Klustron CDC功能初探

**注意：**

  如无特别说明，文中的版本号可以使用任何已发布版本的版本号代替。所有已发布版本详见：<http://doc.klustron.com/zh/Release_notes.html>。

**本文内容：**

Klustron数据库存储节点兼容MySQL binlog dump协议，意味着数据库引擎具有较广泛的兼容性。为了便利现有MySQL的用户将业务数据迁移至Klustron数据库，或基于其他业务需求，需要将Klustron数据库中的业务数据抽取出来同步到其他数据库，甚至是抽取生成其他格式的文件，Klustron从1.2.1开始提供适用于数据同步需求的CDC模块，Klustron CDC缺省支持两种模式，一种是从Klustron集群导出数据(SQL或文件)，一种是支持从开源MySQL导出数据到Klustron集群。

在本文中，我们将实践完成上述的几种功能场景，并且因为Klustron同时支持从XPanel完成CDC任务相关的配置操作，也支持以Rest API调用的方式实现同样的功能，所以文中也将同步阐述。

1. **环境准备**

本文所指的测试场景中，涉及使用CDC进行MySQL与Klustron之间的双向同步测试，所以，需要准备一个MySQL的实例环境，另外再准备一套Klustron的运行环境，理论上，需要4台服务器，1台用于MySQL安装，3台用于Klustron环境，受限于资源有限，故使用3台机器搭建整个测试环境，因为使用了不同的Linux帐户，不同的服务端口，所以不存在环境冲突。

* 1. MySQL服务器安装配置

《略》，具体请参考MySQL社区版安装文档。

MySQL 版本：MySQL 8.0.34

IP：192.168.0.155

端口：3306

Linux：Ubuntu 20.04.2

说明：要完成MySQL向Klustron进行同步，MySQL需要启用binlog(8.0已默认开启)，并且同时启用GTID，按如下指令完成设置：

root@kunlun3:~# mysql -h 192.168.0.155 -P 3306 -u root -p

mysql> SET @@GLOBAL.ENFORCE\_GTID\_CONSISTENCY = WARN;

mysql> SET @@GLOBAL.ENFORCE\_GTID\_CONSISTENCY = ON;

mysql> SET @@GLOBAL.GTID\_MODE = OFF\_PERMISSIVE;

mysql> SET @@GLOBAL.GTID\_MODE = ON\_PERMISSIVE;

mysql> SET @@GLOBAL.GTID\_MODE = ON;

mysql> SET PERSIST GTID\_MODE = on ;

mysql> SET PERSIST ENFORCE\_GTID\_CONSISTENCY = on ;

mysql> SET PERSIST binlog\_row\_metadata='FULL';

mysql> SET PERSIST binlog\_row\_image='FULL';

退出mysql，重启MySQL服务，使参数生效：

systemctl restart mysqld

* 1. Klustron 安装配置

《略》

Klustron 环境说明：

XPanel: http://192.168.0.152:40181/KunlunXPanel/#/cluster

计算节点：192.168.0.155 ,端口: 47001

存储节点（shard1）：192.168.0.153, 端口：57007 (主)

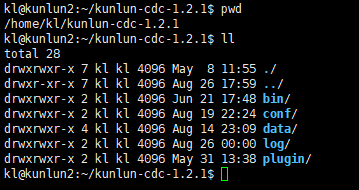
存储节点（shard2）：192.168.0.152, 端口：57005 (主)

Klustron 安装在kl用户下

* 1. CDC安装配置

从[http://zettatech.tpddns.cn:14000/dailybuilds/enterprise/kunlun-cdc-1.2.1.tgz 下载文件kunlun-cdc-1.2.1.tgz](http://zettatech.tpddns.cn:14000/dailybuilds/enterprise/kunlun-cdc-1.2.1.tgz%20下载文件kunlun-cdc-1.2.1.tgz) 释放到192.168.0.152 / 153/ 155 3台机的kl用户的home目录下，直接用tar -zxvf kunlun-cdc-1.2.1.tgz 解开该文件，在此目录下，会生成名为：

