数据传输服务

数据订阅

产品文档

腾讯云

【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。

文档目录

数据订阅
数据订阅旧版
数据订阅任务示例

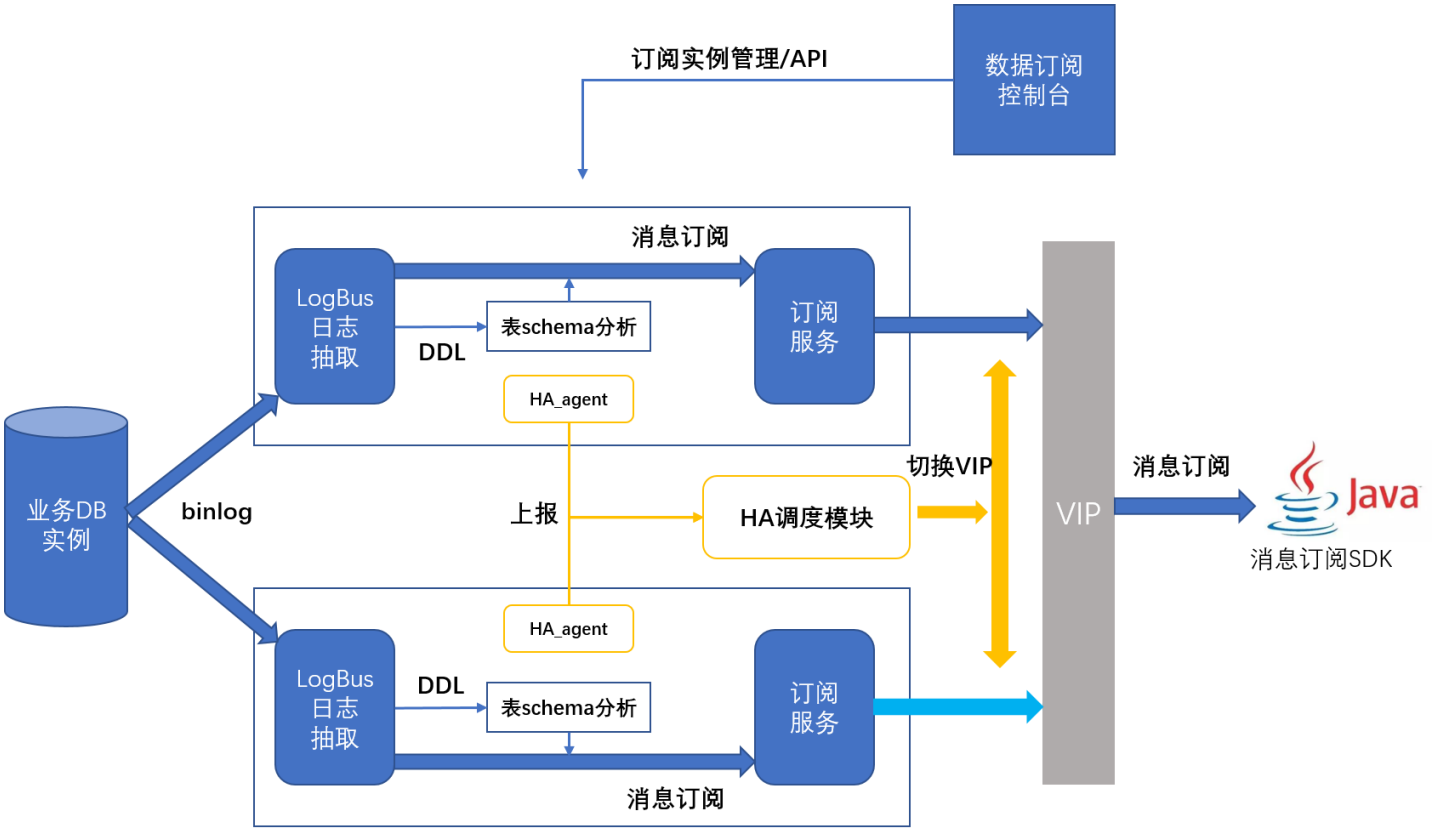
最近更新时间：2021-01-26 18:03:30

数据传输服务 DTS 提供基于 binlog 的增量数据订阅功能，仅需几步简单操作，即可订阅云数据库 TencentDB 的增量更新数据。

1. 在 DTS 控制台 购买并创建 TencentDB 实例的订阅通道。
2. 使用 DTS 数据订阅 SDK 连接订阅通道，订阅并消费增量数据。

订阅原理

数据订阅通过模拟从库向主库获取对应 binlog 内容进行分析，架构图如下，通过解析 binlog，按照订阅通道配置的库表进行分析，对主库几乎无影响。



- 订阅的消息内容目前默认保留最近1天。
- 以实例为单位订阅，后续新增的库表也会在原有订阅通道出现，不需要对原订阅通道进行新增配置操作。
- 以库为单位订阅，后续该库下的新增的表也会在原有订阅通道出现，不需要对原订阅通道进行新增配置操作。此方案下，如果需要新增库，需要进行新增配置操作。

- 数据订阅服务目前支持云数据库 MySQL 5.6、MySQL 5.7。
- 数据订阅暂不支持视图，触发器和外键。
- 数据订阅初次配置时会自动调整相关 binlog_row_image 和 binlog_format 参数，参数调整后，数据订阅会自动 kill 实例中存在的连接使参数立即生效，且数据订阅还会执行 flush tables 获取订阅数据的表结构；这可能会影响您的数据库请求，请您知晓并评估对业务的影响。

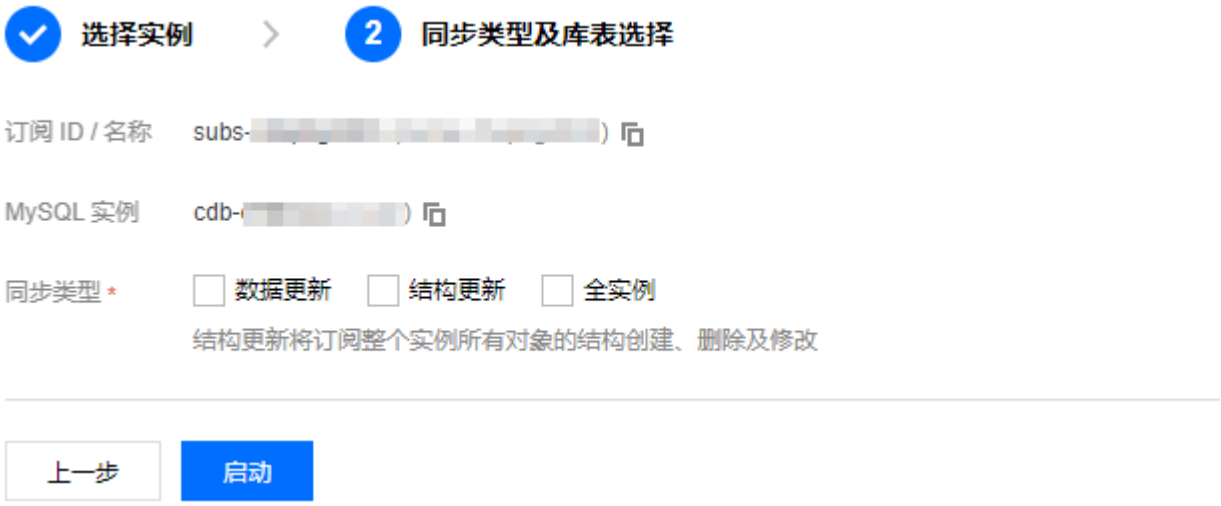
操作步骤

步骤1：创建数据订阅通道

1. 登录 DTS 控制台，在左侧导航选择【数据订阅】，单击【新建数据订阅】。
2. 在新建数据订阅页，选择相应配置，单击【立即购买】。
3. 开通成功后，返回数据订阅列表，单击【初始化配置】，为已购买的数据订阅通道初始化配置。
4. 在配置页，选择源 TencentDB 实例，单击【下一步】。

