消息队列 Pulsar 版

快速入门

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。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。

文档目录

快速入门
使用 SDK 收发普通消息
资源创建与准备
下载并运行 Demo
使用消息重试和死信队列
使用 Share 订阅模式消费消息

版权所有：腾讯云计算（北京）有限责任公司

快速入门
使用 SDK 收发普通消息
资源创建与准备

最近更新时间：2024-01-03 14:12:49

操作场景

该任务指导您通过 TDMQ Pulsar 版控制台创建集群和 Topic 等资源，了解运行一个客户端之前，控制台所需要进行的操作。

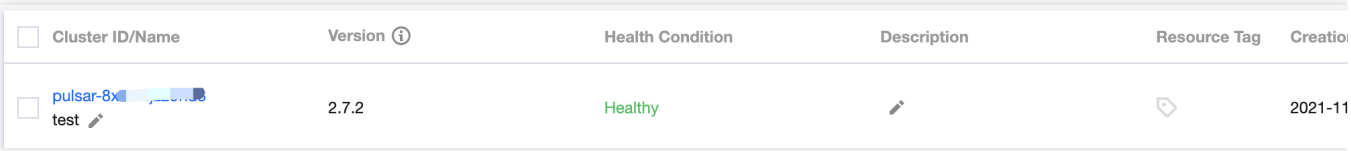
前提条件

已 注册腾讯云账号。

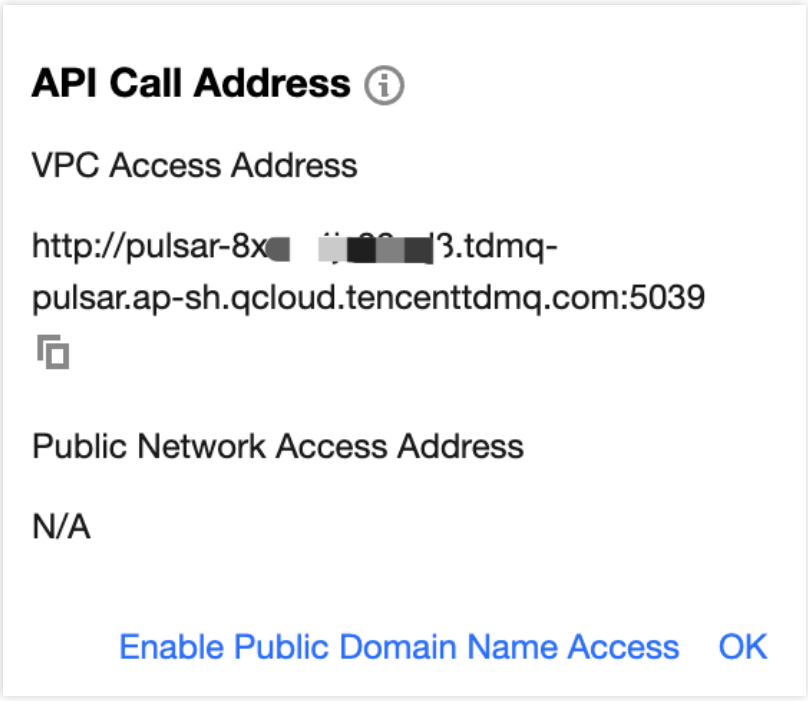
操作步骤

步骤1：新建集群并配置网络

1. 登录 TDMQ Pulsar 版控制台，进入**集群管理**页面，选择目标地域。
2. 单击**新建集群**，创建一个集群。
3. 在创建好的集群中，单击操作列的**接入地址**。



接入地址获取方式如下：
2.7.1及以上版本集群
2.6.1版本集群
直接获取接入地址，如下图。



在接入点列表页，单击**新建**，新建一个 VPC 接入点（和运行客户端的资源所在 VPC 一致即可）。

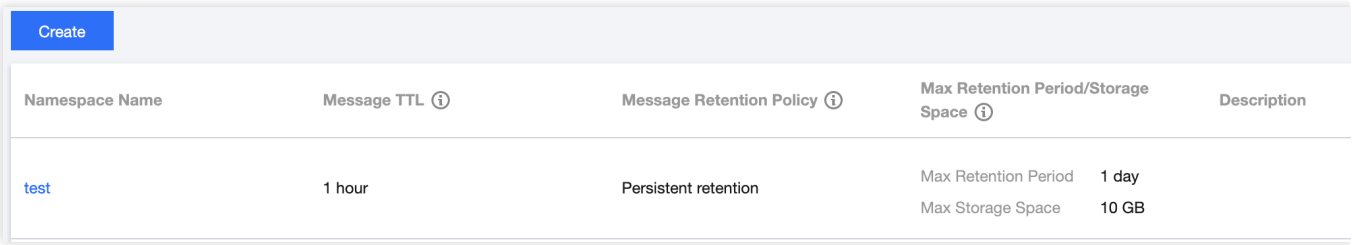
说明：

更多集群相关介绍与操作请参见 集群管理。

VPC 内网接入地址不支持跨地域访问，请确保客户端与虚拟集群在同一个地域。

步骤2：创建命名空间

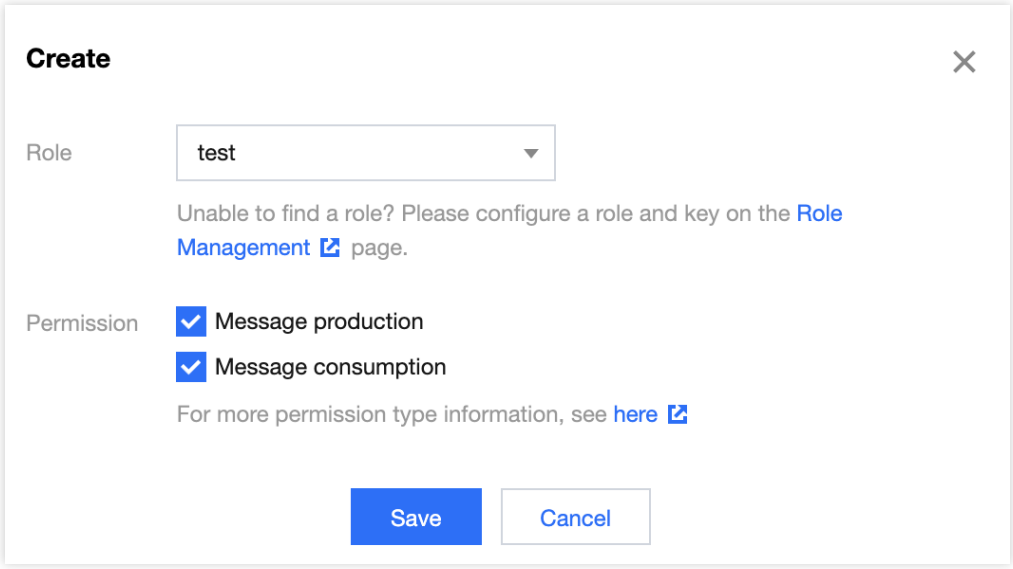
在 TDMQ Pulsar 版控制台的 命名空间 页面，选择好地域和刚刚创建好的集群，单击**新建**，创建一个命名空间。



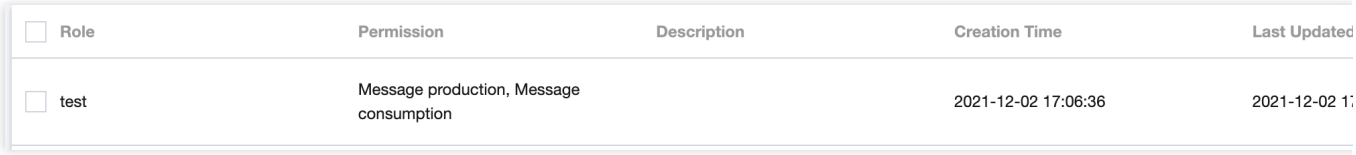
步骤3：创建角色并授权

1. 在 TDMQ Pulsar 版控制台的 角色管理 页面，选择地域和刚刚创建好的集群，单击**新建**进入新建角色页面。
2. 填写角色名称和说明，单击**提交**完成角色创建。
3. 进入 命名空间 页面，在刚刚创建的命名空间中，单击操作列的**配置权限**进入命名空间的权限列表。
4. 在配置权限页面，单击**添加角色**，将刚刚创建的角色添加进来，分配生产和消费的权限。

版权所有：腾讯云计算（北京）有限责任公司

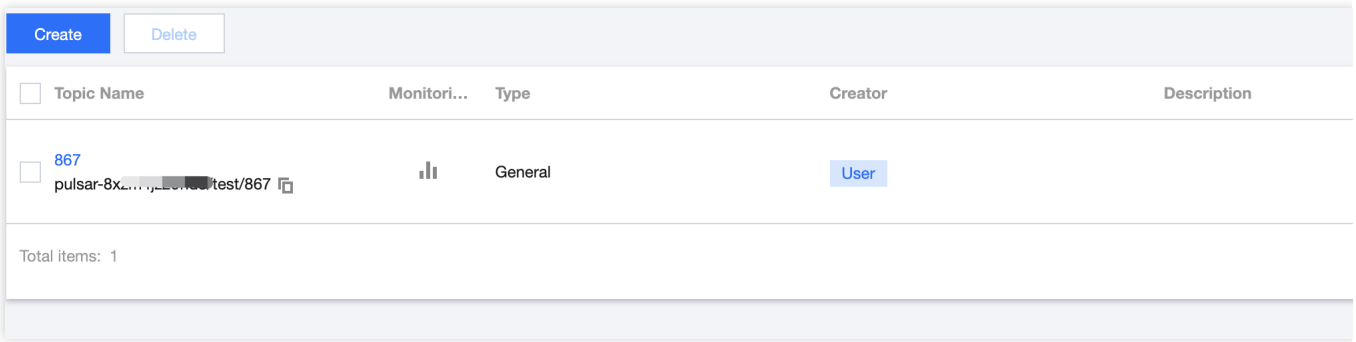


5. 看到下图即代表配置成功。



步骤4：创建 Topic 和订阅关系

1. 在 Topic 管理 页面，在顶部选择目标地域、当前集群和命名空间，单击**新建**，创建一个 Topic。
2. 单击操作列的**新增订阅**，为刚刚新建好的 Topic 创建一个订阅关系。
3. 单击操作列的**更多 > 查看订阅/消费者**，可看到刚刚创建好的订阅。