/home/kl/kunlun-cdc-1.2.1，及相关的子目录，形式如下：



用kl用户进入192.168.0.152 / 192.168.0.153/ 192.168.0.155 的conf目录，编辑参数文件：kunlun\_cdc.cnf

cd /home/kl/kunlun-cdc-1.2.1/conf

vi kunlun\_cdc.cnf

修改如下参数：

local\_ip = 192.168.0.153 #3台机各自的IP

http\_port = 18012 #3台机取一致的端口，实际可以不一致，按需定义

ha\_group\_member = 192.168.0.152:18081,192.168.0.153:18081,192.168.0.155:18081 #3台机组成一个CDC的高可用集群，端口自定义，本例取18081

server\_id = 2 #3台机各自定义一个自已的ID号，不唯一即可

保存退出。

进入3台机的CDC bin目录，该目录下有用于启停CDC服务的命令：

start\_kunlun\_cdc.sh stop\_kunlun\_cdc.sh

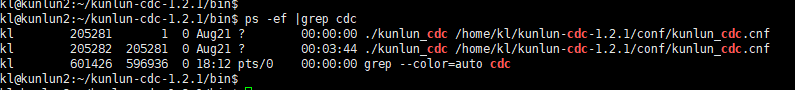
在3台机执行如下指令，启动CDC服务

cd /home/kl/kunlun-cdc-1.2.1/bin

./start\_kunlun\_cdc.sh

ps -ef |grep cdc

输出如下信息，说明启动CDC服务成功



登陆到XPanel，http:// 192.168.0.152:40181/KunlunXPanel/#/cdc/list，将打开如下页面：



点击“新增”配置“CDC服务”，输入如下相应的参数，



分组号可以自定义，本例中取分组号为1，因为在本例中是配置了3台CDC服务器组成高可用集群，所以，需要点击上图中右侧的“+”号继续完成剩余两个CDC服务的配置信息添加工作，并完成后点击“确认”进行保存。



保存后，界面效果如下所示：



注意一下，其中192.168.0.152在“主节点”字段显示为1，意味着当前承担CDC服务的节点是它，其他两个节点作为备节点，在主节点失效的时候，会选择其中一个升级为主节点，继续服务CDC相关的任务。

另外，Klustron也提供了Rest API来获得CDC服务的配置信息，调用示例如下：（在3个Klustron节点机器的其中一个节点上执行即可）

kl@kunlun1:~$ curl -d '

{

"version":"1.0",

"job\_id":"",

"job\_type":"list\_cdc\_conf",

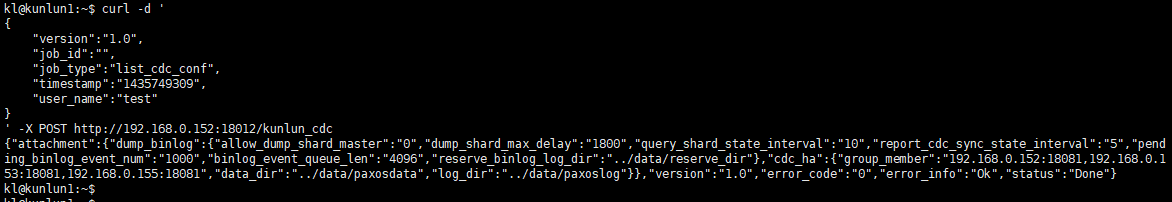
"timestamp":"1435749309",

"user\_name":"test"

}

' -X POST http://192.168.0.152:18012/kunlun\_cdc

输出信息如下：



以及，想通过Rest API了解当前CDC集群中哪个节点是当前主节点，做如下调用：

kl@kunlun1:~$ curl -d '