5. 选择所需的同步类型及库表。
   - 数据订阅的订阅对象粒度细分为库、表，即用户可以选择订阅某些库或者是订阅某几张表。
   - 订阅数据类型细分为数据更新、结构更新、全实例。
     只选择数据更新及订阅对象时，只能订阅到 insert、delete、update 三种数据变更内容。
     如需订阅结构更新（DDL），需要选择同步类型中的结构变更。一旦订阅结构更新，那么 DTS 会将整个 TencentDB 实例所有结构变更拉取出来，用户需要使用 SDK 过滤需要的数据。

选择全实例即数据更新+结构更新。



6. 确定订阅对象和类型后，单击【启动】即可启动订阅通道。

步骤2：修改消费时间起点

DTS 支持在消费的过程中，随时修改消费时间起点。一旦修改完消费时间起点，那么下游 SDK 拉取到的增量数据从修改后的消费时间起点开始。

> 说明：
> DTS 目前只支持在控制台修改消费点，不支持在 SDK 中指定消费点。

修改步骤如下：

1. 停止 SDK 消费进程。
   在 数据订阅 列表，单击订阅 ID 或“操作”列的【查看订阅详情】，进入订阅详情页，“消费者来源（IP）”为无，即为进程已停止。
2. 修改消费时间起点。
   在 数据订阅 列表，将鼠标移至订阅的消费时间起点上，会出现“修改消费时间起点”图标，单击图标进入修改页。

   > 说明：
   > 修改的消费时间起点必须在订阅通道的数据范围之内。

3. 重启 SDK 消费进程。
   修改完消费时间起点后，即可重启本地的 SDK 消费进程，此时 SDK 会从修改的消费点开始订阅增量数据。

步骤3：修改订阅对象

DTS 支持在订阅消费的过程中，动态增加或减少订阅对象。

- 如果增加了订阅对象，那么修改完成后，订阅通道会从当前时间拉取新增订阅对象的增量数据。
- 如果减少了订阅对象，那么修改完成后，SDK 中将无法再订阅到这个对象的数据。

修改步骤如下：

1. 在 数据订阅 列表，选择"操作"列的【更多】>【修改订阅对象】，进入订阅配置页。

2. 在订阅配置页，修改订阅对象，单击【保存】。

步骤4：使用 SDK 消费数据

请参见 SDK 使用指南。

订阅案例

拉取对应订阅表的变化至本地文件
拉取对应订阅表的变化至 Redis
拉取对应订阅表的变化至 Kafka

数据订阅库表到本地

最近更新时间：2020-12-11 10:14:44

本文将以一个简单案例来说明数据订阅中拉取对应表到本地文件的功能，并且提供简易 LocalDemo 下载 。以下操作将在 CentOS 操作系统中完成。

配置环境

- Java 环境配置

```
yum install java-1.8.0-openjdk-devel
```

- 数据订阅 SDK 下载

获取密钥

登录 访问管理控制台 获取密钥。

选择数据订阅

1. 登录 DTS 控制台，选择左侧的【数据订阅】，进入数据订阅页面。
2. 选择需同步的 TencentDB 实例名，然后单击启动，再返回数据订阅，单击您所创建的数据订阅。详细介绍请参考 如何获取数据订阅。
3. 查看对应的 DTS 通道、IP 和 port，然后结合之前的密钥填写到对应 LocalDemo.java 里面。

```
// 从云API获取密钥,填写到此处
context.setSecretId("AKIDfdsfdsfsdt1331431sdfds"); 请填写您从云API获取的secretID
context.setSecretKey("test111usdfsdfsddsfRkeT"); 请填写
您从云API获取的secretKey
// 在数据迁移服务里面通过数据订阅获取到对应的ip,port,填写到此处
context.setServiceIp("10.66.112.181"); 请填写您从数据订阅配置获取到的IP
context.setServicePort(7507);
请填写您从数据订阅配置获取到的PORT
final DefaultSubscribeClient client = new DefaultSubscribeClient(context);
// 填写对应要同步的数据库和表名,并修改对应要落地存储的文件名.
final String targetDatabase = "test"; 填写您所要订阅的库名
```

```
final String targetTable = "alantest"; 填写您所要订阅的表名
final String fileName = "test.alan.txt"; 填写您希望落在本地的文件名

client.addClusterListener(listener);
// 通过数据迁移订阅的配置选项获取到dts-channel的配置信息,填写到此处
client.askForGUID("dts-channel-e4FQxtYV3It4test"); 请填写您从数据订阅获取的通道dts的名称
client.start();
```

编译操作和检验

1.
```
javac -classpath binlogsdk-2.6.0-release.jar -encoding UTF-8 LocalDemo.java
```

2. 然后执行启动，如果没有异常报错就是正常在服务了，然后查看对应之前设置的落地文件。

```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.6.0-release.jar LocalDemo
```

3. 查看对应之前我们设置的 test.alan.txt 能看到已经有数据拉取到了本地。

```
[root@VM_71_10_centos ~]# cat test.alan.txt
checkpoint:144251@3@357589@317713
record_id:000001000000000004D9110000000000000001
record_encoding:utf8
fields_enc:latin1,utf8
gtid:4f21864b-3bed-11e8-a44c-5cb901896188:1562
source_category:full_recorded
source_type:mysql
table_name:alantest
record_type:INSERT
db:test
timestamp:1523356039
primary:
Field name: id
Field type: 3
Field length: 2
```

```
Field value: 26
Field name: name
Field type: 253
Field length: 4
Field value: alan
```