下载并运行 Demo

最近更新时间：2024-07-04 16:07:40

操作场景

该任务指导您在购买 TDMQ Pulsar 版服务和腾讯云服务器后，下载 Demo 并进行简单的测试，了解运行一个客户端的操作步骤。

说明：

本示例以 Java 客户端为例说明，其他语言客户端请参见 SDK 文档。

前提条件

已 购买云服务器。

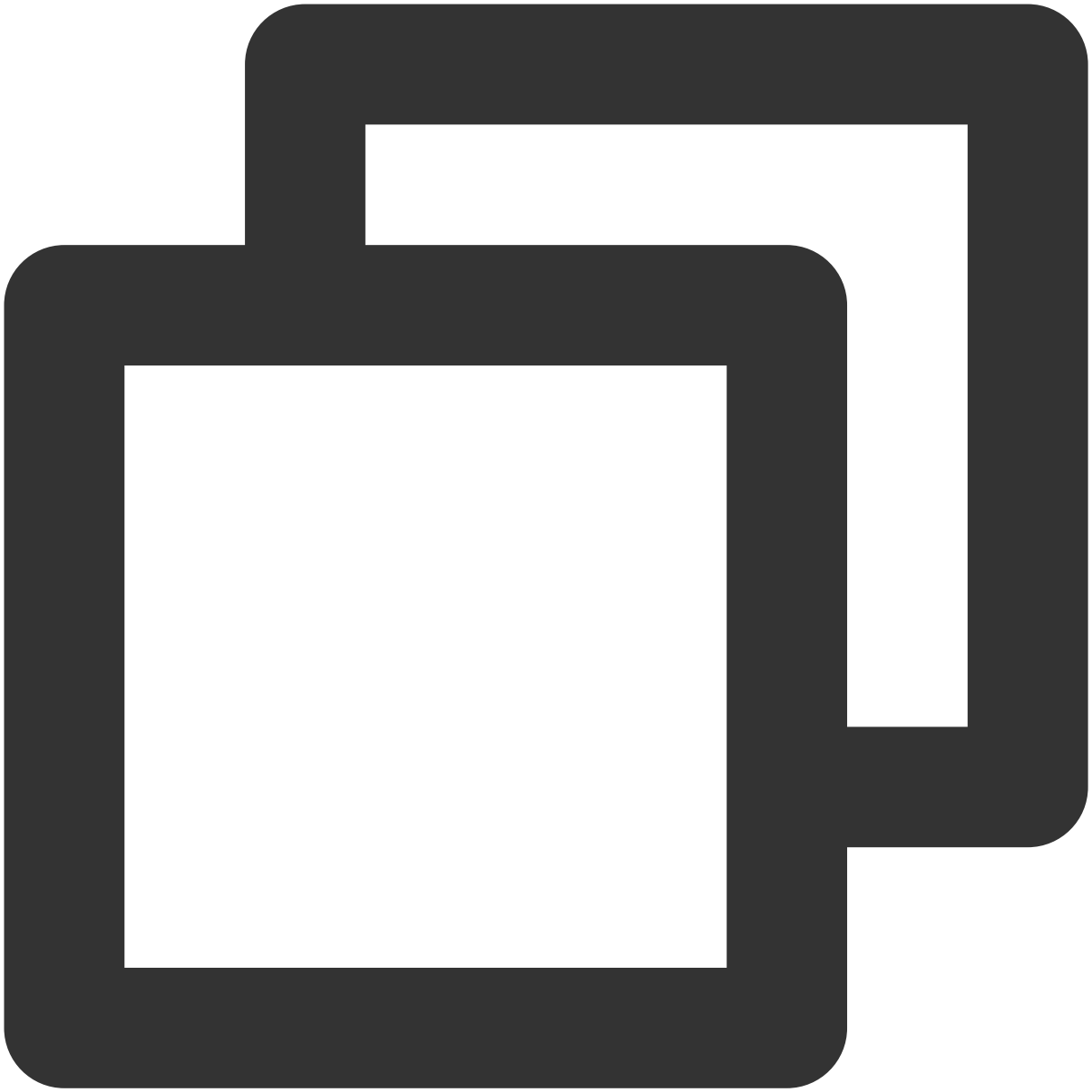
操作步骤

1. 下载 Demo（Demo 下载地址），并配置相关参数。

关于 Maven 依赖：

pom.xml 文件中的依赖是按照 Pulsar 的官方依赖进行配置的，详情可以参见 其官方文档。

版权所有：腾讯云计算（北京）有限责任公司

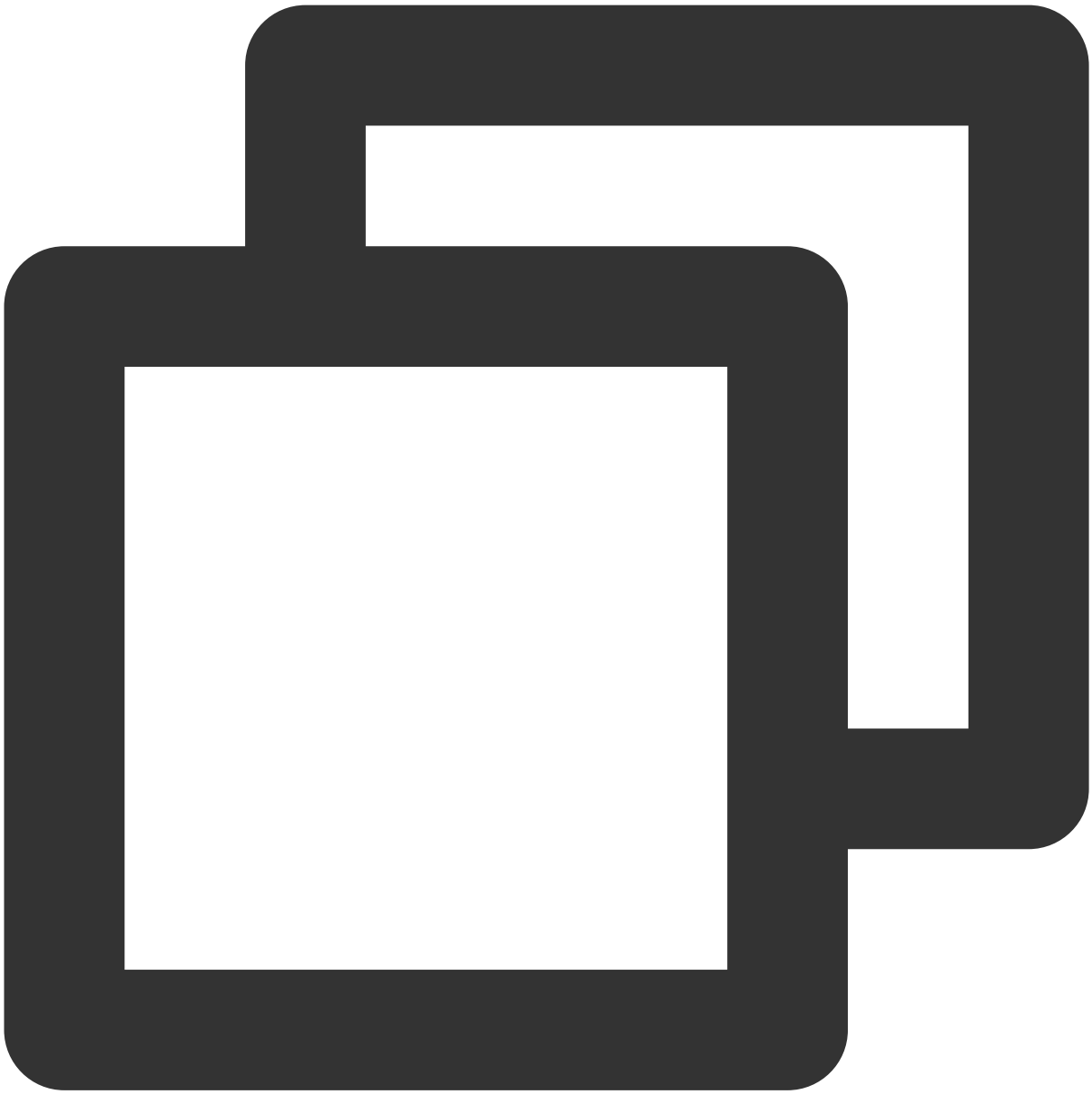


```
<!-- in your <properties> block -->
<pulsar.version>2.7.2</pulsar.version>
<!-- in your <dependencies> block -->
<dependency>
    <groupId>org.apache.pulsar</groupId>
    <artifactId>pulsar-client</artifactId>
    <version>${pulsar.version}</version>
</dependency>
```

创建 Client。

2.7.1版本及以上集群接入示例

2.6.1版本集群接入示例



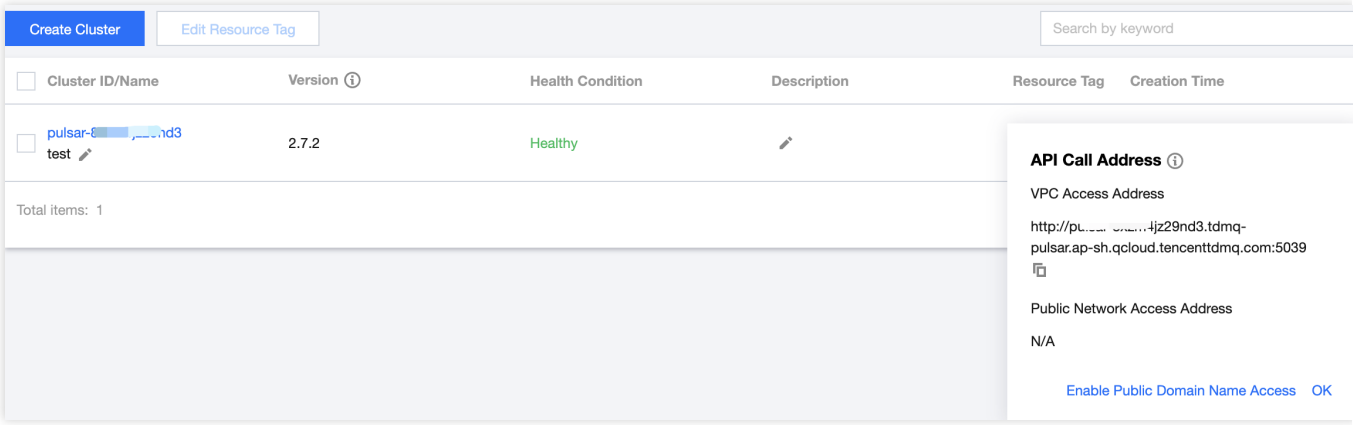
```
// 一个Pulsar client对应一个客户端链接
// 原则上一个进程一个client，尽量避免重复创建，消耗资源
// 关于客户端和生产消费者的实践教程，可以参考官方文档 https://cloud.tencent.com/document/pro

PulsarClient client = PulsarClient.builder()
        //替换成集群接入地址，位于【集群管理】页面接入地址
        .serviceUrl("http://pulsar-..tencenttdmq.com:8080")
        //替换成角色密钥，位于【角色管理】页面
```

```
        .authentication(AuthenticationFactory.token("eyJr"))
        .build();

System.out.println(">> pulsar client created.");
```

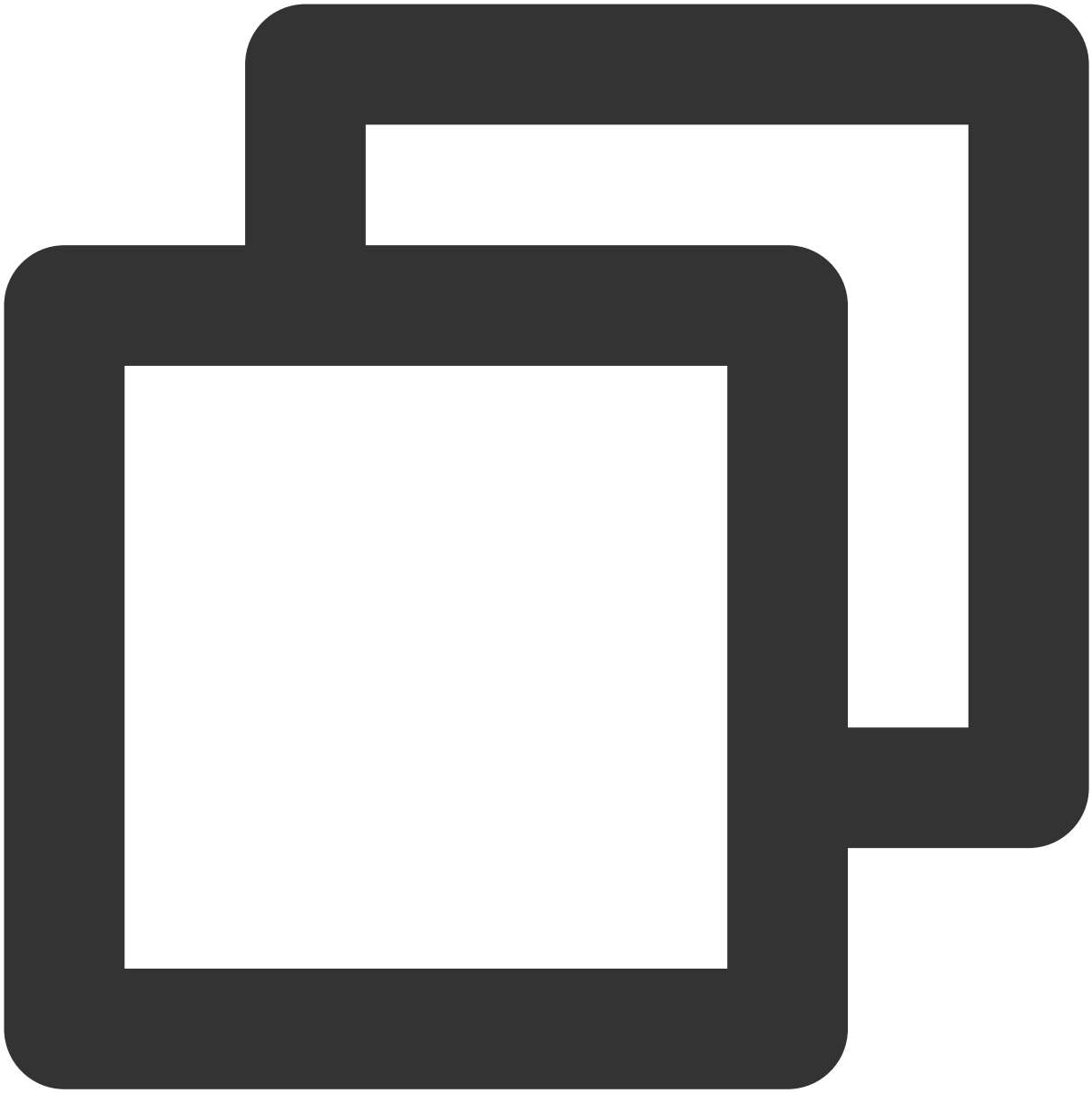
serviceUrl 即接入地址，可以在控制台 集群管理 页面查看并复制。



token 即角色的密钥，角色密钥可以在**角色管理**中复制。

注意：

密钥泄露很可能导致您的数据泄露，请妥善保管您的密钥。



```
PulsarClient client = PulsarClient.builder()
    .serviceUrl("pulsar://...:6000/")//接入地址到集群管理-接入点列表完整复制
    .listenerName("custom:pulsar-/vpc-/subnet-")//custom:替换成路由ID，位于**集群管理**
    .authentication(AuthenticationFactory.token("eyJr"))//替换成角色密钥，位于**角色管理
    .build();
System.out.println(">> pulsar client created.");
```

serviceUrl 即接入地址，可以在控制台 集群管理 接入点页面查看并复制。

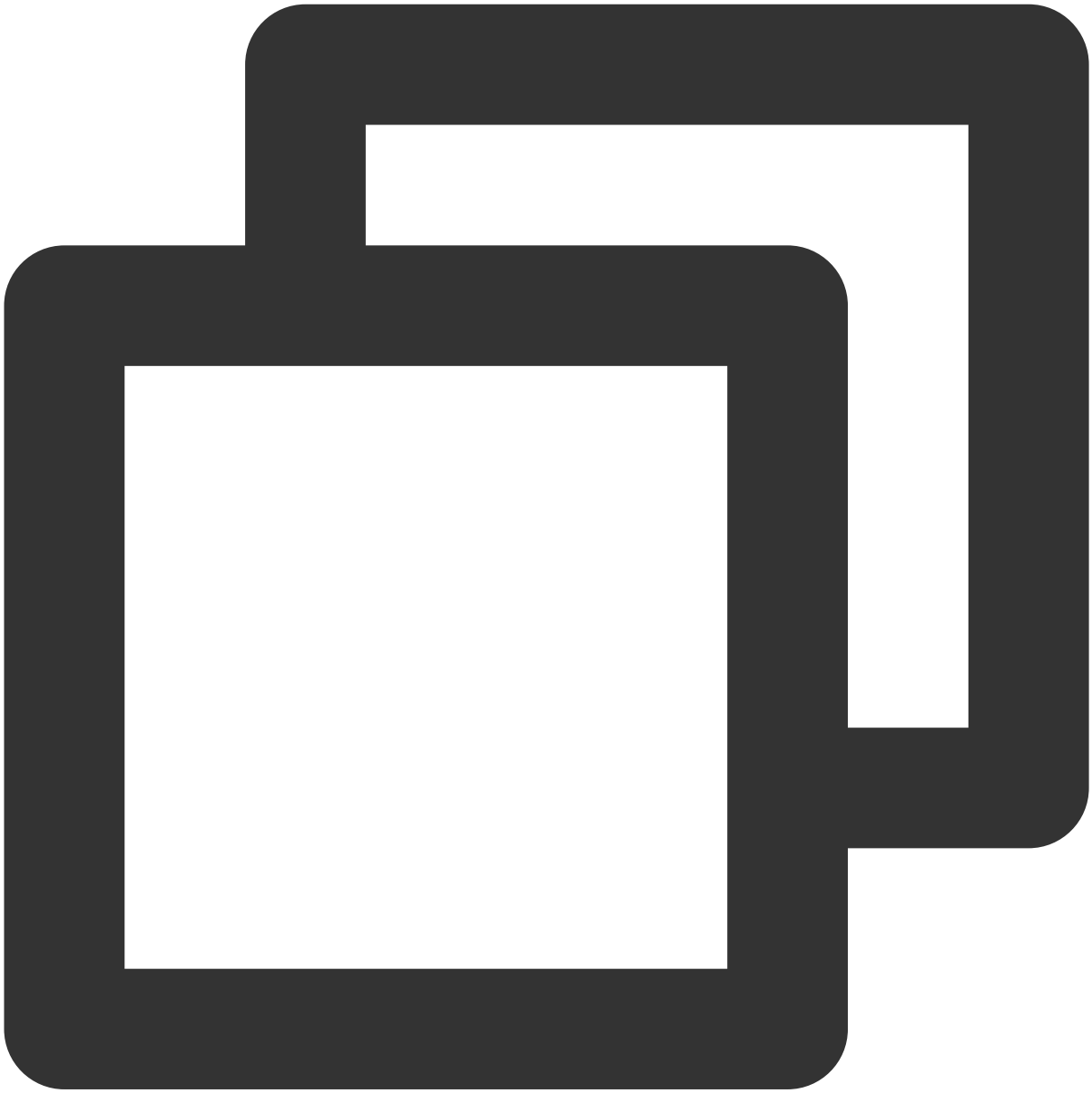
listenerName 即 “custom:” 拼接路由 ID（NetModel），路由 ID 可以在控制台 集群管理 接入点页面查看并复制。

token 即角色的密钥，角色密钥可以在**角色管理**中复制。

注意：

密钥泄露很可能导致您的数据泄露，请妥善保管您的密钥。

创建消费者进程。

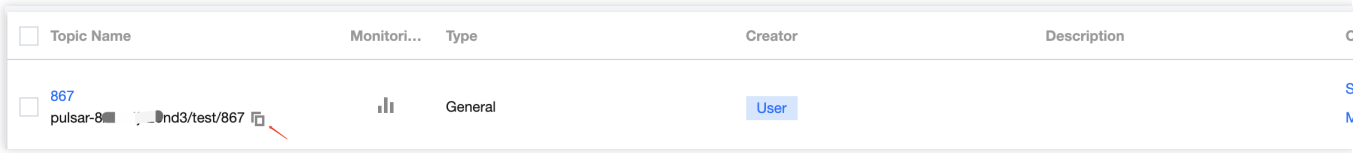


```
Consumer<byte[]> consumer = client.newConsumer()
                //topic完整路径，格式为persistent://集群（租户）ID/命名空间/Topic名称，从【
                .topic("persistent://pulsar-****/namespace/topicName")
                //需要在控制台Topic详情页创建好一个订阅，此处填写订阅名
                .subscriptionName("subscriptionName")
                //声明消费模式为exclusive（独占）模式
```

```
                .subscriptionType(SubscriptionType.Exclusive)
                //配置从最早开始消费，否则可能会消费不到历史消息
                .subscriptionInitialPosition(SubscriptionInitialPosition.Earliest)
                .subscribe();
        System.out.println(">> pulsar consumer created.");
```