{

"version":"1.0",

"job\_id":"",

"job\_type":"get\_leader",

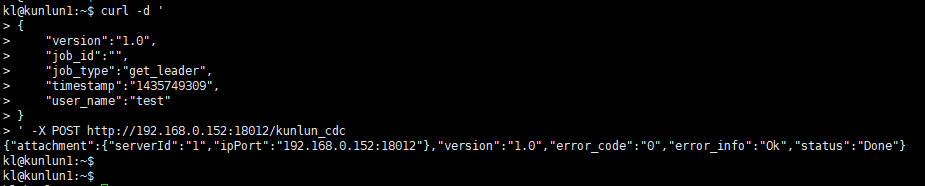
"timestamp":"1435749309",

"user\_name":"test"

}

' -X POST <http://192.168.0.152:18012/kunlun_cdc>

输出如下：



1. **CDC测试**
   1. 从MySQL向Klustron同步数据

先在MySQL中建立源库相应的数据库，用户，表，按如下指令创建相应对象：

root@kunlun3:~# mysql -h 192.168.0.155 -P 3306 -u root -p

mysql> CREATE USER 'repl'@'%' IDENTIFIED WITH mysql\_native\_password BY 'repl';

mysql> GRANT REPLICATION SLAVE ON \*.\* TO 'repl'@'%';

mysql> create database test ;

mysql> create user 'test'@'%' identified by 'test';

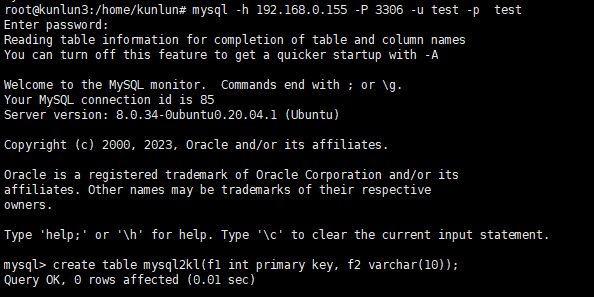
mysql> grant all on test.\* to 'test'@'%';

mysql> flush privileges;

mysql> exit;

root@kunlun3:~# mysql -h 192.168.0.155 -P 3306 -u test -p test

mysql> create table mysql2kl (f1 int primary key, f2 varchar(10));



连接Klustron计算节点，建立目标库相应的用户，schema，表，按如下指令创建相应对象：

kl@kunlun3:~$ psql -h 192.168.0.155 -p 47001 -U adc postgres

create user test with password 'test';

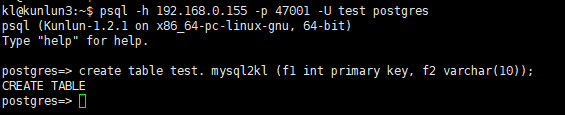
grant create on database postgres to test ;

exit

kl@kunlun3:~$ psql -h 192.168.0.155 -p 47001 -U test postgres

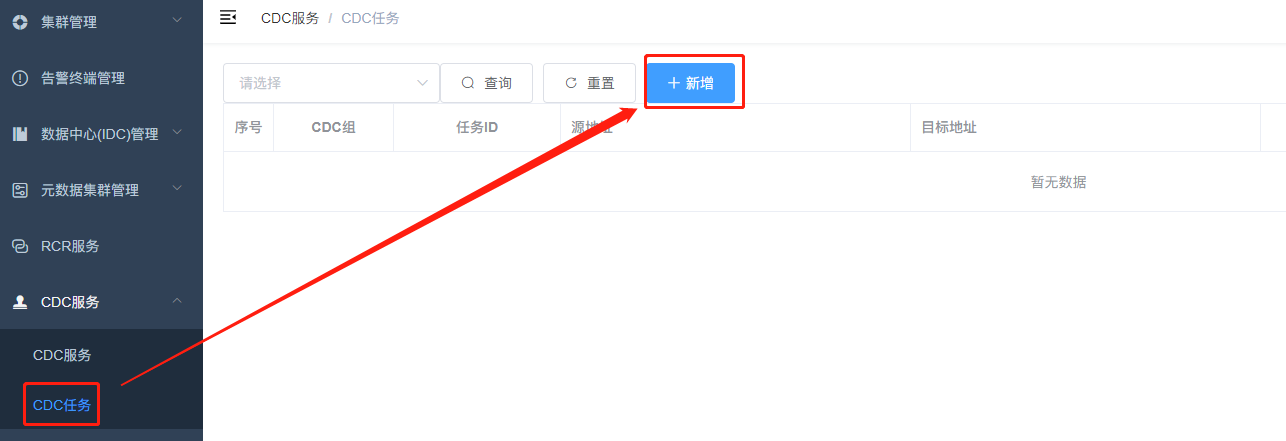
create schema test ;