数据订阅库表到 Redis

最近更新时间：2021-07-22 17:25:56

本文将以一个简单案例来说明数据订阅中拉取对应表到 Redis 的功能，并且提供简易 RedisDemo 下载 。以下操作将在 CentOS 操作系统中完成。

配置环境

- Java 环境配置

```
yum install java-1.8.0-openjdk-devel
```

- 数据订阅 SDK 下载
- jedis-2.9.0.jar 下载

获取密钥

登录 访问管理控制台 获取密钥。

选择数据订阅

1. 登录 DTS 控制台，选择左侧的【数据订阅】，进入数据订阅页面。
2. 选择需同步的 TencentDB 实例名，然后单击启动，再返回数据订阅，单击您所创建的数据订阅。详细介绍请参考 数据订阅。
3. 查看对应的 DTS 通道、IP 和 Port，然后结合之前的密钥填写到对应 RedisDemo.java 里面。

```
context.setSecretId("AKIDfdsfdsfsdt1331431sdfds"); 请填写您从云 API 获取的 secretID
context.setSecretKey("test111usdfsdfsddsfRkeT"); 请填写您从云 API 获取的 secretKey
// 在数据迁移服务里面通过数据订阅获取到对应的 IP,PORT,填写到此处
context.setServiceIp("10.66.112.181"); 请填写您从数据订阅配置获取到的 IP
context.setServicePort(7507); 请填写您从数据订阅配置获取到的 PORT
// 创建消费者
//SubscribeClient client=new DefaultSubscribeClient(context,true);
final DefaultSubscribeClient client = new DefaultSubscribeClient(context);
final Jedis jedis = new Jedis("127.0.0.1", 6379); 请填写您对应的 Redis 主机和端口
final String targetDatabase = "test"; 填写您所要订阅的库名
final String targetTable = "alantest"; 填写您所要订阅的表名，表有2个字段分别是 id,name（i
```

```
d 是做了主键)
// 创建订阅监听者 listener
ClusterListener listener = new ClusterListener() {
@Override
public void notify(List<ClusterMessage> messages) throws Exception {
// System.out.println("--------------------:" + messages.size());
for(ClusterMessage m:messages){
DataMessage.Record record = m.getRecord();
//过滤不感兴趣的订阅信息
if(!record.getDbName().equalsIgnoreCase(targetDatabase) || !record.getTablename().eq
ualsIgnoreCase(targetTable)){
//注意：对于不感兴趣的信息也必须 Ack
m.ackAsConsumed();
continue;
}
if(record.getOpt() != DataMessage.Record.Type.BEGIN && record.getOpt() != DataMessag
e.Record.Type.COMMIT){
List<DataMessage.Record.Field> fields = record.getFieldList();
//INSERT RECORD
//String pk = record.getPrimaryKeys();

if(record.getOpt() == DataMessage.Record.Type.INSERT){

String keyid="";
String value="";
for (DataMessage.Record.Field field : fields) {
//先获取 id 值,需要有 primary key,然后找到名为 name 的列，赋值给 Redis 插入 key 和 name
对应的 value
if(field.getFieldname().equalsIgnoreCase("id")){
keyid=field.getValue();
continue;
}
if(field.getFieldname().equalsIgnoreCase("name")){
value=field.getValue();

}
jedis.set(keyid, value);
```

```
}
}
```

编译操作与检验

1. 使用 javac 命令进行编译。

```
[root@VM_71_10_centos ~]# javac -classpath binlogsdk-2.6.0-release.jar:jedis-2.9.0.jar -encoding UTF-8 RedisDemo.java
```

2. 执行启动，如果没有异常报错就是正常在服务了，然后查看对应之前设置的落地文件。

```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.6.0-release.jar:jedis-2.9.0.jar RedisDemo
```

3. 查看进行数据库插入和 update 操作，并从 redis 观察发现确实插入并修改成功了,最后进行 delete 操作，redis 对应的数据也被删除掉了。

```
MySQL [test]> insert into alantest values(1001,'alan1');
Query OK, 1 row affected (0.00 sec)
MySQL [test]> update alantest set name='alan2' where id=1001;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
————————————————————
127.0.0.1:6379> get 1001
"alan2"
MySQL [test]> update alantest set name='alan3' where id=1001;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
————————————————————
127.0.0.1:6379> get 1001
"alan3"
MySQL [test]> delete from alantest where id=1001;
Query OK, 1 row affected (0.00 sec)
————————————————————
127.0.0.1:6379> get 1001
(nil)
```

数据订阅库表到 Kafka

最近更新时间：2020-12-11 16:41:47

本文以一个简单案例来说明数据订阅中拉取对应表到 Kafka 的功能，并提供简易 KafkaDemo 下载。

配置环境

- 操作系统：CentOS
- 相关下载
  - 数据订阅 SDK
  - SLF4J 组件
  - Kafka-clients
- Java 环境配置

```
yum install java-1.8.0-openjdk-devel
```

安装 Kafka

1. 请参考 Kafka 介绍 安装 Kafka。
2. 启动之后创建一个 testtop 主题。

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --p
artitions 1 --topic testtop
Created topic "testtop".
```

获取密钥

登录 访问管理控制台 获取密钥。

选择数据订阅

1. 登录 DTS 控制台，选择左侧的【数据订阅】，进入数据订阅页面。
2. 在订阅列表，单击订阅名，进入订阅详情页，查看对应的通道 ID、服务 IP 和服务端口。
3. 结合之前的密钥填写到对应 KafkaDemo.java 里。

```
import com.qcloud.dts.context.SubscribeContext;
import com.qcloud.dts.message.ClusterMessage;
import com.qcloud.dts.message.DataMessage;
import com.qcloud.dts.subscribe.ClusterListener;
import com.qcloud.dts.subscribe.DefaultSubscribeClient;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.StringSerializer;
import org.apache.log4j.Logger;
import java.util.List;
import java.util.Properties;
public class KafkaDemo {
public static void main(String[] args) throws Exception {
//初始化一个 kafka producer
final String TOPIC = "testtop";
Properties props = new Properties();
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "10.168.1.6:9092");
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
final Producer<String, String> producer = new KafkaProducer<String, String>(props);
// 创建一个 context
SubscribeContext context = new SubscribeContext();
context.setSecretId("AKIDPko5fVtvTDE0WffffkCwd4NzKcdePt79uauy");
context.setSecretKey("ECtY8F5e2QqtdXAe18yX0EBqK");
// 订阅通道所在 region
context.setRegion("ap-beijing");
final DefaultSubscribeClient client = new DefaultSubscribeClient(context);
// 创建订阅监听者 listener
ClusterListener listener = new ClusterListener() {
@Override
public void notify(List<ClusterMessage> messages) throws Exception {
System.out.println("--------------------:" + messages.size());
for(ClusterMessage m:messages){
DataMessage.Record record = m.getRecord();
if(record.getOpt() != DataMessage.Record.Type.BEGIN && record.getOpt() != DataMessag
```

```
e.Record.Type.COMMIT){
List<DataMessage.Record.Field> fields = record.getFieldList();
//打印每个列的信息
for (int i = 0; i < fields.size(); i++) {
DataMessage.Record.Field field = fields.get(i);
System.out.println("Database Name:" + record.getDbName());
System.out.println("Table Name:" + record.getTablename());
System.out.println("Field Value:" + field.getValue());
System.out.println("Field Value:" + field.getValue().length());
System.out.println("Field Encoding:" + field.getFieldEnc());
}
//将整个 record 发送到指定的 kafka topic 中
System.out.println("Record++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++");
producer.send(new ProducerRecord<String, String>(TOPIC, record.toString()));
}
m.ackAsConsumed();
}
}
@Override
public void onException(Exception e){
System.out.println("listen exception" + e);
}
};
// 添加监听者
client.addClusterListener(listener);
client.askForGUID("dts-channel-p15e9eW9rn8hA68K");
client.start();
}
}
```

编译操作与检验

1. 编译客户端程序 KafkaDemo.java。