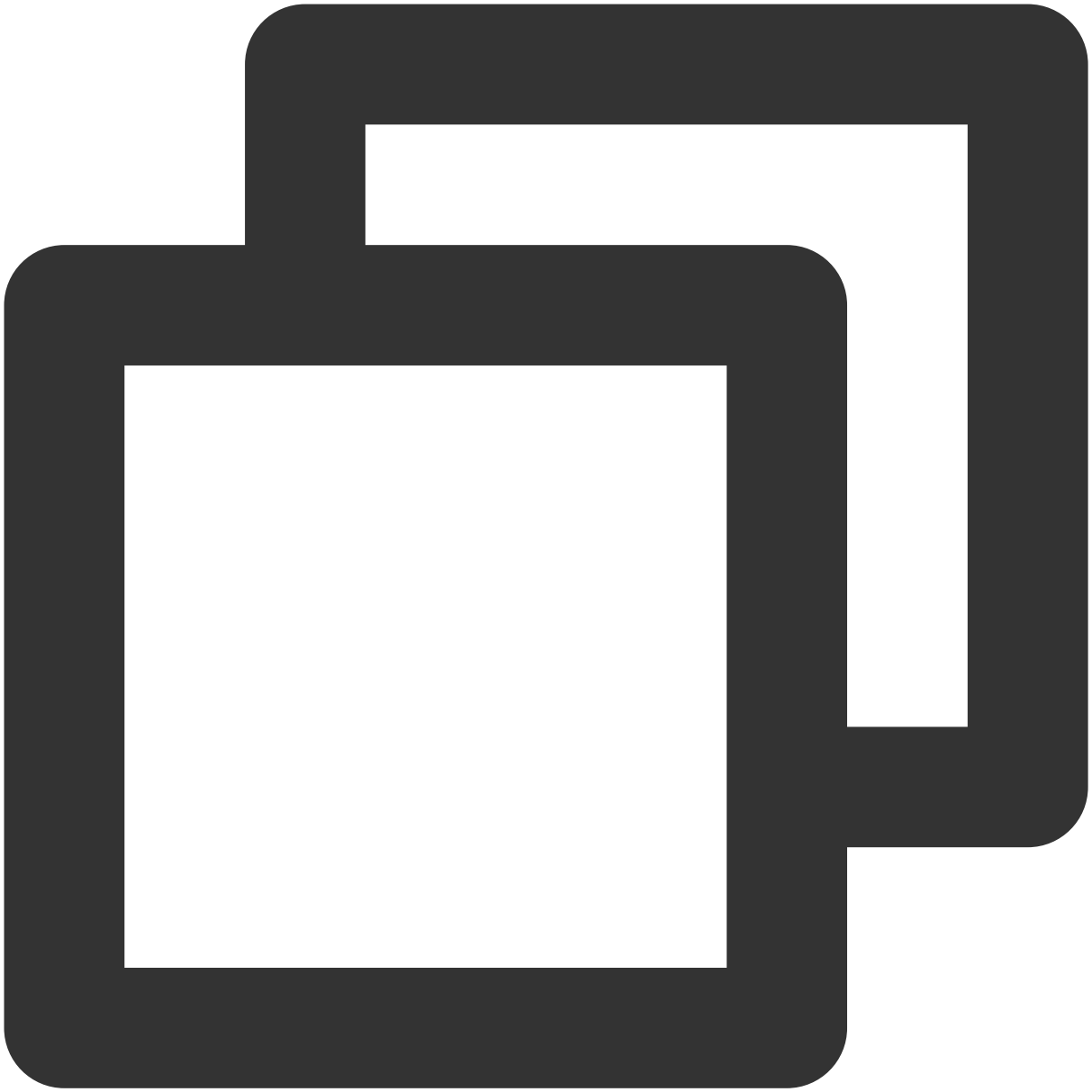
说明：

Topic 名称需要填入完整路径，即“persistent://clusterid/namespace/Topic”，clusterid/namespace/topic 的部分可以从控制台上 Topic 管理 页面直接复制。



subscriptionName 需要写入订阅名，可在**消费管理**界面查看。

创建生产者进程。

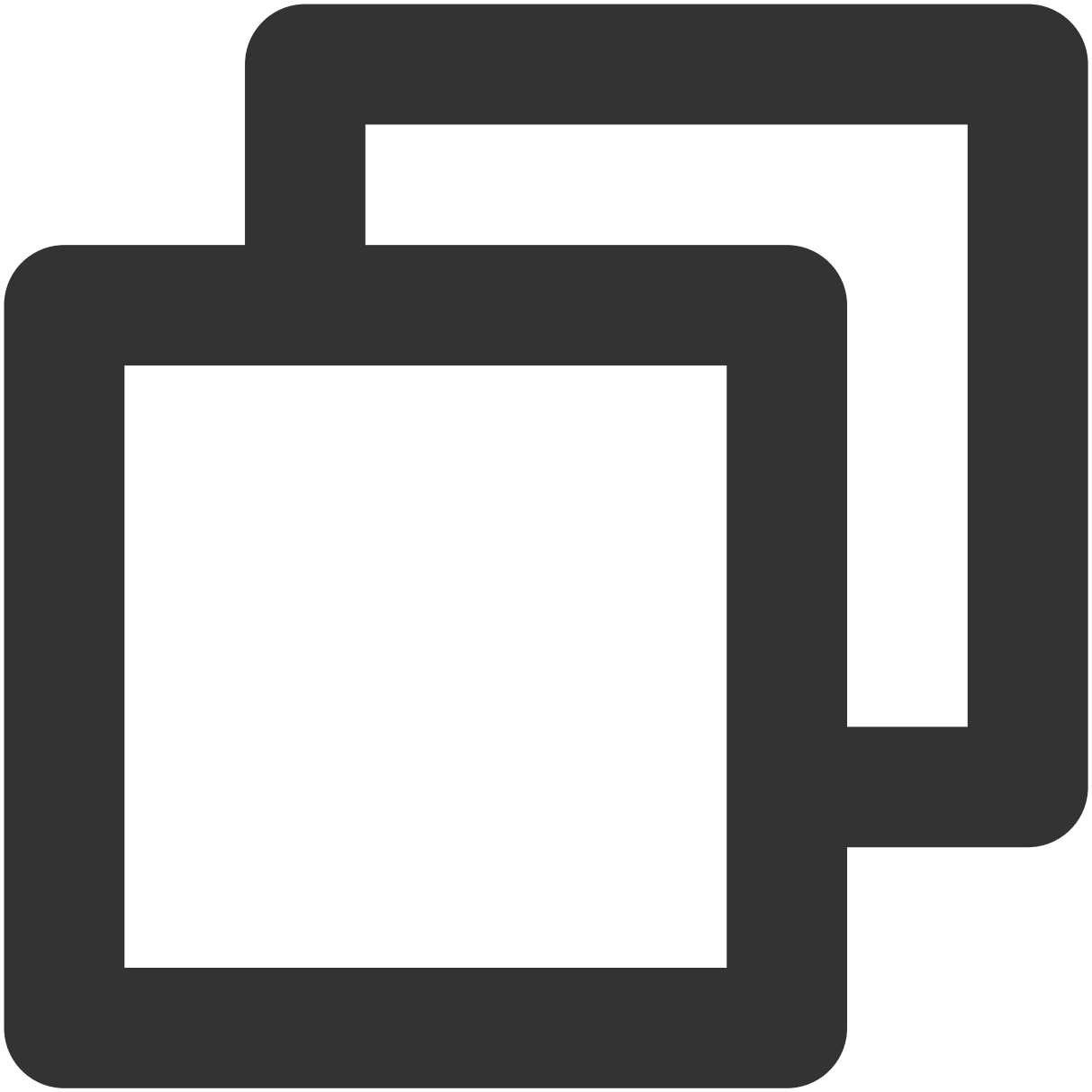


```
Producer<byte[]> producer = client.newProducer()
                //topic完整路径，格式为persistent://集群（租户）ID/命名空间/Topic名称
                .topic("persistent://pulsar-****/namespace/topicName")
                .create();
        System.out.println(">> pulsar producer created.");
```

说明：

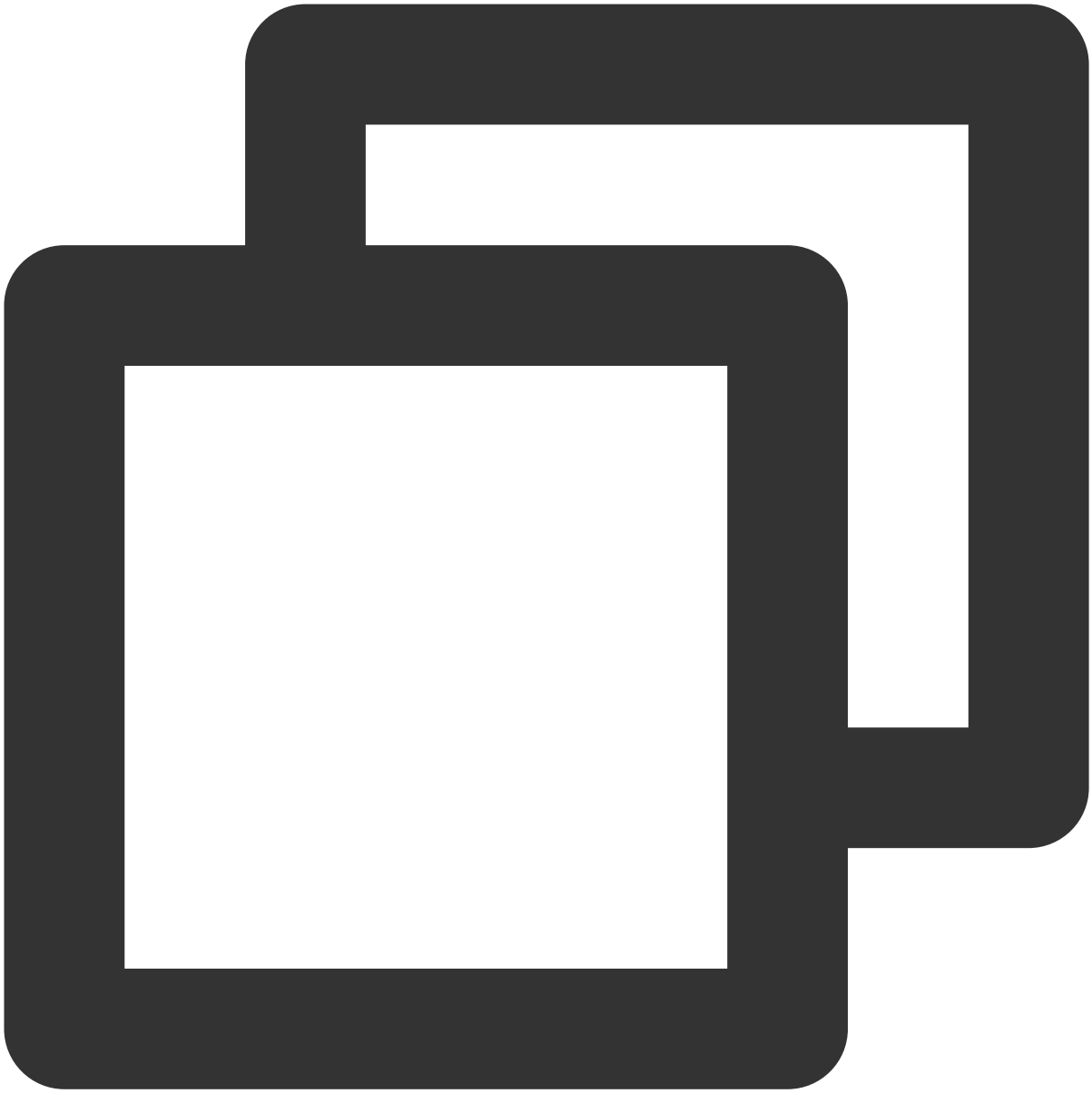
Topic 名称需要填入完整路径，即“persistent://clusterid/namespace/Topic”，clusterid/namespace/topic 的部分可以从控制台上 Topic 管理 页面直接复制。

生产消息



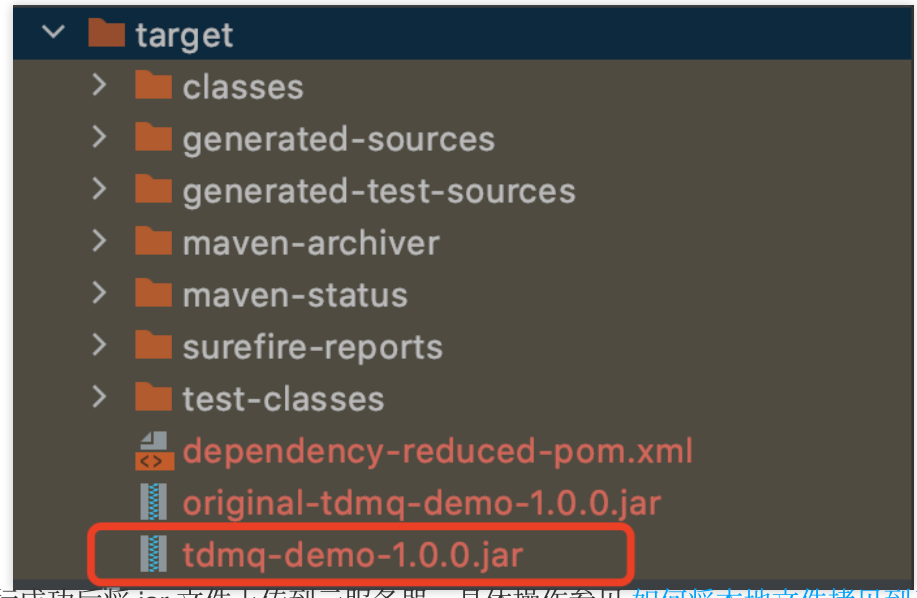
```
for (int i = 0; i < 5; i++) {
            String value = "my-sync-message-" + i;
            //发送消息
            MessageId msgId = producer.newMessage().value(value.getBytes()).send();
            System.out.println("deliver msg " + msgId + ",value:" + value);
        }
        //关闭生产者
        producer.close();
```

消费消息



```
for (int i = 0; i < 5; i++) {
            //接收当前offset对应的一条消息
            Message<byte[]> msg = consumer.receive();
            MessageId msgId = msg.getMessageId();
            String value = new String(msg.getValue());
            System.out.println("receive msg " + msgId + ",value:" + value);
            //接收到之后必须要ack，否则offset会一直停留在当前消息，无法继续消费
            consumer.acknowledge(msg);
        }
```