create table test. mysql2kl (f1 int primary key, f2 varchar(10));

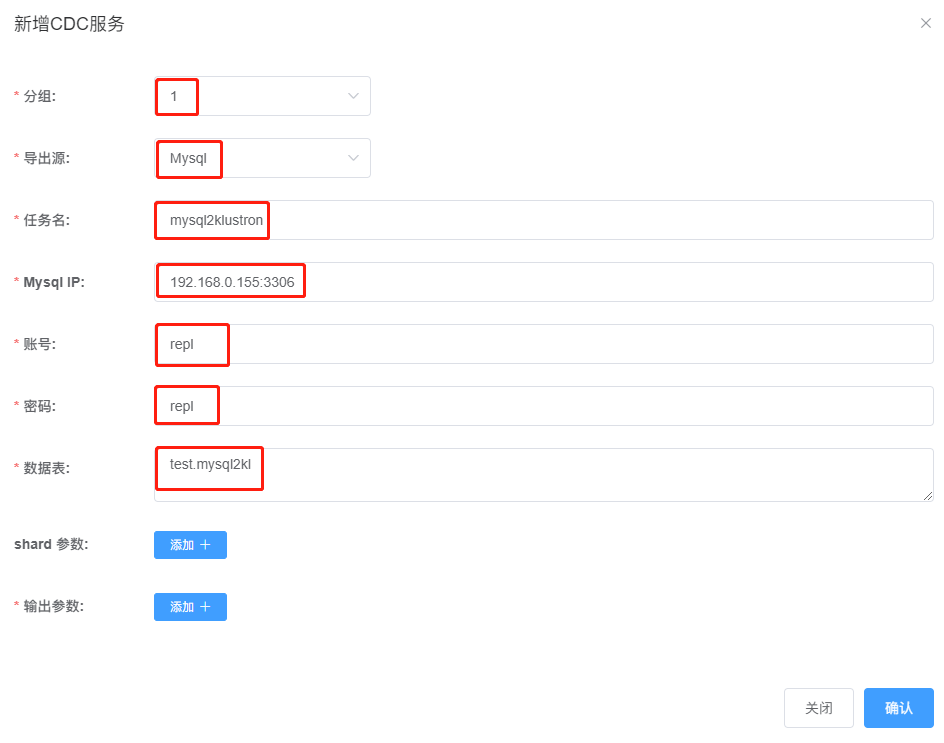


注意：目标端的数据表必须有主键，才能确保不会重复执行SQL语句，否则新的CDC主节点会把上一个主节点在退出之前最后若干秒之内的所有操作会再次执行一遍，那样就会导致相关数据行被重复地插入、删除或者更新，导致目的端数据与源数据库中的数据不再一致，那样还可能会导致后续数据同步失败和无法继续。