```
javac -classpath binlogsdk-2.9.1-jar-with-dependencies.jar:slf4j-api-1.7.25.jar:sl
f4j-log4j12-1.7.2.jar:kafka-clients-1.1.0.jar -encoding UTF-8 KafkaDemo.java
```

2. 执行启动，如无异常报错即为正常在服务。

```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.9.1-jar-with-dependencies.jar:kafka-clients-1.1.0.jar:slf4j-api-1.7.25.jar:slf4j-log4j12-1.7.2.jar KafkaDemo
```

3. 通过对表 alantest 插入一条数据，发现在 Kafka 订阅的 testtop 里面能看到已经有数据过来了。

```
MySQL [test]> insert into alantest values(123456,'alan');
Query OK, 1 row affected (0.02 sec)
[root@VM_71_10_centos kafka_2.11-1.1.0]# bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic testtop --from-beginning
checkpoint:144251@3@1275254@1153089
record_id:00000100000000001198410000000000000001
record_encoding:utf8
fields_enc:latin1,utf8
gtid:4f21864b-3bed-11e8-a44c-5cb901896188:5552
source_category:full_recorded
source_type:mysql
table_name:alantest
record_type:INSERT
db:test
timestamp:1524649133
primary:id
Field name: id
Field type: 3
Field length: 6
Field value: 123456
Field name: name
Field type: 253
Field length: 4
Field value: alan
```

数据订阅 Kafka 版
消费订阅数据
新增消费组

最近更新时间：2021-01-26 18:03:55

数据订阅 Kafka 版支持用户创建多个消费组，进行多点消费。通过创建多个消费组，您可以进行多次消费，增加消费通道降低使用成本，提升消费数据速度。

前提条件

已创建 数据订阅 Kafka 版。

> 说明：
> 数据订阅列表中，带有“kafka 版”标志的订阅即为数据订阅 Kafka 版，如下图：
>
> 
>

注意事项

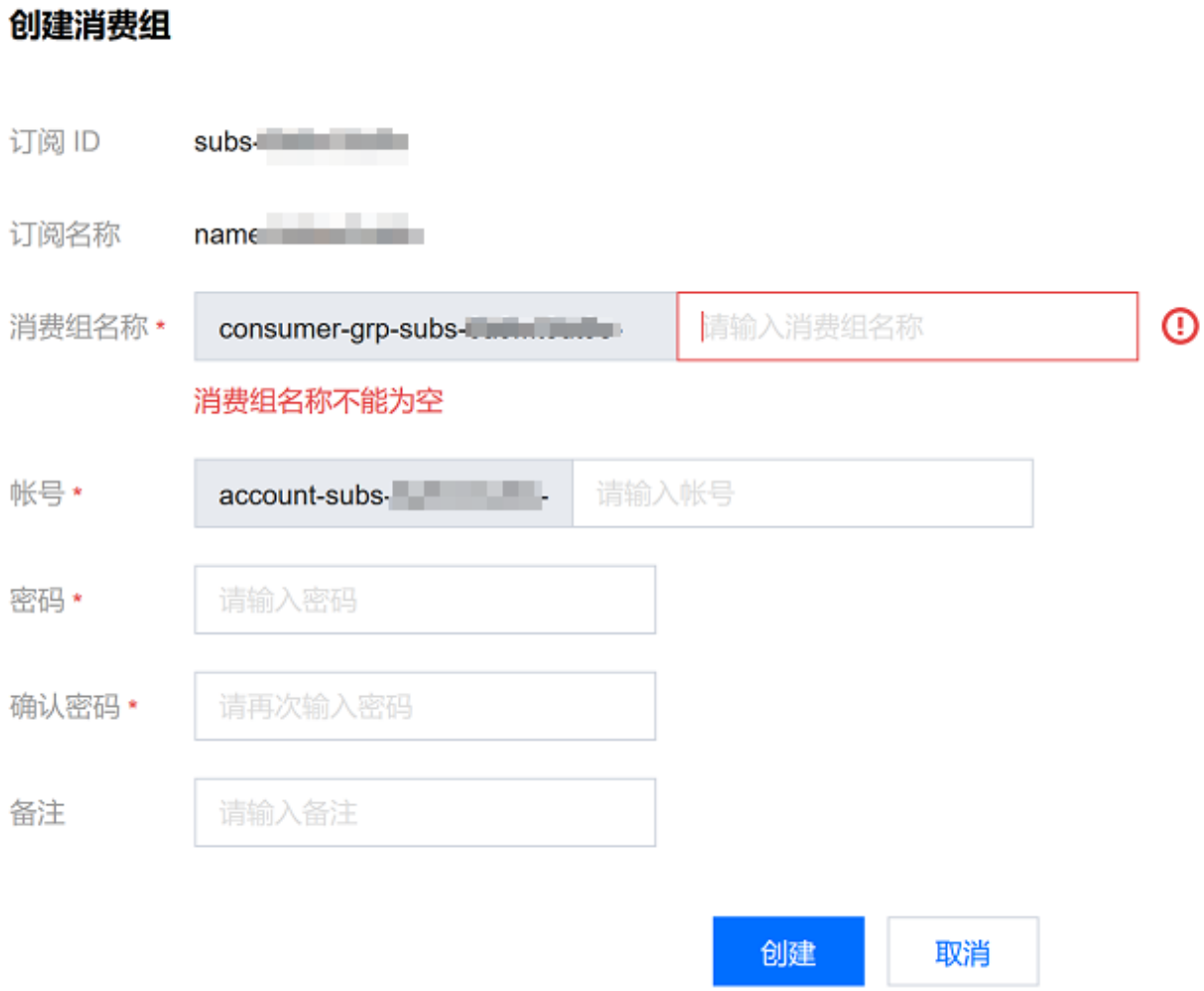
- 数据订阅任务须为运行中状态。
- 一个数据订阅任务最多创建10个消费组。
- 一个消费组只能有一个消费者进行消费。

操作步骤

1. 登录 DTS 控制台，在左侧导航选择【数据订阅】，进入数据订阅页面。
2. 在数据订阅列表，选择所需的数据订阅，单击订阅名或“操作”列的【查看订阅详情】，进入订阅管理页面。

3. 在订阅管理页面，选择【消费管理】页，单击【新增消费组】。

4. 在弹窗的新建消费对话框，设置消费组信息，单击【创建】。
   - 消费组名称：根据实际情况创建消费组名称，方便使用和识别。
   - 帐号：设置消费组帐号。
   - 密码：设置消费组帐号对应密码。
   - 确认密码：再次输入相同密码。
   - 备注：设置备注信息，方便备注信息记录。



管理消费组

最近更新时间：2021-01-26 18:05:53

数据订阅 Kafka 版支持对消费组进行管理，包括修改密码和删除操作。 本文为您介绍如何通过控制台修改消费组密码和删除消费组。

前提条件

已创建 消费组。

修改消费组密码

1. 登录 DTS 控制台，在左侧导航选择【 数据订阅 】，进入数据订阅页面。
2. 在数据订阅列表，选择所需的数据订阅，单击订阅名或“操作”列的【 查看订阅详情 】，进入订阅管理页面。
3. 在订阅管理页面，选择【 消费管理 】页，单击”操作“列的【 修改密码 】。

4. 在弹出的对话框，修改密码，单击【 修改 】。

删除消费组

1. 登录 DTS 控制台，在左侧导航选择【 数据订阅 】，进入数据订阅页面。
2. 在数据订阅列表，选择所需的数据订阅，单击订阅名或“操作”列的【 查看订阅详情 】，进入订阅管理页面。
3. 在订阅管理页面，选择【 消费管理 】页，单击“操作”列的【 删除 】。
4. 在弹出的对话框，确认无误后，单击【 确定 】。

> ⚠ 注意：
> 删除消费组后，该消费组中的消费点位将被删除，但数据订阅中的数据不会被删除，请谨慎操作。

使用 Kafka 客户端消费订阅数据

最近更新时间：2021-08-03 10:56:30

数据订阅 Kafka 版中，您可以直接通过0.11版本及以上的 Kafka 客户端进行消费订阅数据，本文为您提供了 Java、Go、Python 语言的客户端消费 Demo。

消费 Demo 下载（云数据库 MySQL、MariaDB）

参考下表下载数据订阅 Kafka 版客户端消费 Demo 代码：

Demo 语言	下载地址
Go	地址
Java	地址
Python3	地址

消费 Demo 下载（TDSQL MySQL版）

参考下表下载数据订阅 Kafka 版客户端消费 Demo 代码：

Demo 语言	下载地址
Go	地址
Java	地址
Python	地址

Protobuf 协议文件下载

协议文件	下载地址
Protobuf	地址

配置参数说明

参数	说明

参数	说明
brokerlist	数据订阅 Kafka 的内网访问地址
topic	数据订阅通道的订阅 topic
group	消费组名称
user	消费组账号名
password	消费组密码
trans2sql	是否转换为 SQL 语句

Demo 关键逻辑讲解

生产逻辑

下文对消息生产逻辑进行简要说明，助于用户理解消费逻辑。
我们采用 Protobuf 进行序列化，各语言 Demo 中均附带有 Protobuf 定义文件。文件中定义了几个关键结构：Envelope 是最终发送的 Kafka 消息结构；Entry 是单个订阅事件结构；Entries 是多个 Entry 的集合。

生产过程如下：

1. 拉取 Binlog 消息，将每个 Binlog Event 编码为一个 Entry 结构体。