2. 在 `pom.xml` 所在目录执行命令 `mvn clean package`，或者通过 IDE 自带的功能打包整个工程，在 target 目录下生成一个可运行的 jar 文件。



3. 运行成功后将 jar 文件上传到云服务器，具体操作参见 如何将本地文件拷贝到云服务器。

4. 登录云服务器，进入到刚刚上传 jar 文件所在的目录，可看到文件已上传到云服务器。

```
[root@VM-252-4-centos ~]#
[root@VM-252-4-centos /home]#
total 57984
-rw-r--r-- 1 root root 29684882 May 17 17:21 tdmq-demo-1.0.0.jar
```

执行命令 `java -jar tdmq-demo-1.0.0.jar`，运行 Demo，可查看运行日志。

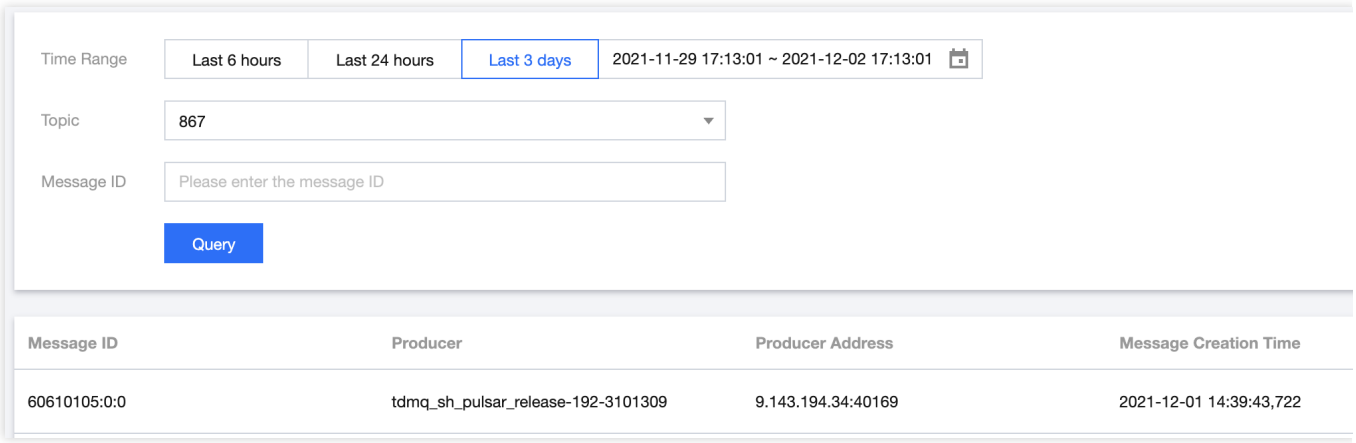
```
>> pulsar client created.
>> pulsar consumer created.
>> pulsar producer created.
deliver msg msgId,value:my-sync-message-0
deliver msg msgId,value:my-sync-message-1
deliver msg msgId,value:my-sync-message-2
deliver msg msgId,value:my-sync-message-3
deliver msg msgId,value:my-sync-message-4
receive msg org.apache.pulsar.client.impl.TopicMessageIdImpl@abd759b0,valu
receive msg org.apache.pulsar.client.impl.TopicMessageIdImpl@abdb138b,valu
receive msg org.apache.pulsar.client.impl.TopicMessageIdImpl@abdb174c,valu
receive msg org.apache.pulsar.client.impl.TopicMessageIdImpl@abdb1b0d,valu
receive msg org.apache.pulsar.client.impl.TopicMessageIdImpl@abd16df5,valu
```

5. 登录 TDMQ Pulsar 版控制台，依次点击 **Topic 管理** > **Topic 名称**进入消费管理页面，点开订阅名下方右三角号，可查看生产消费记录。

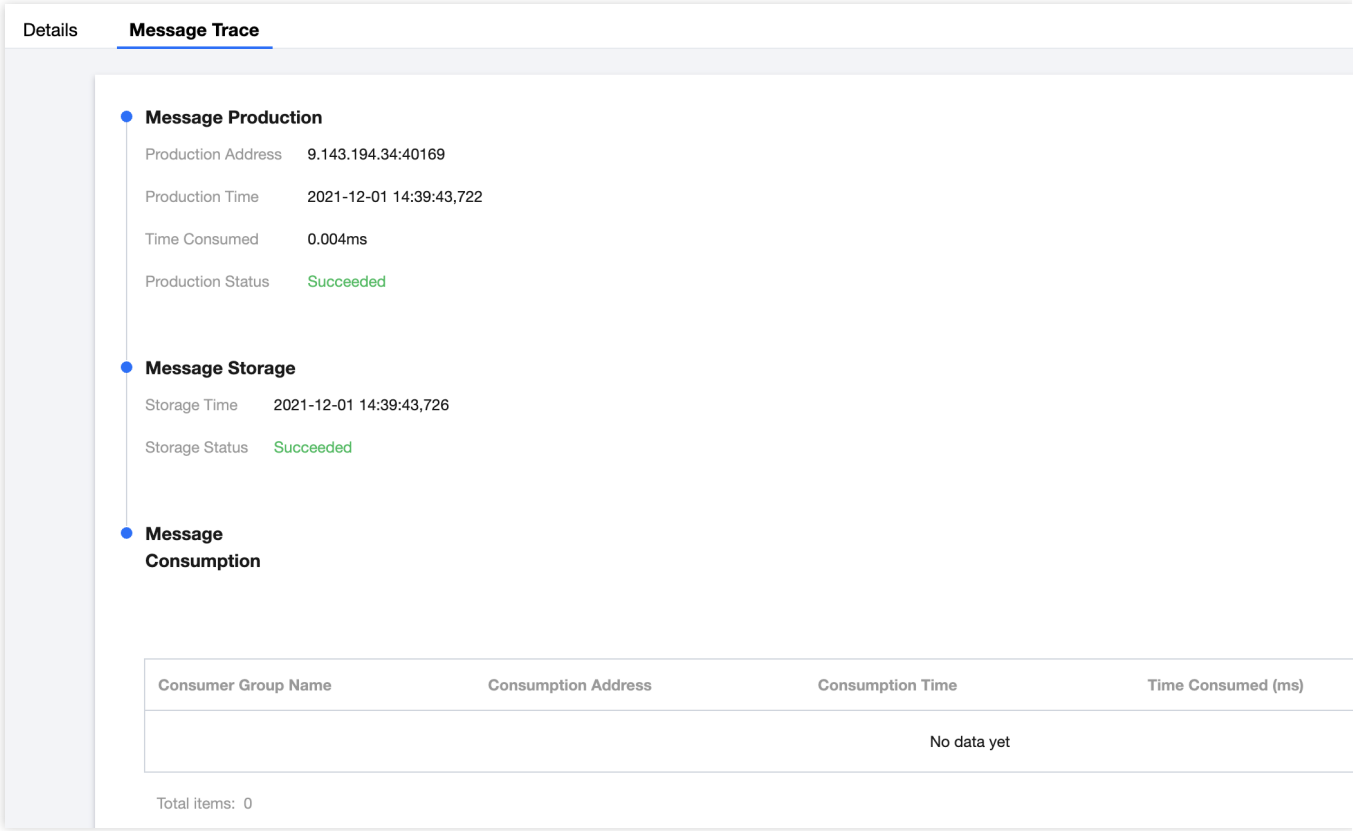
6. 进入 消息查询 页面，可查看 Demo 运行后的消息轨迹。

说明：

消息轨迹的查询只支持单条消息，如果用户在 Producer 侧开启了 Batch 功能，则在消息查询中，同一个 Batch 的消息只可以查询到 Batch 中的第一条消息。



消息轨迹如下：



使用消息重试和死信队列

最近更新时间：2024-08-19 16:04:55

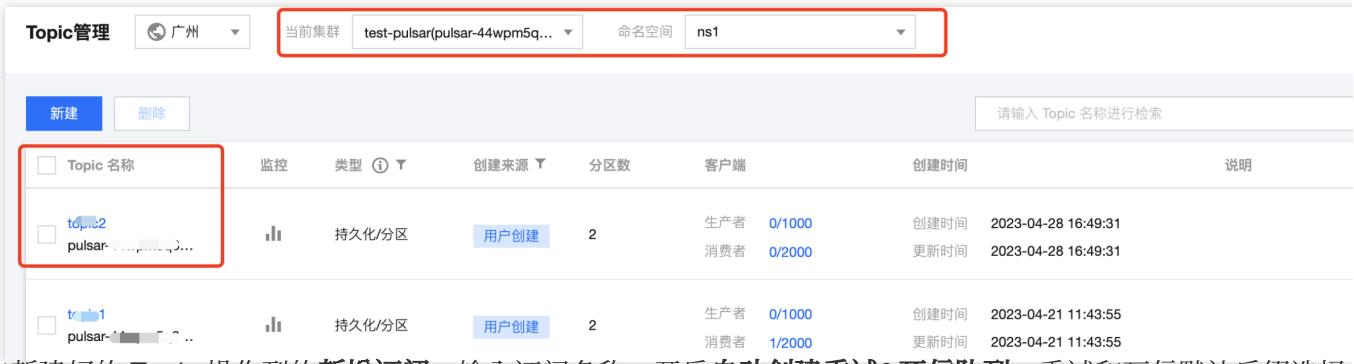
操作背景

重试 Topic 是一种为了确保消息被正常消费而设计的 Topic 。当某些消息第一次被消费者消费后，没有得到正常的回应，则会进入重试 Topic 中，当重试达到一定次数后，停止重试，投递到死信 Topic 中。
当消息进入到死信队列中，表示 TDMQ Pulsar 版已经无法自动处理这批消息，一般这时就需要人为介入来处理这批消息。您可以通过编写专门的客户端来订阅死信 Topic，处理这批之前处理失败的消息。
更多关于消息重试与死信的介绍请参考文档消息重试与死信机制。
本文以官网提供的 Demo 为例，介绍消息重试与死信队列功能与使用方式。

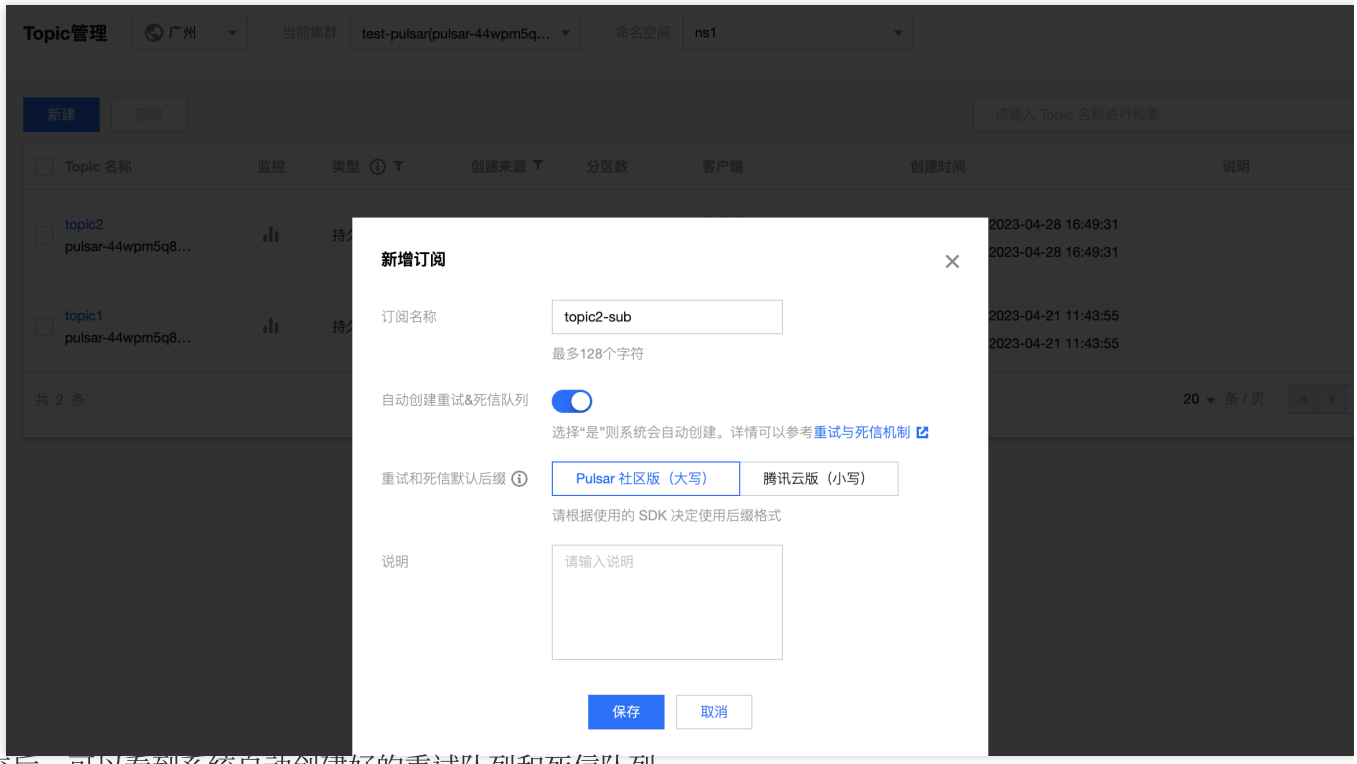
操作步骤

步骤1. 在控制台新建 Pulsar 资源

1. 登录 TDMQ Pulsar 版控制台，创建一个集群和命名空间。
2. 在左侧导航栏选择 **Topic 管理**页签，当前集群和命名空间分别勾选入门课程中创建好的集群和命名空间。
3. 单击**新建**，输入 Topic 名称和说明，其他选项可保持默认，单击**保存**，创建一个 topic。



4. 单击新建好的 Topic 操作列的**新增订阅**，输入订阅名称，开启**自动创建重试&死信队列**，重试和死信默认后缀选择 **Pulsar 社区版（大写）**，为刚新建好的 Topic 创建一个订阅关系并开启重试&死信队列。