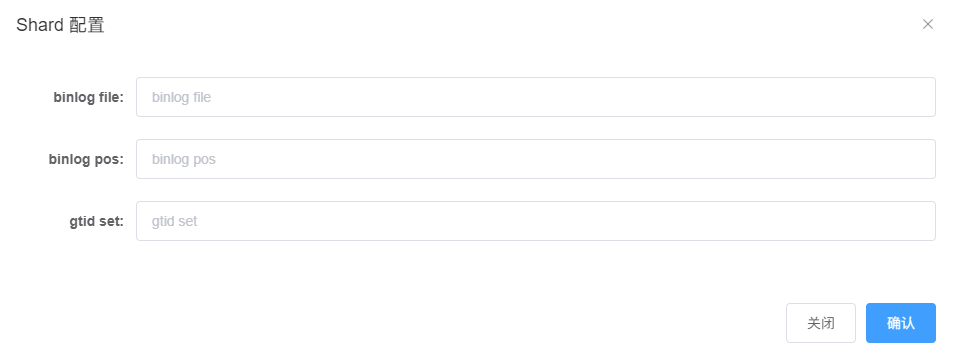
打开XPanel：http:// 192.168.0.152:40181/KunlunXPanel/#/cdc/worker ，为MySQL向Klustron进行数据同步添加CDC任务：



点击“+新增”，输入如下参数：



然后，点击 “shard参数:“右侧的“添加+”，弹出如下对话框：

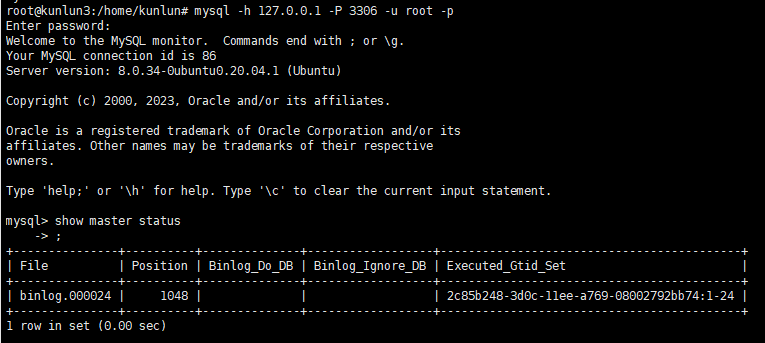


这里，需要连接到MySQL，获取该服务实例当前的这几个参数信息，按如下指令操作：

root@kunlun3:~# mysql -h 127.0.0.1 -P 3306 -u root -p

mysql> show master status ;