```
message Entry {
Header header = 1; //事件的头部
Event event = 2;
}
message Header {
int32 version = 1;
SourceType sourceType = 2; //源库的类型信息，包括 mysql，oracle 等类型
MessageType messageType = 3; //消息的类型
uint32 timestamp = 4; //Event在原始 binlog 中的时间戳
int64 serverId = 5; //源的 serverId
string fileName = 6; //源 binlog 的文件名称
uint64 position = 7; //事件在源 binlog 文件中的偏移量
string gtid = 8; //当前事务的 gtid
string schemaName = 9; //变更影响的 schema
string tableName = 10; //变更影响的 table
uint64 seqId = 11; //如果 event 分片，同一分片的 seqId 一致
uint64 eventIndex = 12; //大的 event 分片，序号从0开始，当前版本无意义，留待后续扩展用
```

```
bool isLast = 13; //当前 event 是不是 event 分片的最后一块，是则为 true，当前版本无意义，留待后续扩展用
repeated KVPair properties = 15;
}
message Event {
BeginEvent beginEvent = 1;
DMLEvent dmlEvent = 2;
CommitEvent commitEvent = 3;
DDLEvent ddlEvent = 4;
RollbackEvent rollbackEvent = 5;
HeartbeatEvent heartbeatEvent = 6;
CheckpointEvent checkpointEvent = 7;
repeated KVPair properties = 15;
}
```

2. 为减少消息量，将多个 Entry 合并，合并后的结构为 Entries，Entries.items 字段即为 Entry 顺序列表。合并的数量以合并后不超过 Kafka 单个消息大小限制为标准。对单个 Event 就已超过大小限制的，则不再合并，Entries 中只有唯一 Entry 。

```
message Entries {
repeated Entry items = 1; //entry list
}
```

3. 对 Entries 进行 Protobuf 编码得到二进制序列。
4. 将 Entries 的二进制序列放入 Envelope 的 data 字段。当存在单个 Binlog Event 过大时，二进制序列可能超过 Kafka 单个消息大小限制，此时我们会将其分割为多段，每段装入一个 Envelope。Envelope.index 和 Evelope.total 分别记录总段数和当前 Envelope 的序号（从0开始）。

```
message Envelope {
int32 version = 1; //protocol version, 决定了 data 内容如何解码
uint32 total = 2;
uint32 index = 3;
bytes data = 4; //当前 version 为1, 表示 data 中数据为 Entries 被 PB 序列化之后的结果，通过 PB 反序列化可以得到一个 Entries 对象
repeated KVPair properties = 15;
}
```

5. 对上一步生成的一个或多个 Envelope 依次进行 Protobuf 编码，然后投递到 Kafka 分区。同一个 Entries 分割后的多个 Envelope 顺序投递到同一个分区。

消费逻辑

下文对消费逻辑进行简要说明。我们提供的三种语言的 Demo 均遵循相同的流程。

1. 创建 Kafka 消费者。
2. 启动消费。
3. 依次消费原始消息，并根据消息中的分区找到分区对应的 partitionMsgConsumer 对象，由该对象对消息进行处理。
4. partitionMsgConsumer 将原始消息反序列化为 Envelope 结构。

```
message Envelope {
int32 version = 1; //protocol version, 决定了 data 内容如何解码
uint32 total = 2;
uint32 index = 3;
bytes data = 4; //当前 version 为1, 表示 data 中数据为 Entries 被 PB 序列化之后的结果,
通过 PB 反序列化可以得到一个 Entries 对象
repeated KVPair properties = 15;
}
```

5. partitionMsgConsumer 根据 Envelope 中记录的 index 和 total 连续消费一条或者多条消息，直到 Envlope.index 等于 Envelope.total-1（参见上面消费生产逻辑，表示收到了一个完整的 Entries ）。
6. 将收到的连续多条 Envelope 的 data 字段顺序组合到一起。将组合后的二进制序列用 Protobuf 解码为 Entries 。

```
message Entries {
repeated Entry items = 1; //entry list
}
```

7. 对 Entries.items 依次处理，打印原始 Entry 结构或者转化为 SQL 语句。

```
message Entry {
Header header = 1; //事件的头部
Event event = 2;
}
message Header {
int32 version = 1;
```