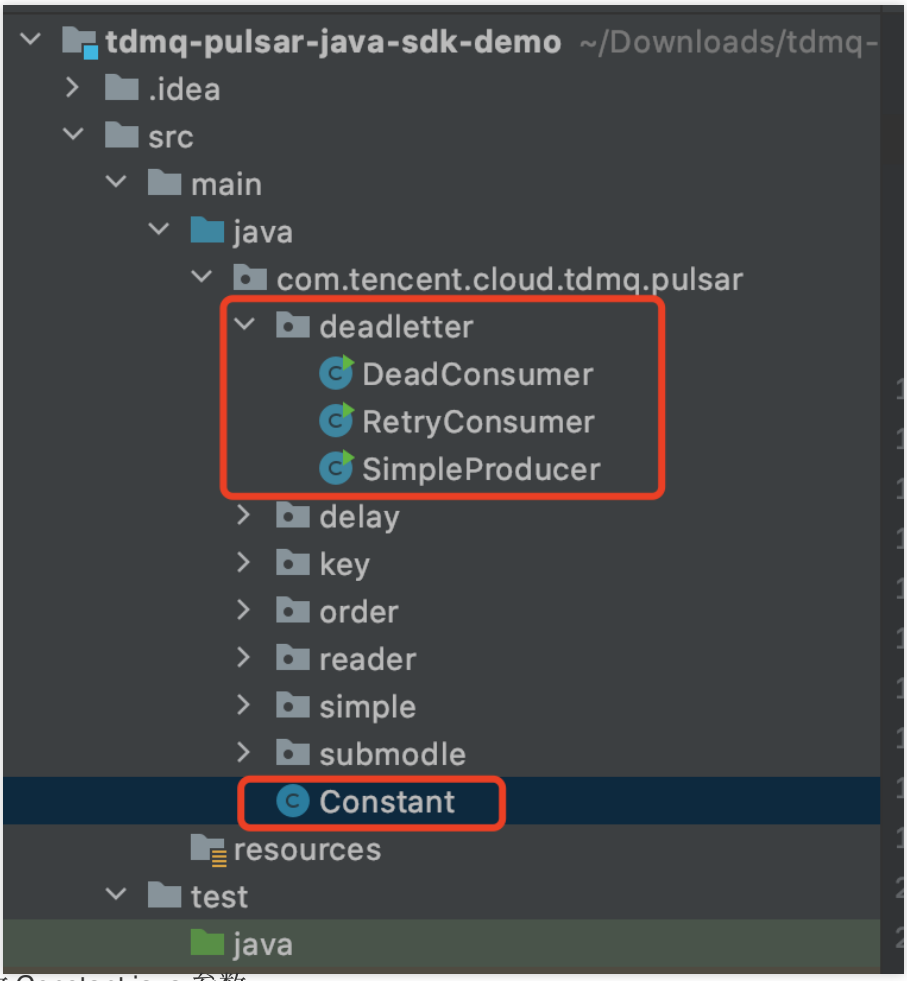


5. 提交后，可以看到系统自动创建好的重试队列和死信队列。

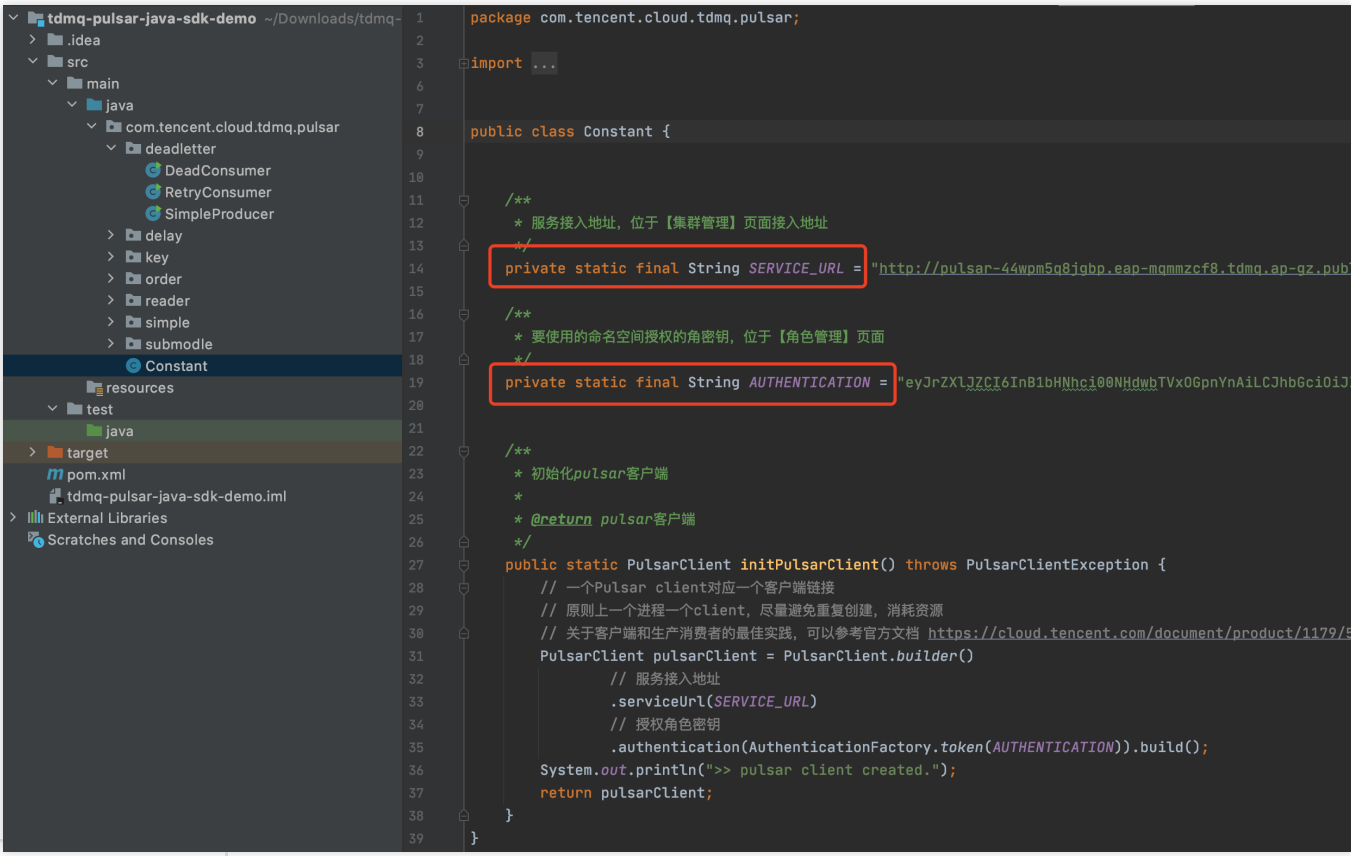
6. 单击操作列的**更多 > 查看订阅/消费者**，可看到刚创建好的订阅。

步骤2. 下载 Demo 并配置相关参数

1. 下载官方 Demo并解压。



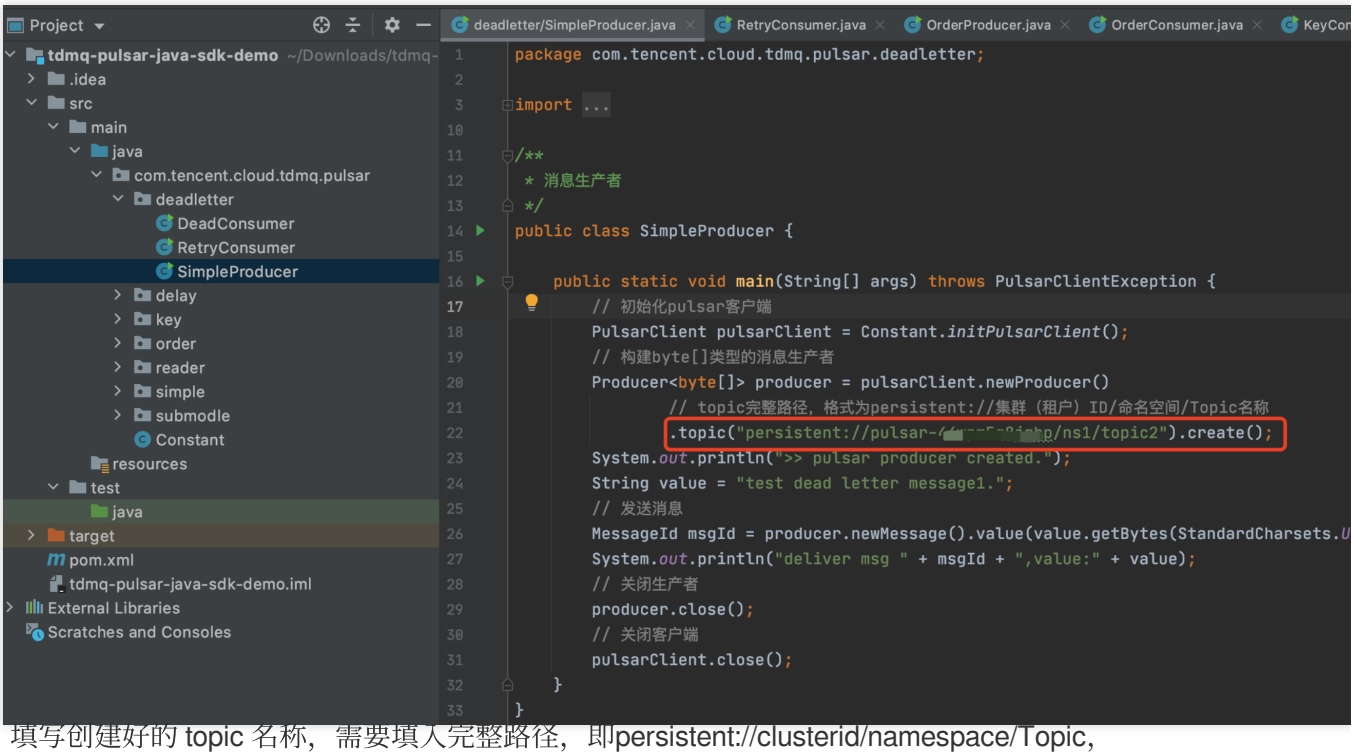
2. 修改 Constant.java 参数。



参数	说明
SERVICE_URL	集群接入地址，可以在控制台集群管理页面查看并复制。
AUTHENTICATION	角色的密钥，角色密钥可以在角色管理中复制。

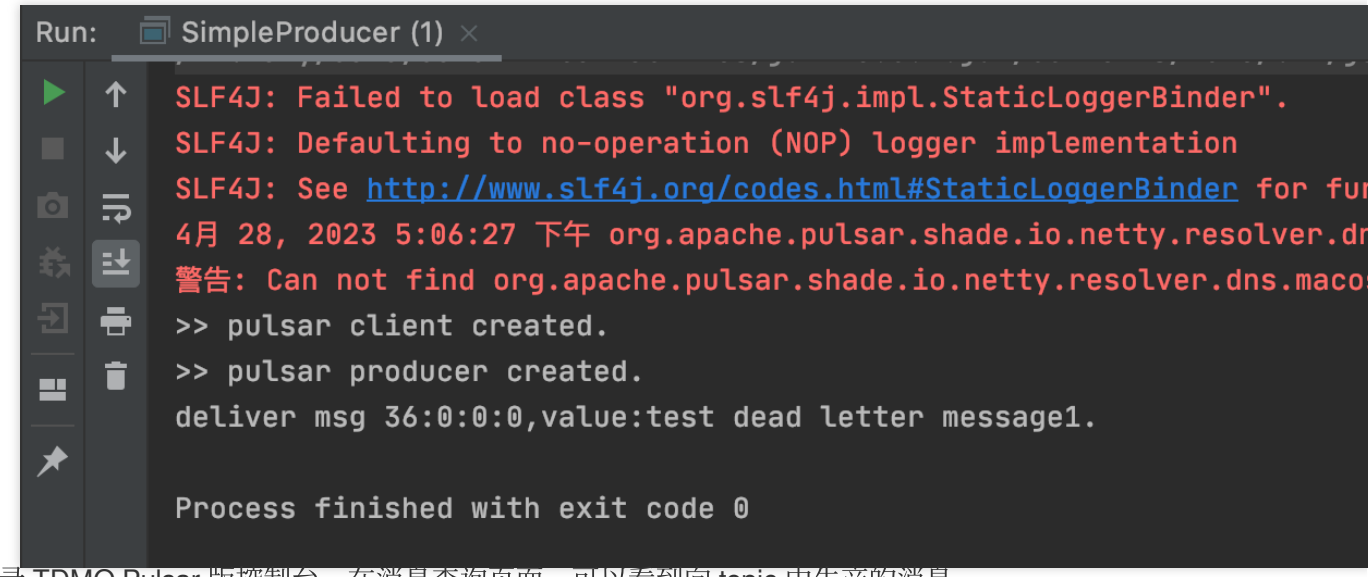
步骤3. 生产消息

1. 进入/deadletter目录下，修改 SimpleProducer.java 参数。

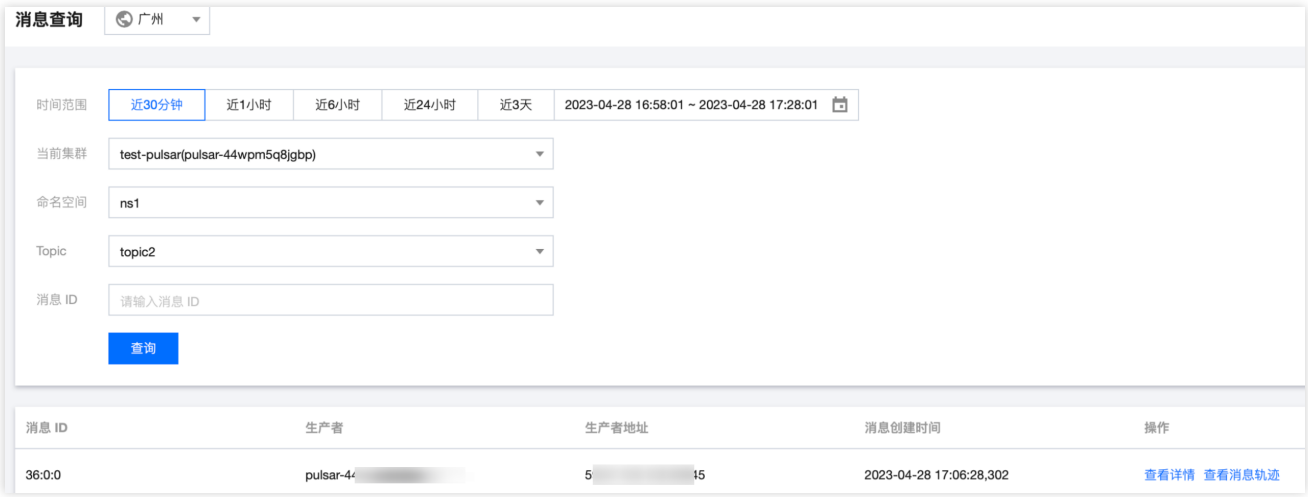


topic：填写创建好的 topic 名称，需要填入完整路径，即persistent://clusterid/namespace/Topic，clusterid/namespace/topic 的部分可以从控制台上 Topic管理页面直接复制。