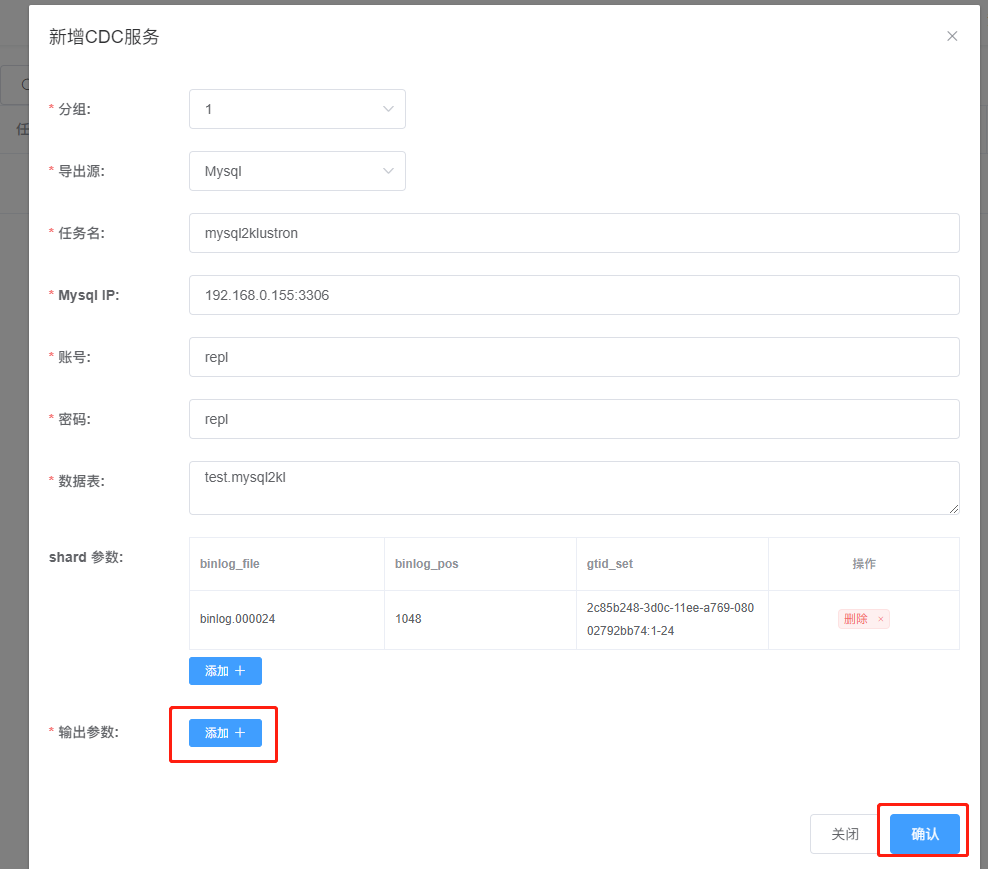
输出信息如下所示：



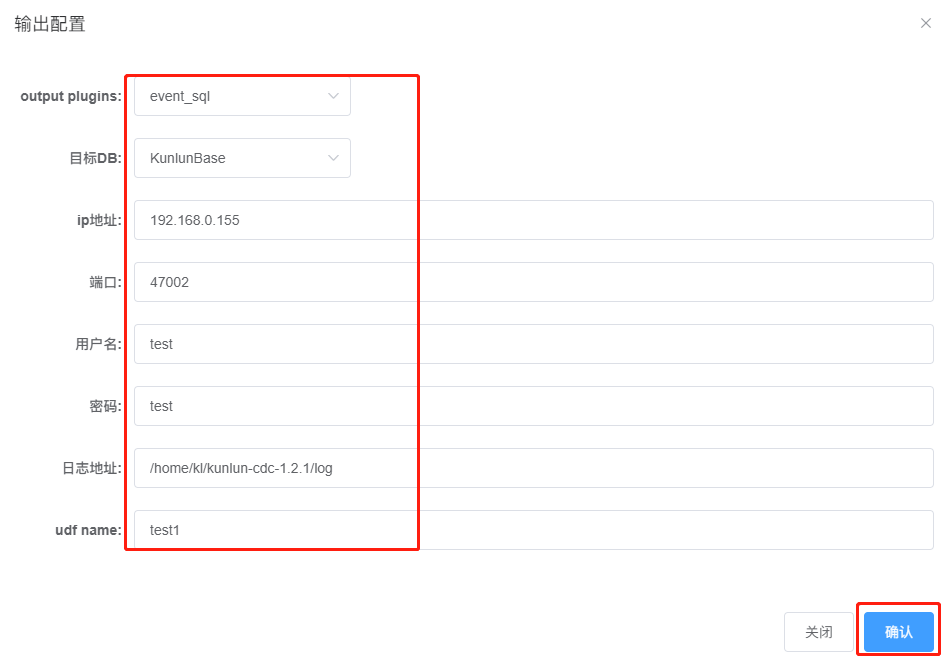
将 file, position, executed\_gtid\_set 填写至前述shard 配置信息对话框中，如下所示：



点击“确认”保存配置信息，继续进行“输出参数：”配置：



点击“添加+”后，弹出如下对话框：



如下对几个参数做出填写说明：

Output plugins: 我们这里是要完成MySQL到Klustron的数据同步，所以，插件类型选“event\_sql”；

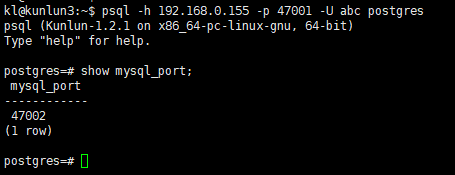
目标DB：选KunlunBase；

Ip地址：指的是目标Klustron 集群的IP，这里是192.168.0.155，即计算节点的IP;