```
SourceType sourceType = 2; //源库的类型信息，包括 mysql，oracle 等类型
MessageType messageType = 3; //消息的类型
uint32 timestamp = 4; //Event在原始 binlog 中的时间戳
int64 serverId = 5; //源的 serverId
string fileName = 6; //源 binlog 的文件名称
uint64 position = 7; //事件在源 binlog 文件中的偏移量
string gtid = 8; //当前事务的 gtid
string schemaName = 9; //变更影响的 schema
string tableName = 10; //变更影响的 table
uint64 seqId = 11; //如果 event 分片，同一分片的 seqId 一致
uint64 eventIndex = 12; //大的 event 分片，序号从0开始，当前版本无意义，留待后续扩展用
bool isLast = 13; //当前 event 是不是 event 分片的最后一块，是则为 true，当前版本无意义，留待后续扩展用
repeated KVPair properties = 15;
}
message Event {
BeginEvent beginEvent = 1;
DMLEvent dmlEvent = 2;
CommitEvent commitEvent = 3;
DDLEvent ddlEvent = 4;
RollbackEvent rollbackEvent = 5;
HeartbeatEvent heartbeatEvent = 6;
CheckpointEvent checkpointEvent = 7;
repeated KVPair properties = 15;
}
```

Java Demo 使用说明

编译环境：Maven 或者 Gradle 包管理工具，JDK8。
运行环境：腾讯云服务器（需要与订阅实例相同地域，才能够访问到 Kafka 服务器的内网地址），安装 JRE8 。
操作步骤：

1. **创建新版数据订阅通道，详情请参见 数据订阅 Kafka版。**
2. **创建一个或多个消费组，详情请参见 新增消费组。**
3. **下载 Java Demo ，然后解压该文件。**
4. **进入解压后的目录，为方便使用，目录下分别放置了 Maven 模型文件、pom.xml文件和 Gradle 相关配置文件，用户根据需要选用。**
   - **使用 Maven 进行打包：mvn clean package 。**

- 使用 Gradle 进行打包：gradle fatJar 打包并包含所有依赖，或者 gradle jar 进行打包。

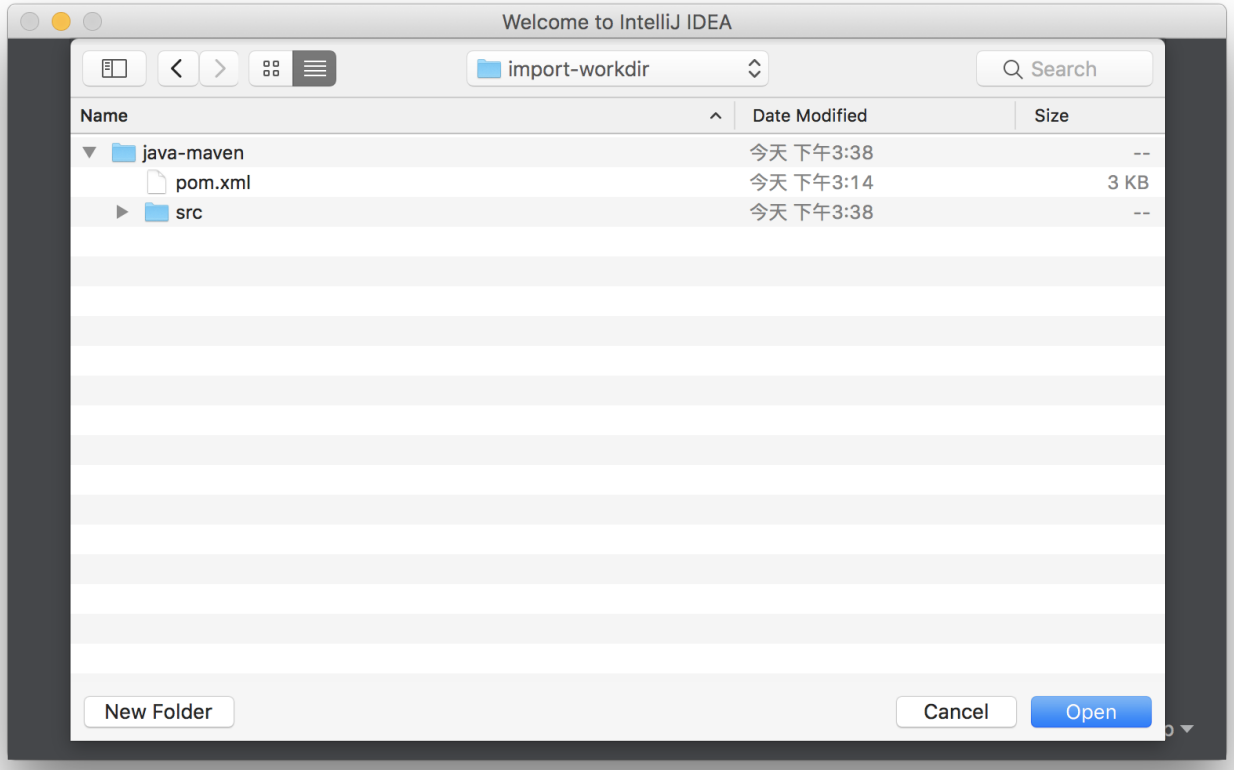
5. 运行：使用 Maven 打包后，进入目标文件夹 target ，运行 `java -jar sub_demo-1.0-SNAPSHOT-jar-with-dependencies.jar --brokers=xxx --topic=xxx --group=xxx--user=xxx --password=xxx --trans2sql`。
   使用 Gradle 打包后，进入文件夹 build/libs ，运行 `java -jar sub_demo-with-dependencies-1.0-SNAPSHOT.jar --brokers=xxx --topic=xxx --group=xxx--user=xxx --password=xxx --trans2sql`。
6. 观察消费情况。