2. 编译并运行 SimpleProducer.java 程序发送消息。运行结果如下：

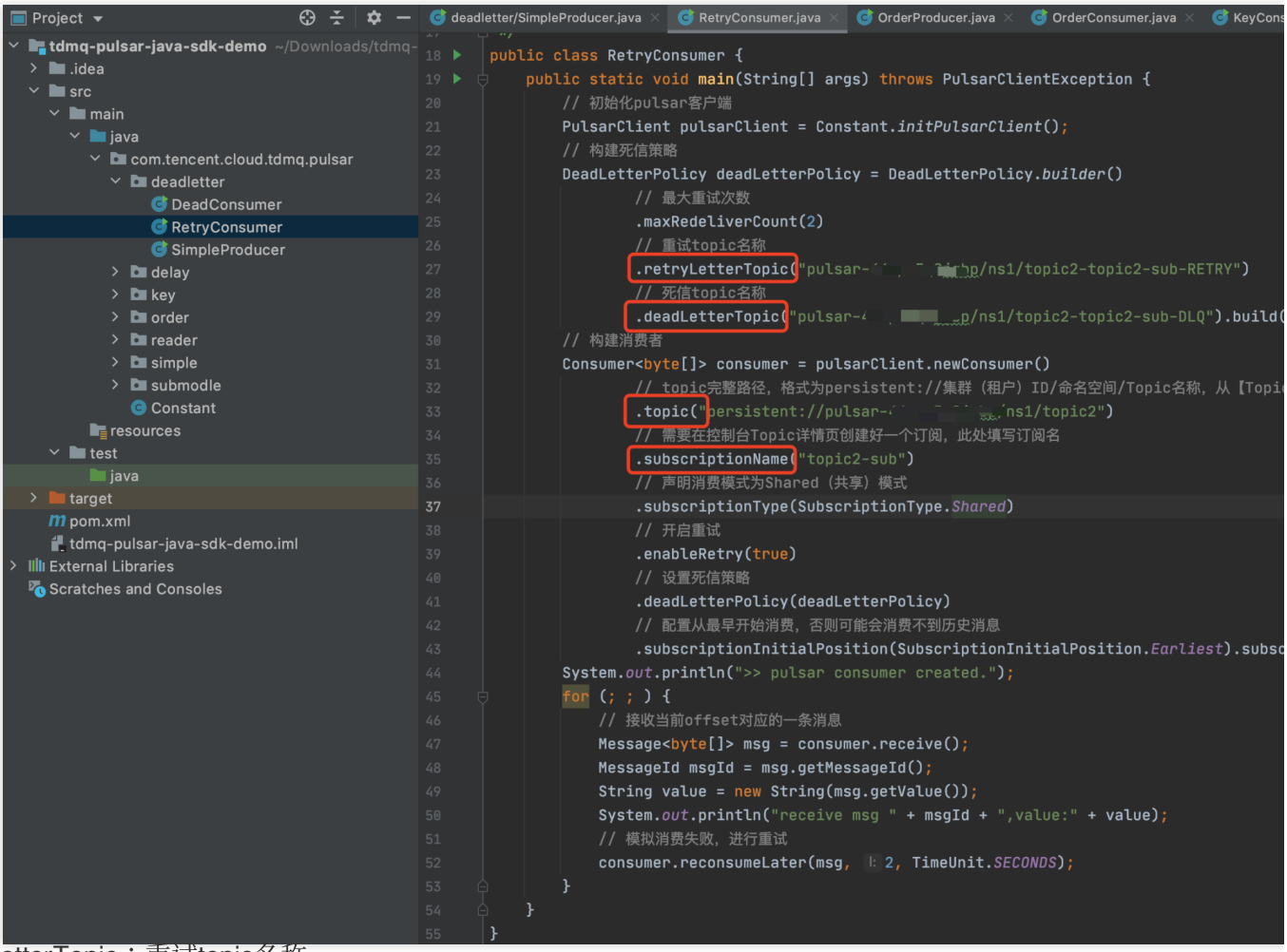


3. 登录 TDMQ Pulsar 版控制台，在消息查询页面，可以看到向 topic 中生产的消息。



步骤4. 消费消息

1. 修改 RetryConsumer.java 程序参数。



.retryLetterTopic：重试topic名称。

.deadLetterTopic：死信topic名称。

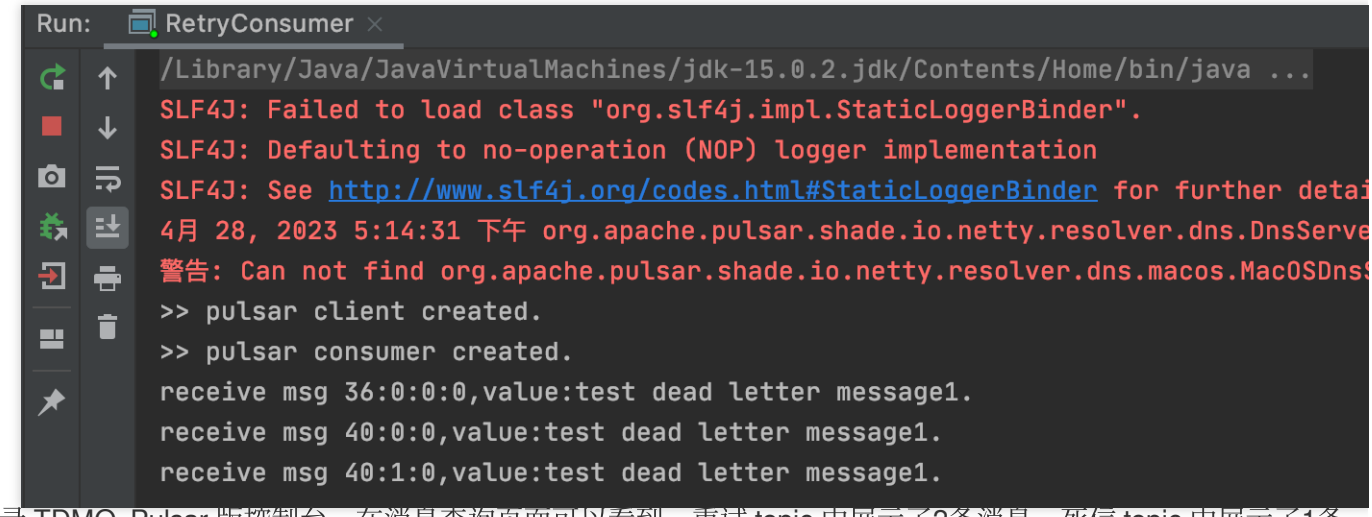
.topic：topic名称。

以上3个 topic 需要填入完整路径，即 `persistent://clusterid/namespace/Topic`，`clusterid/namespace/topic` 的部分可以从控制台上 Topic 管理页面直接复制。

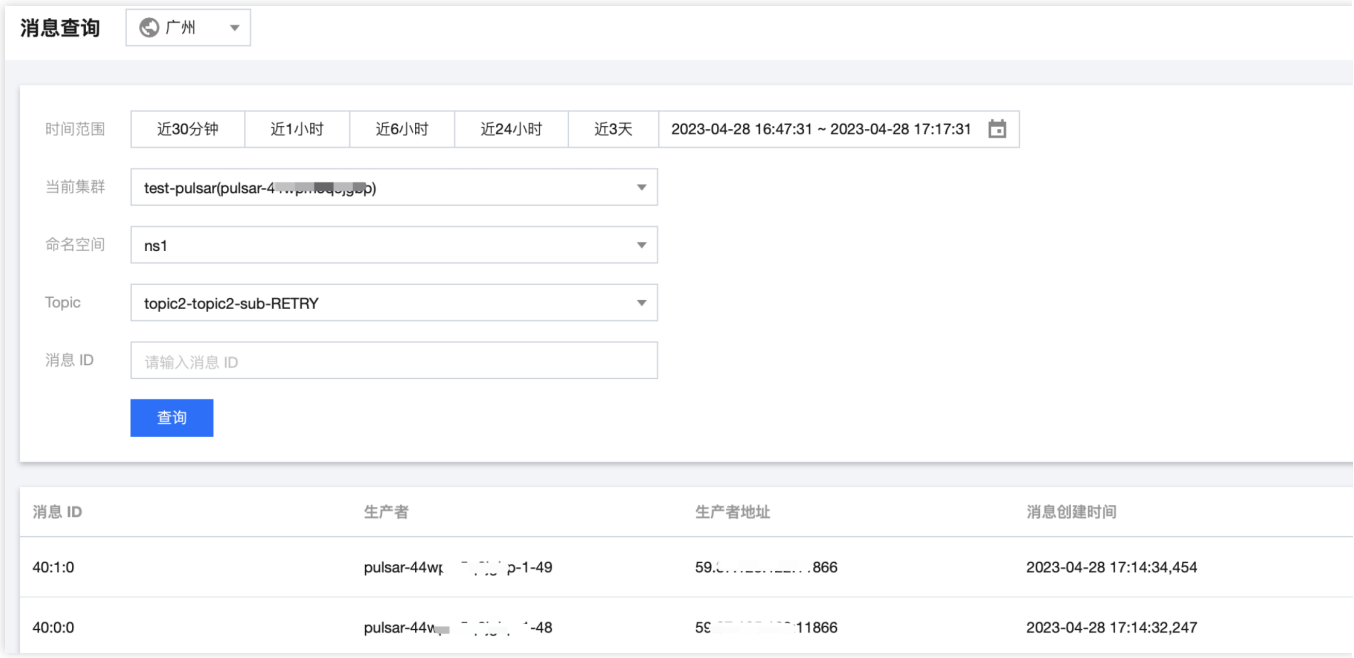
.subscriptionName：需要写入订阅名，可在 Topic 消费者界面查看。

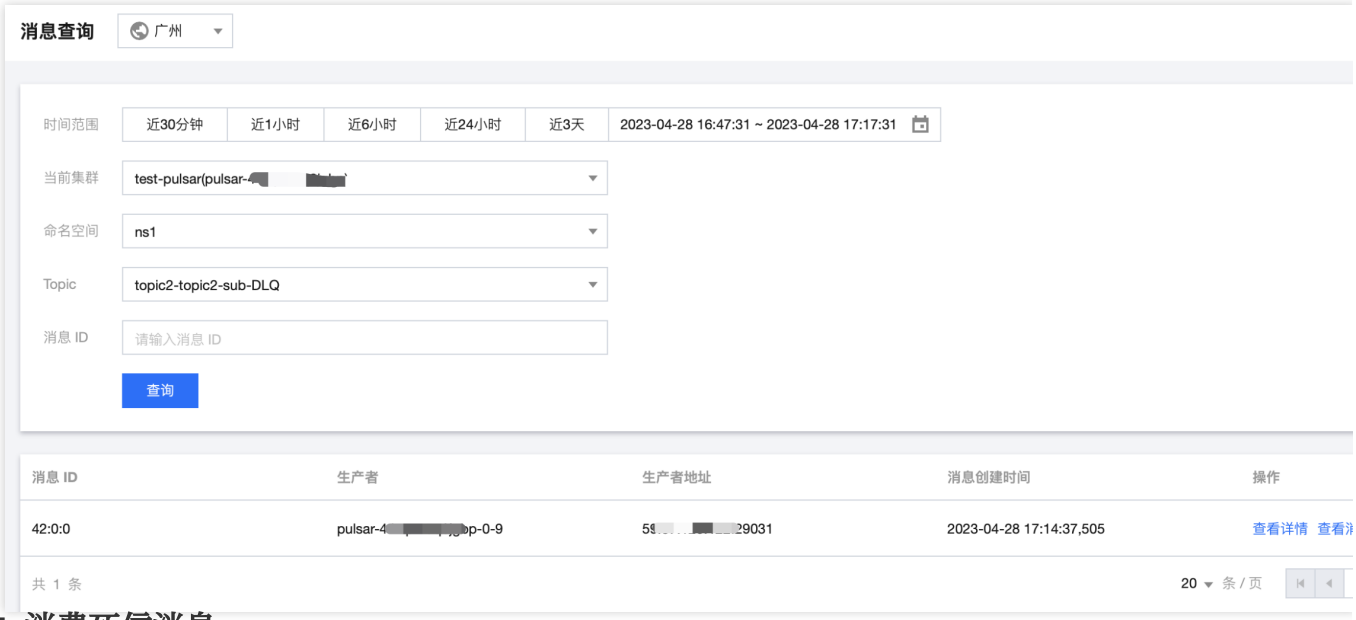
版权所有：腾讯云计算（北京）有限责任公司

2. 编译并运行消息重试程序 RetryConsumer.java 。运行结果如下：



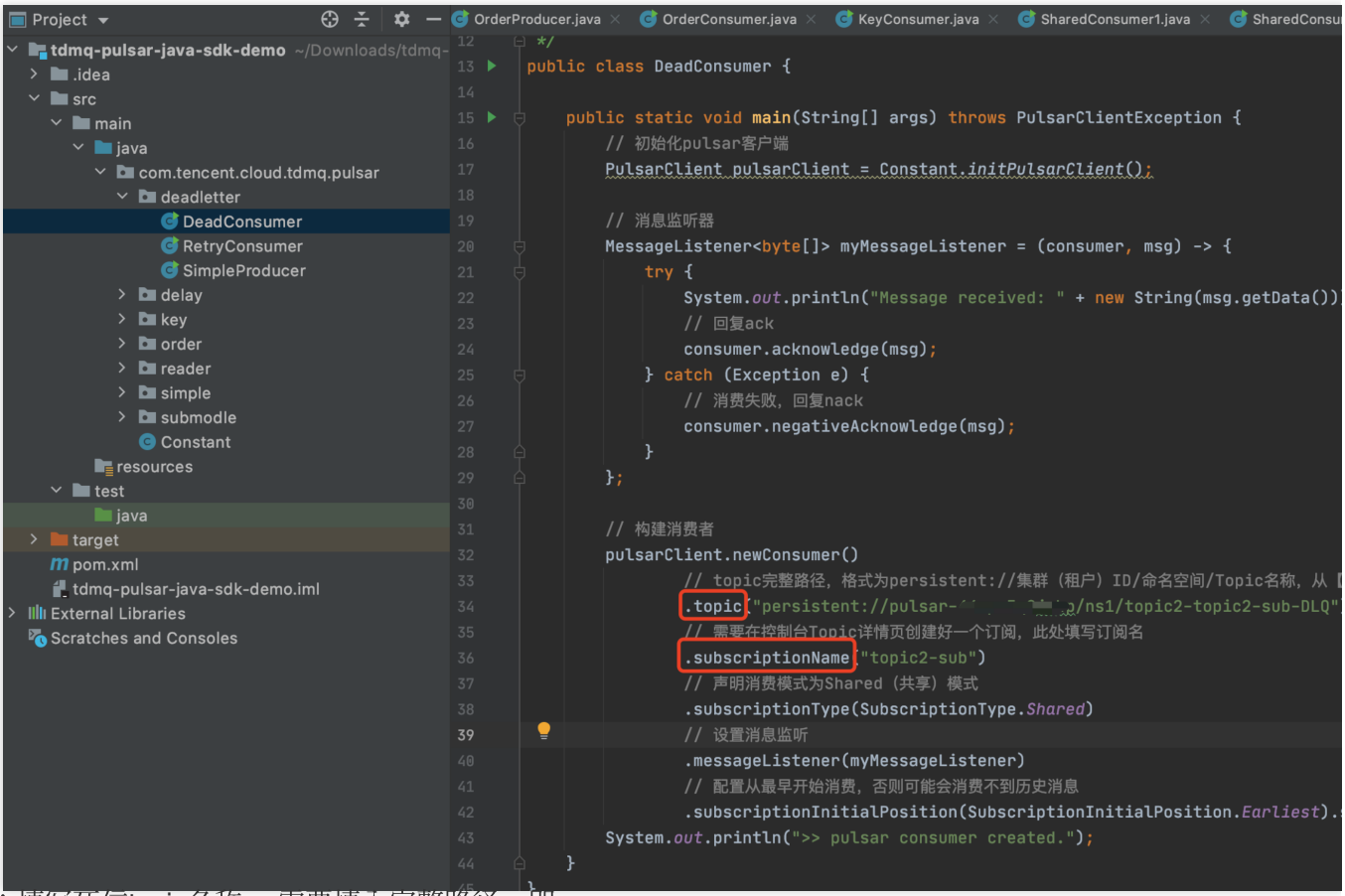
3. 登录 TDMQ Pulsar 版控制台，在消息查询页面可以看到，重试 topic 中展示了2条消息，死信 topic 中展示了1条消息，代表消息经过2次重试投递后，被投递到死信队列。