端口：注意一下，这里不能是PG协议的端口47001，必须是MySQL协议的服务端口，通常它是PG协议的端口号加1，即47002，也可以通过在计算节点执行指令来查看MySQL协议的服务端口，示例如下：

psql -h 192.168.0.155 -p 47001 -U abc postgres

show mysql\_port;

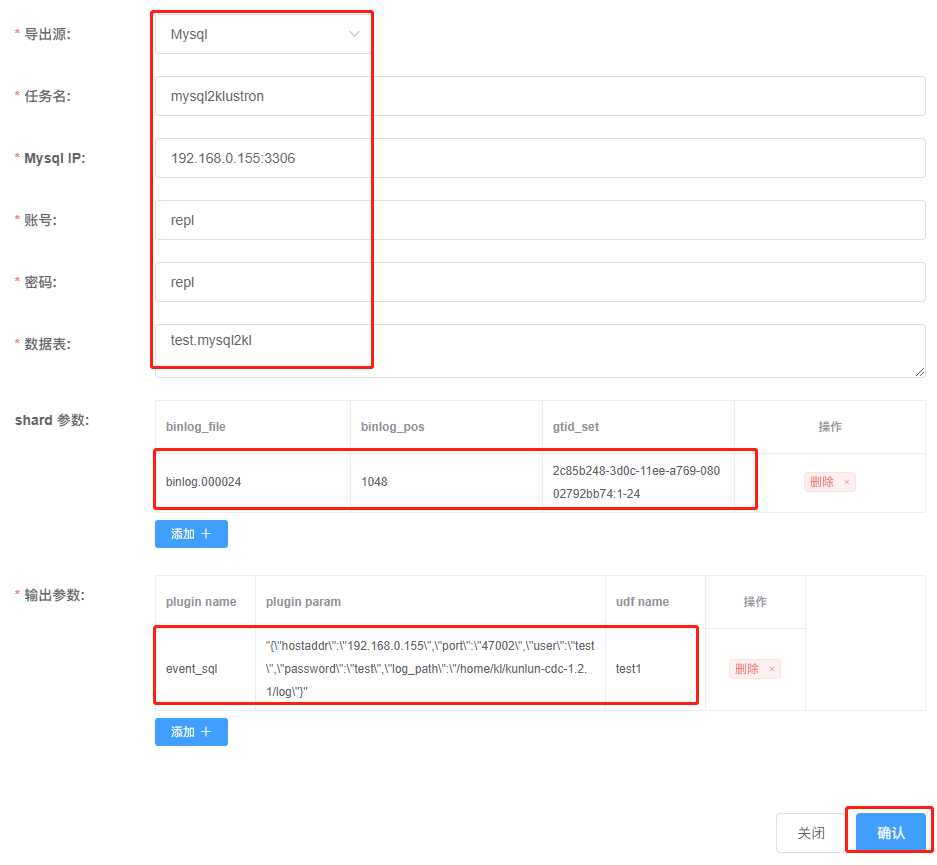


用户名：/密码：就是用于被同步的表所属的用户/密码；

日志地址：在CDC服务执行该CDC任务的时候，会产生日志，该参数用于指定日志存放的地址；

udf name: 在此例无特别意义，会将该处输入的字符串，作为一部分加入到日志文件名中，稍后可在CDC任务运行过程中，在“日志地址：”所指的路径下找到日志文件，来了解日志文件名的生成规则。

上述参数录入完成后，点“确认”保存参数配置，配置任务窗口当前显示如下：



再次点击“确认”后，XPanel将会向CDC服务提交一个MySQL到Klustron的同步服务任务，界面示例如下：



以上MySQL到Klustron的同步任务的创建，也可以通过调用Rest API的方式来完成，任意一台能连通192.168.0.152的机器上，执行如下调用指令即可：

kl@kunlun3:~$curl -d '