```
BEGIN
-->[partition: 0, offset: 87008, partitionSeq: 87009] [mysql-bin.000004:24574], happenedAt: 2021-03-01T
17:49:14
INSERT INTO `kafka-subscribe`.`table1` VALUES (_binary'subscribe-kafka', 61)
-->[partition: 0, offset: 87008, partitionSeq: 87009] [mysql-bin.000004:24605], happenedAt: 2021-03-01T
17:49:14
COMMIT
```

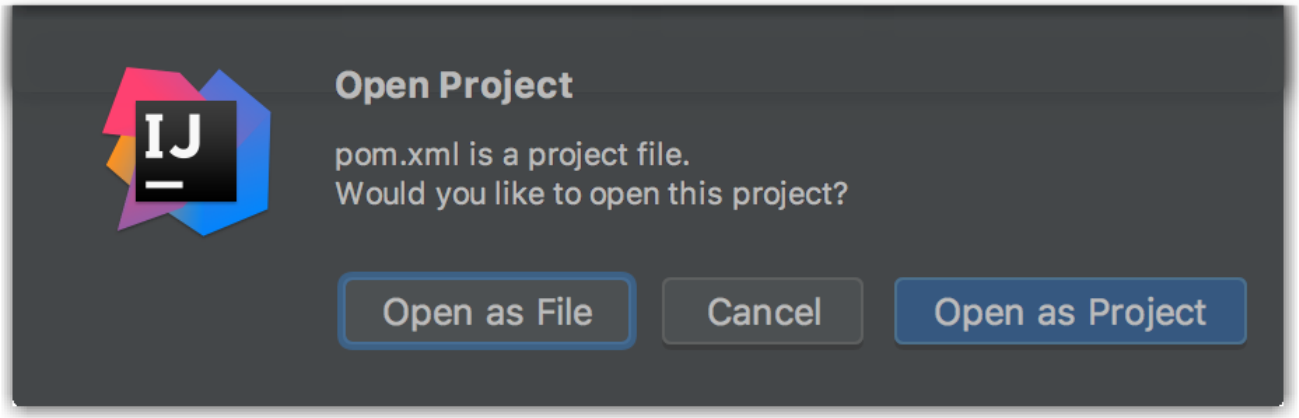
用户也可以使用 IDE 进行编译打包，以 IntelliJ IDEA 软件为例：

1. 打开 IntelliJ IDEA 软件，然后单击【Open】。

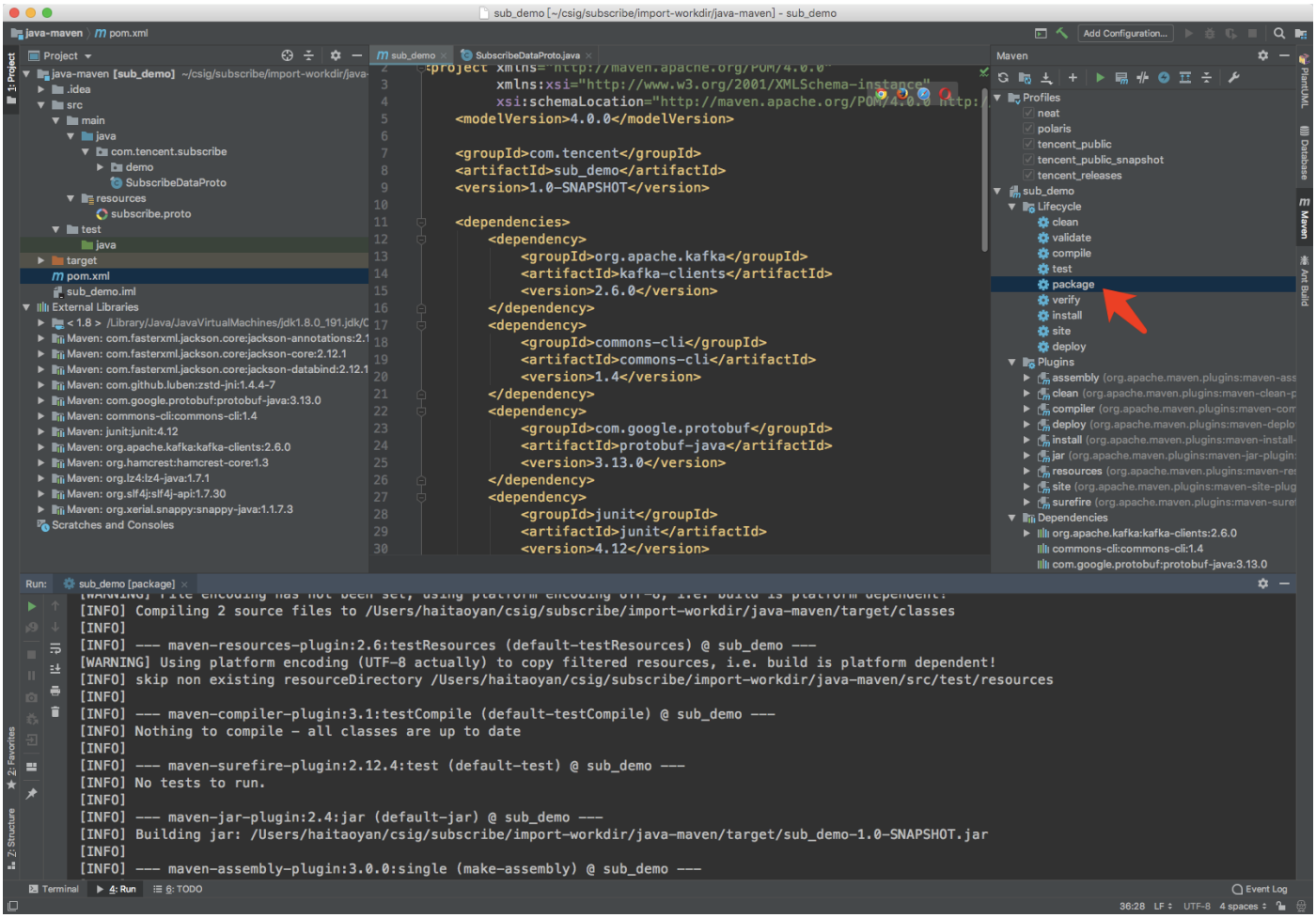
2. 在弹出的对话框，找到 Demo 解压的目录，参照下图找到目录下的 pom.xml 文件，单击【Open】。



3. 在弹出对话框，单击【Open as Project】。



4. 在打开的窗口，双击右侧 Maven 管理窗口 Lifecycle 中的 package，进行打包。



打包完成后。工程根目录 target 文件夹下的 sub_demo-1.0-SNAPSHOT-jar-with-dependencies 即为一个包含了所需依赖的可运行的 jar 包。

对于 Gradle 也是相似的操作流程。

Golang Demo 使用说明

编译环境：Golang 1.12 及以上版本，配置好 Go Module 环境。

运行环境：腾讯云服务器（需要与订阅实例相同地域，才能够访问到 Kafka 服务器的内网地址）。

操作步骤：

1. 创建新版数据订阅通道，详情请参见 数据订阅 Kafka版。
2. 创建一个或多个消费组，详情请参见 新增消费组。
3. 下载 Golang Demo，然后解压该文件。
4. 进入解压后的目录，运行`go build -o subscribe ./main`，生成可执行文件 subscribe。
5. 运行 `./subscribe --brokers=xxx --topic=xxx --group=xxx --user=xxx \--password=xxx --trans2sql=true`。

6. 观察消费情况。

```
BEGIN
-->[partition: 0, offset: 86991, partitionSeq: 86992] [mysql-bin.000004:24272], happenedAt: 2021-03-01
17:47:49 +0800 CST
INSERT INTO `kafka-subscribe`.`table1` VALUES (_binary'subscribe-kafka', 60)
-->[partition: 0, offset: 86991, partitionSeq: 86992] [mysql-bin.000004:24303], happenedAt: 2021-03-01
17:47:49 +0800 CST
COMMIT
```

Python3 Demo 使用说明

编译运行环境：腾讯云服务器（需要与订阅实例相同地域，才能够访问到 Kafka 服务器的内网地址），安装 Python3，pip3（用于依赖包安装）。

使用 pip3 安装依赖包：