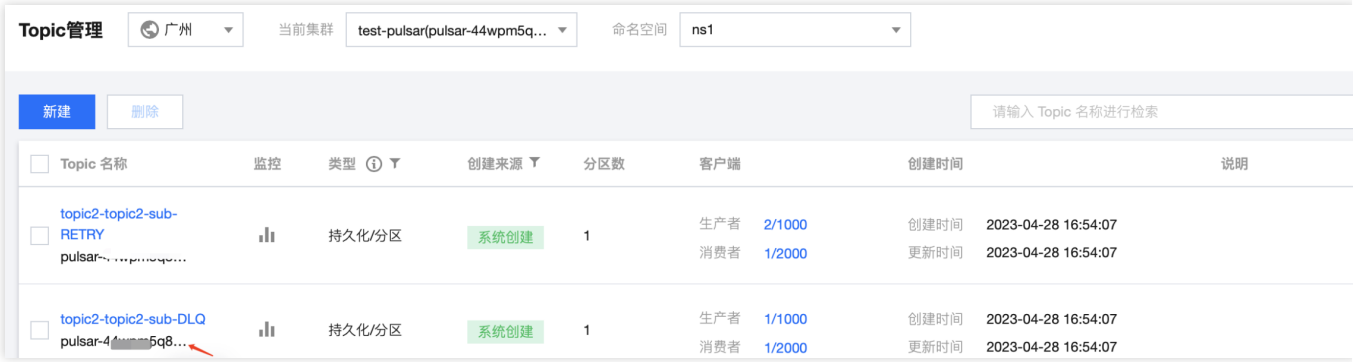


步骤5. 消费死信消息

1. 修改 DeadConsumer.java 参数。

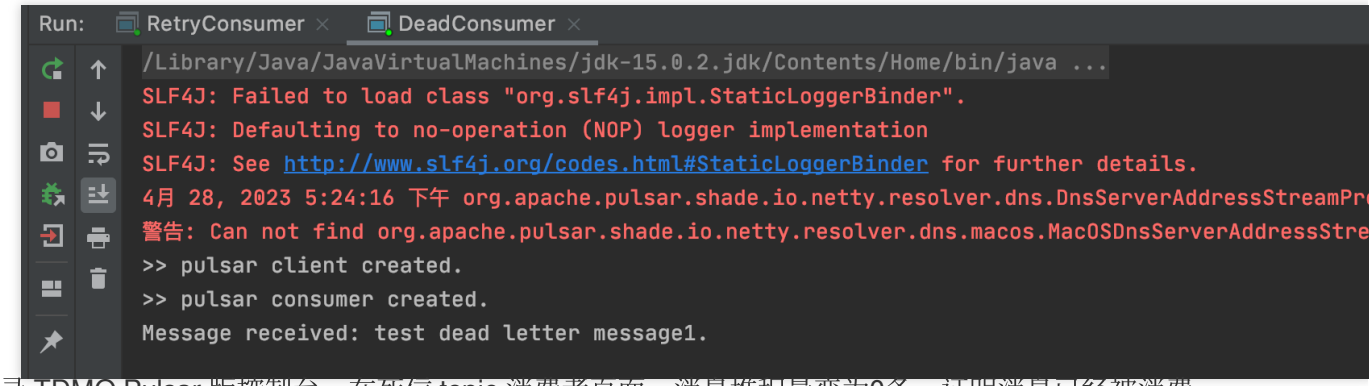


.topic：填写死信topic名称，需要填入完整路径，即 `persistent://clusterid/namespace/Topic`，`clusterid/namespace/topic` 的部分可以从控制台上 Topic 管理页面直接复制。



.subscriptionName：需要写入订阅名，可在 Topic 消费者界面查看。

2. 编译并运行消息重试程序 RetryConsumer.java 。运行结果如下：



3. 登录 TDMQ Pulsar 版控制台，在死信 topic 消费者页面，消息堆积量变为0条，证明消息已经被消费。

pulsar-[redacted]ns1/topic2-topic2-sub-DLQ
生产者
消费者
新建
删除
订阅名称
Topic
监控
状态
订阅模式
消息堆积量
说明
topic2-sub
topic2-topic2-sub-DLQ
在线
共享模式
0
消费连接实例
消费者名称
客户端地址
分区ID
版本
开始时间
c9da3
[redacted]654
0
2.7.2
2023-04-28T17:
消费进度
分区ID
消费速率(条/秒)
消费带宽(字节/秒)
进度差
空洞消息
0
0
0
0
查看
共 1 条
20 条 / 页

使用 Share 订阅模式消费消息

最近更新时间：2024-08-19 16:12:33

操作背景

为了适用不同场景的需求，Pulsar 支持四种订阅模式：Exclusive、Shared、Failover、Key_Shared。
1. Exclusive 独占模式（默认模式）：一个 Subscription 只能与一个 Consumer 关联，只有这个 Consumer 可以接收到 Topic 的全部消息，如果该 Consumer 出现故障了就会停止消费。
2. 共享模式（Shared）：消息通过 round robin 轮询机制（也可以自定义）分发给不同的消费者，并且每个消息仅会被分发给一个消费者。当消费者断开连接，所有被发送给他，但没有被确认的消息将被重新安排，分发给其它存活的消费者。
3. 灾备模式（Failover）：当存在多个 consumer 时，将会按字典顺序排序，第一个 consumer 被初始化为唯一接受消息的消费者。当第一个 consumer 断开时，所有的消息（未被确认和后续进入的）将会被分发给队列中的下一个 consumer。
4. KEY 共享模式（Key_Shared）：当存在多个 Consumer 时，将根据消息的 Key 进行分发，Key 相同的消息只会被分发到同一个消费者。
更多关于订阅模式的介绍请参考文档订阅模式。
本文以官网提供的 Demo 为例，介绍共享模式（Shared）订阅模式功能与使用方式。

操作步骤

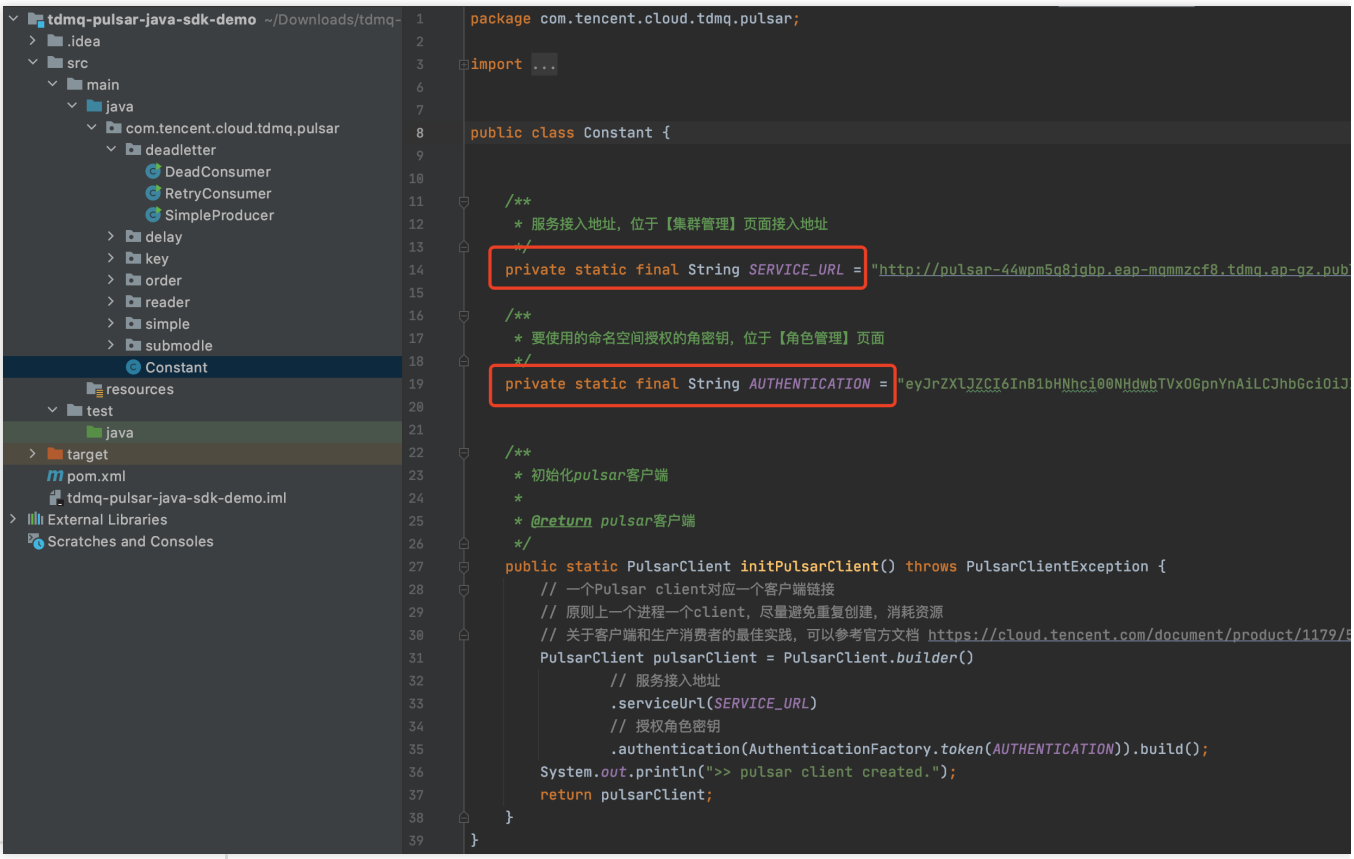
步骤1. 在控制台新建 Pulsar 资源

1. 登录 TDMQ Pulsar 版控制台，创建一个集群和命名空间。
2. 在左侧导航栏选择 **Topic 管理**页签，当前集群和命名空间分别勾选创建好的集群和命名空间。
3. 单击**新建**，输入 Topic 名称和说明，其他选项可保持默认，单击保存，创建一个 topic。

4. 单击操作列的**新增订阅**，为刚新建好的 Topic 创建俩个订阅关系。
5. 单击操作列的**更多** > **查看订阅/消费者**，可看到刚创建好的订阅。

