscribe(topicFilter, 1);
System.out.println("Subscribed to: " + topicFilter);

TimeUnit.MINUTES.sleep(10);

// 取消订阅
client.unsubscribe();

// 断开与 broker 的连接
client.disconnect();

// 关闭 client
client.close();
```