```
pip install flag
pip install kafka-python
pip install protobuf
```

操作步骤：

1. 创建新版数据订阅通道，详情请参见 数据订阅 Kafka版。
2. 创建一个或多个消费组，详情请参见 新增消费组。
3. 下载 Python3 Demo ，然后解压该文件。
4. 运行python main.py --brokers=xxx --topic=xxx --group=xxx --user=xxx \--password=xxx --trans2sql=1。
5. 观察消费情况。

```
BEGIN
-->[partition: 0, offset: 89083, partitionSeq: 89084] [mysql-bin.000004:24876], happenedAt: 2021-03-01
20:43:31
INSERT INTO `kafka-subscribe`.`table1` VALUES (_binary'subscribe-kafka', 62)
-->[partition: 0, offset: 89083, partitionSeq: 89084] [mysql-bin.000004:24907], happenedAt: 2021-03-01
20:43:31
COMMIT
```

字段映射和存储

具体的 MySQL/TDSQL 字段值在 Protobuf 协议中用下图所示的 Data 结构来存储。

```
message Data {
DataType dataType = 1;
string charset = 2; //DataType_STRING 的编码类型，值存储在 bv 里面
string sv = 3; //DataType_INT8/16/32/64/UINT8/16/32/64/Float32/64/DataType_DECIMAL
```

```
的字符串值
bytes bv = 4; //DataType_STRING/DataType_BYTES 的值
}
```

其中 DataType 字段代表存储的字段类型，可取枚举值如下图所示。

```
enum DataType {
NIL = 0; //值为 NULL
INT8 = 1;
INT16 = 2;
INT32 = 3;
INT64 = 4;
UINT8 = 5;
UINT16 = 6;
UINT32 = 7;
UINT64 = 8;
FLOAT32 = 9;
FLOAT64 = 10;
BYTES = 11;
DECIMAL = 12;
STRING = 13;
NA = 14; //值不存在(N/A)
}
```

其中 bv 字段存储 STRING 和 BYTES 类型的二进制表示，sv 字段存储 INT8/16/32/64/UINT8/16/32/64/DECIMAL 类型的字符串表示，charset 字段存储 STRING 的编码类型。

MySQL/TDSQL 原始类型与 DataType 映射关系如下（对 UNSIGNED 修饰的 MYSQL_TYPE_INT8/16/24/32/64 分别映射为 UINT8/16/32/32/64）：

> 说明：
>
> - DATE，TIME，DATETIME 类型不支持时区。
> - TIMESTAMP 类型支持时区，该类型字段表示：存储时，系统会从当前时区转换为 UTC（Universal Time Coordinated）进行存储；查询时，系统会从 UTC 转换为当前时区进行查询。
> - 综上，如下表中 "MYSQL_TYPE_TIMESTAMP" 和 "MYSQL_TYPE_TIMESTAMP_NEW" 字段会携带时区信息，用户在消费数据时可自行转换。（例如，DTS 输出的时间格式是带时区的字符

串"2021-05-17 07:22:42 +00:00"，其中，"+00:00"表示 UTC 时间，用户在解析和转换的时候需要考虑时区信息。）

MySQL 字段类型（TDSQL 支持与 MySQL 相同的类型）	对应的 Protobuf DataType 枚举值
MYSQL_TYPE_NULL	NIL
MYSQL_TYPE_INT8	INT8
MYSQL_TYPE_INT16	INT16
MYSQL_TYPE_INT24	INT32
MYSQL_TYPE_INT32	INT32
MYSQL_TYPE_INT64	INT64
MYSQL_TYPE_BIT	INT64
MYSQL_TYPE_YEAR	INT64
MYSQL_TYPE_FLOAT	FLOAT32
MYSQL_TYPE_DOUBLE	FLOAT64
MYSQL_TYPE_VARCHAR	STRING
MYSQL_TYPE_STRING	STRING
MYSQL_TYPE_VAR_STRING	STRING
MYSQL_TYPE_TIMESTAMP	STRING
MYSQL_TYPE_DATE	STRING
MYSQL_TYPE_TIME	STRING
MYSQL_TYPE_DATETIME	STRING
MYSQL_TYPE_TIMESTAMP_NEW	STRING
MYSQL_TYPE_DATE_NEW	STRING
MYSQL_TYPE_TIME_NEW	STRNG
MYSQL_TYPE_DATETIME_NEW	STRING
MYSQL_TYPE_ENUM	STRING

MySQL 字段类型（TDSQL 支持与 MySQL 相同的类型）	对应的 Protobuf DataType 枚举值
MYSQL_TYPE_SET	STRING
MYSQL_TYPE_DECIMAL	DECIMAL
MYSQL_TYPE_DECIMAL_NEW	DECIMAL
MYSQL_TYPE_JSON	BYTES
MYSQL_TYPE_BLOB	BYTES
MYSQL_TYPE_TINY_BLOB	BYTES
MYSQL_TYPE_MEDIUM_BLOB	BYTES
MYSQL_TYPE_LONG_BLOB	BYTES
MYSQL_TYPE_GEOMETRY	BYTES

任务管理
修改订阅对象

最近更新时间：2021-01-26 18:04:14

数据订阅 Kafka 版支持在数据消费的过程中，动态增加或减少订阅对象。本文为您介绍如何通过控制台修改数据订阅 Kafka 版的订阅对象。

前提条件

已创建 数据订阅 Kafka 版。

操作步骤

1. 登录 DTS 控制台，在左侧导航选择【数据订阅】，进入数据订阅页面。
2. 在数据订阅列表，选择所需的数据订阅，选择“操作”列的【更多】>【修改订阅对象】，进入修改订阅对象页面。

3. 在修改订阅对象页面，重新选择订阅类型，并编辑订阅对象，单击【保存】。

4. 返回订阅列表，订阅实例进入启动中状态，进行任务预检查并初始化。启动成功后，订阅实例进入运行中状态，即可使用 Kafka 客户端消费订阅数据。

重置数据订阅

最近更新时间：2021-01-27 09:22:09

数据订阅 Kafka 版支持通过重新配置任务，来删除当前订阅实例的信息和数据，并重新配置新的数据订阅任务。 本文为您介绍如何通过控制台重置数据订阅。

前提条件

- 已创建 数据订阅 Kafka 版。
- 数据订阅状态为“运行中”或“异常中”。

操作步骤

1. 登录 DTS 控制台，在左侧导航选择【 数据订阅 】，进入数据订阅页面。
2. 在数据订阅列表，选择所需的数据订阅，选择“操作”列的【 更多 】>【 重置订阅 】。

3. 在弹出的对话框，确认无误后，单击【 确定 】。

> **注意：**
>
> - 订阅重置后，会解除订阅实例和源实例的绑定关系，同时恢复至未启动状态，您可以再次进行初始化配置。
> - 重置订阅操作将停止订阅源库增量数据，订阅中已存储的增量数据将被删除，请谨慎操作。

4. 在数据订阅列表，单击“操作”列的【 配置订阅 】，重新配置任务。