步骤2. 下载 Demo 并配置相关参数

1. 下载官方 Demo 并解压。
2. 修改 Constant.java 参数。

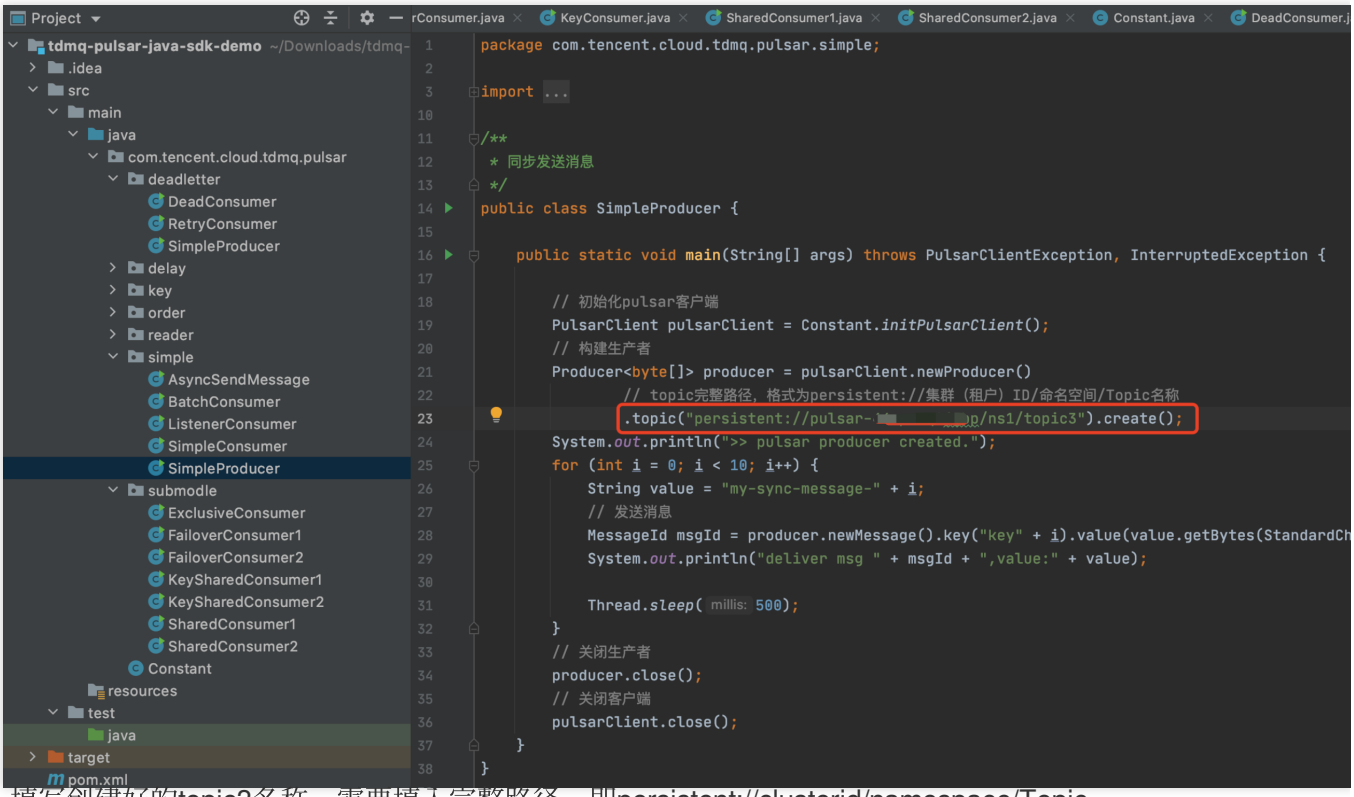


参数	说明
SERVICE_URL	集群接入地址，可以在控制台集群管理页面查看并复制。

AUTHENTICATION	角色的密钥，角色密钥可以在角色管理中复制。

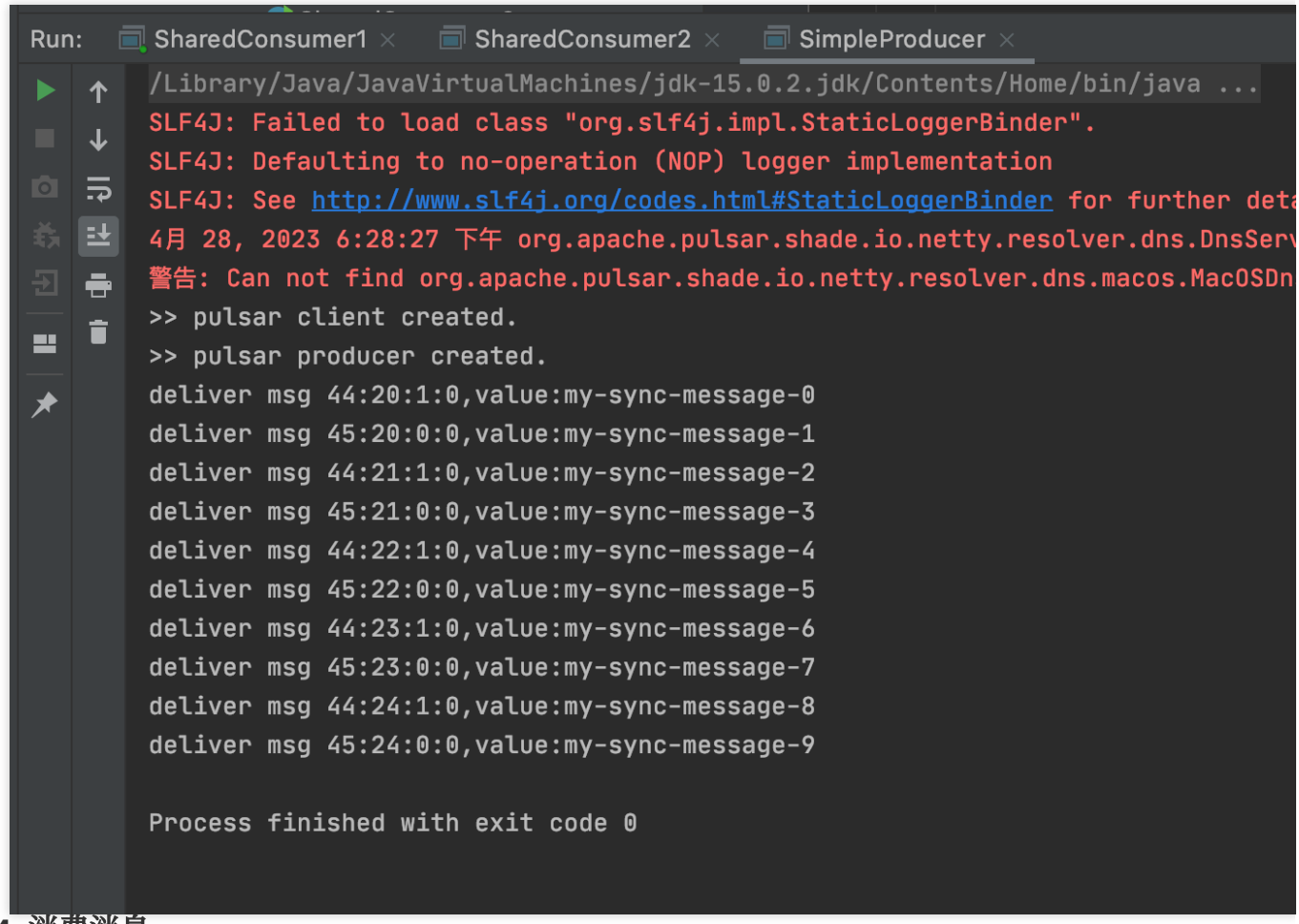
步骤3. 生产消息

1. 进入/simple目录下，修改 SimpleProducer.java 参数。



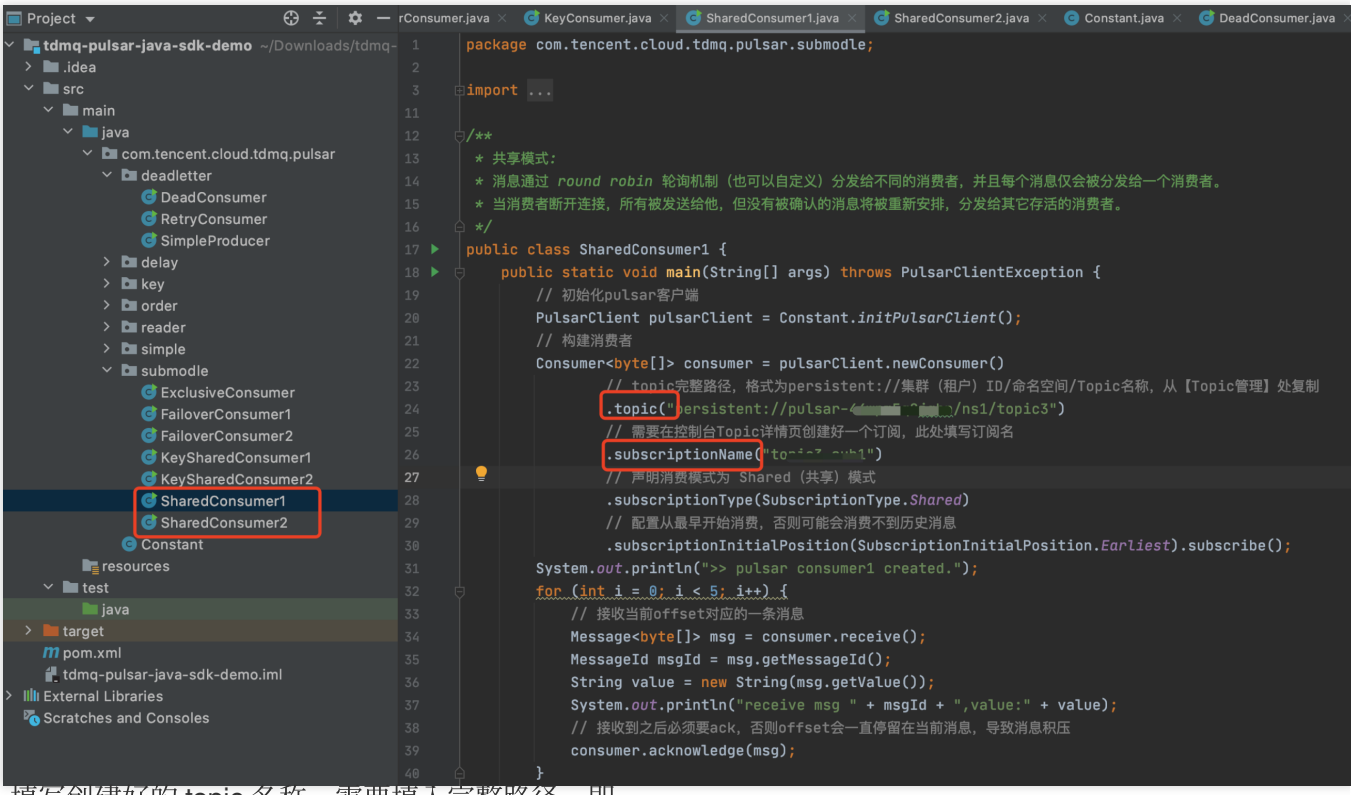
topic：填写创建好的topic3名称，需要填入完整路径，即persistent://clusterid/namespace/Topic，clusterid/namespace/topic 的部分可以从控制台上 Topic管理页面直接复制。

2. 编译并运行 SimpleProducer.java 程序发送消息。运行结果如下，可以看到生产者已经向 topic 中生产了10条消息。



步骤4. 消费消息

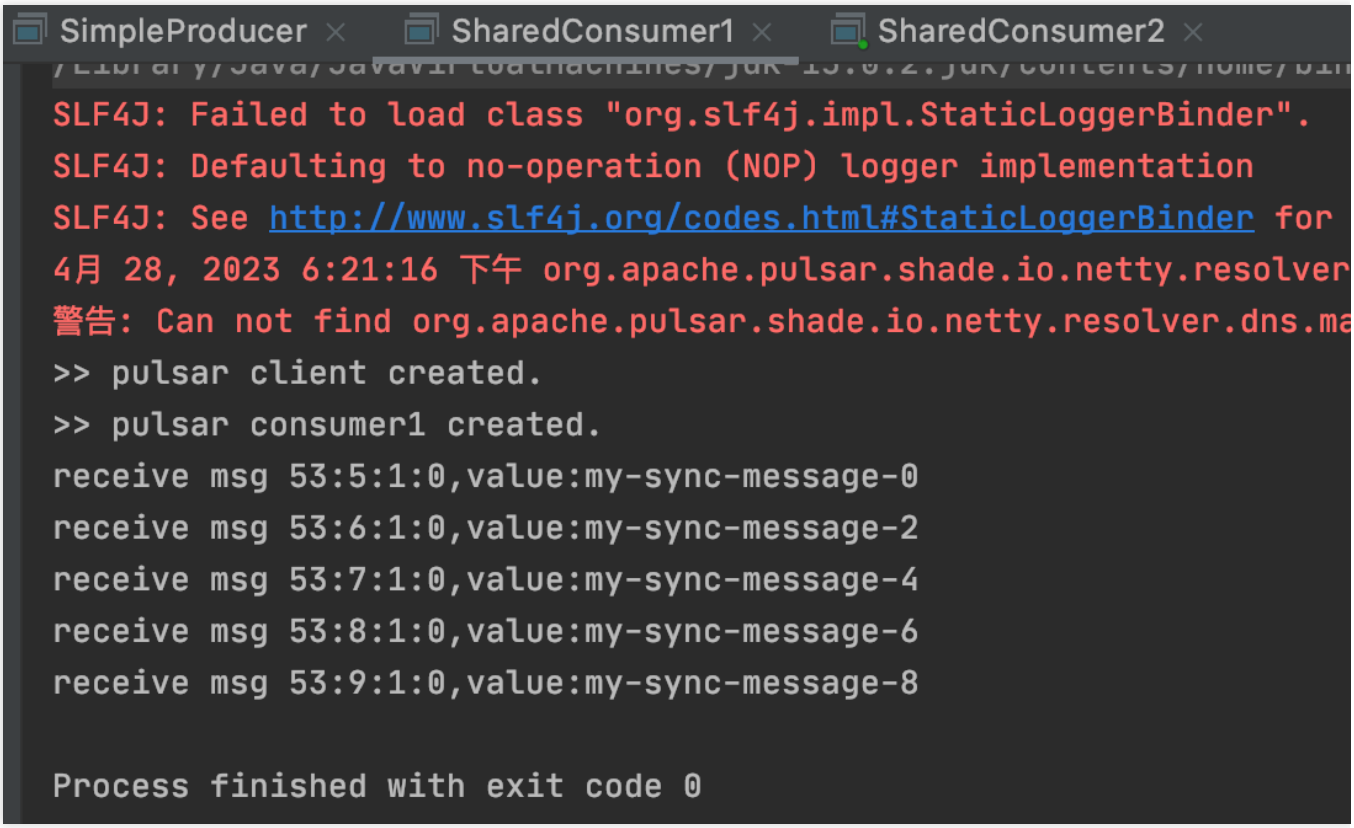
1. 进入/submodle目录下，修改SharedConsumer1.java 和 SharedConsumer1.java 程序参数。

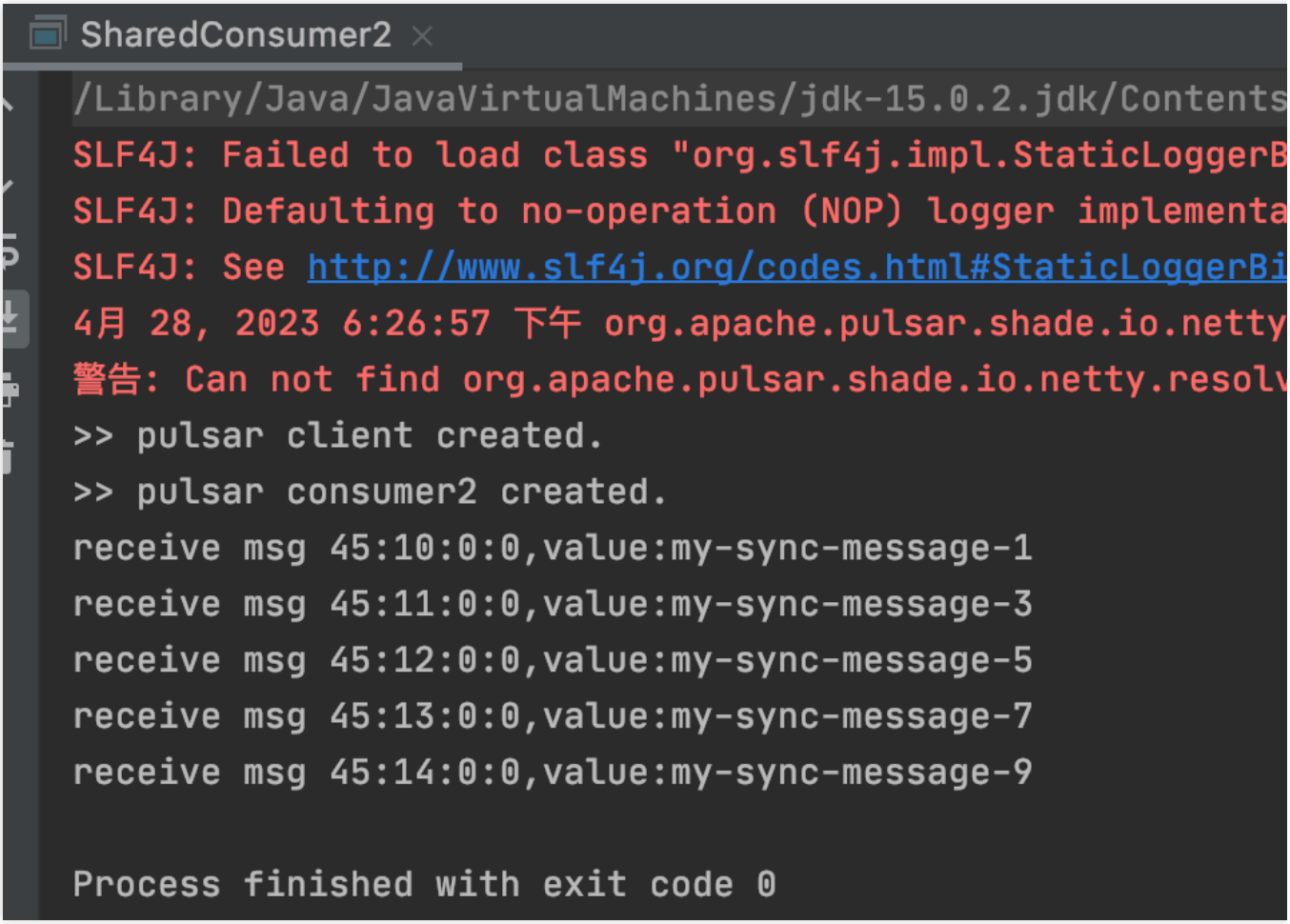


topic：填写创建好的 topic 名称，需要填入完整路径，即 persistent://clusterid/namespace/Topic，`clusterid/namespace/topic` 的部分可以从控制台上 Topic 管理页面直接复制。

.subscriptionName：填写 topic 的订阅名称，可在 Topic 消费者界面查看。SharedConsumer1 中填写 sub1 订阅名称，SharedConsumer2 中填写sub2 订阅名称。

2. 编译并运行消息重试程序 SharedConsumer1.java 和 SharedConsumer2.java 。运行结果如下：


SimpleProducer
SharedConsumer1
SharedConsumer2
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for
4月 28, 2023 6:21:16 下午 org.apache.pulsar.shade.io.netty.resolver
警告: Can not find org.apache.pulsar.shade.io.netty.resolver.dns.ma
>> pulsar client created.
>> pulsar consumer1 created.
receive msg 53:5:1:0,value:my-sync-message-0
receive msg 53:6:1:0,value:my-sync-message-2
receive msg 53:7:1:0,value:my-sync-message-4
receive msg 53:8:1:0,value:my-sync-message-6
receive msg 53:9:1:0,value:my-sync-message-8
Process finished with exit code 0


SharedConsumer2
/Library/Java/JavaVirtualMachines/jdk-15.0.2.jdk/Contents
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerB
SLF4J: Defaulting to no-operation (NOP) logger implementa
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBi
4月 28, 2023 6:26:57 下午 org.apache.pulsar.shade.io.netty
警告: Can not find org.apache.pulsar.shade.io.netty.resolv
>> pulsar client created.
>> pulsar consumer2 created.
receive msg 45:10:0:0,value:my-sync-message-1
receive msg 45:11:0:0,value:my-sync-message-3
receive msg 45:12:0:0,value:my-sync-message-5
receive msg 45:13:0:0,value:my-sync-message-7
receive msg 45:14:0:0,value:my-sync-message-9
Process finished with exit code 0