{

"version":"1.0",

"job\_id":"",

"job\_type":"add\_dump\_table",

"timestamp":"1435749309",

"user\_name":"test",

"paras":{

"meta\_db":"192.168.0.155:3306", --- dump MySQL的ip:port

"meta\_user":"repl", --- 连接mysql的账户(有replication权限的用户)

"meta\_passwd":"repl", --- 连接mysql的密码

"cluster\_name":" mysql2klustron ",

"dump\_tables":" mysql2kl ",

"is\_kunlun":"0",

"shard\_params":[{

"binlog\_file":"binlog.000024",

"binlog\_pos":"1048",

"gtid\_set":" 2c85b248-3d0c-11ee-a769-08002792bb74:1-24 "

}],

"output\_plugins":[

{

"plugin\_name":"event\_file", --- 将捕获的binlog写入指定的文件

"plugin\_param":"/home/kl/kunlun-cdc-1.2.1/log/event.log", --指定在哪个路径及文件中存储捕获的binlog

"udf\_name":"test1" ---本示例中无特殊意义

},

{

"plugin\_name":"event\_sql", "plugin\_param":"{\"hostaddr\":\"192.168.0.155\",\"port\":\"47002\",\"user\":\"test\",\"password\":\"test\",\"log\_path\":\"/home/kl/kunlun-cdc-1.2.1/log\"}",

"udf\_name":"test2"

}

]

}

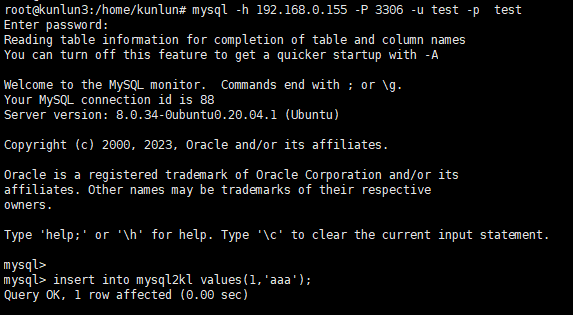
} ' -X POST http://192.168.0.152:18012/kunlun\_cdc

注意：本调用示例对比前述XPanel配置CDC任务的方式，多了一个event\_file插件的使用，意即对于CDC任务，对于捕获的日志，输出时可以有多个目标，且可以同时是多个类型的输出目标。

如果同步任务的状态正常，则我们可以接下来进行数据同步测试，连接到MySQL服务，对mysql2kl表插入记录：

root@kunlun3:~#mysql -h 192.168.0.155 -P 3306 -u test -p test

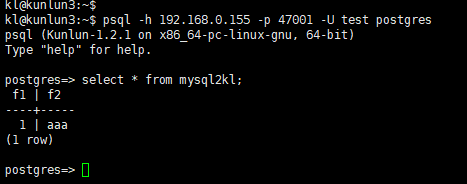
insert into mysql2kl values(1,'aaa');



再连接到Klustron，检查目标表的同步情况：

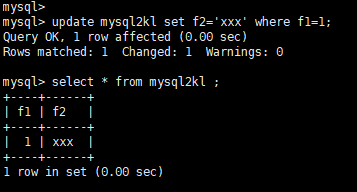
kl@kunlun3:~$ psql -h 192.168.0.155 -p 47001 -U test postgres

postgres=> select \* from mysql2kl;



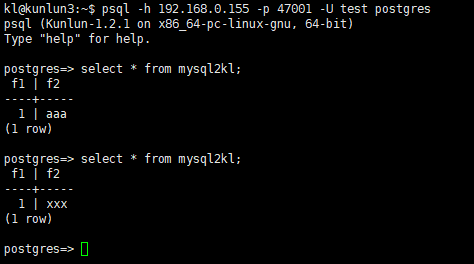
可以看到，目标库的同名表中已有之前在MySQL中插入的记录，证明同步任务执行正常。再次到MySQL实例上更新刚才的记录字段值，如下操作：

update mysql2kl set f2='xxx' where f1=1;



切换到Klustron集群中查看mysql2kl表的情况：

Select \* from mysql2kl;

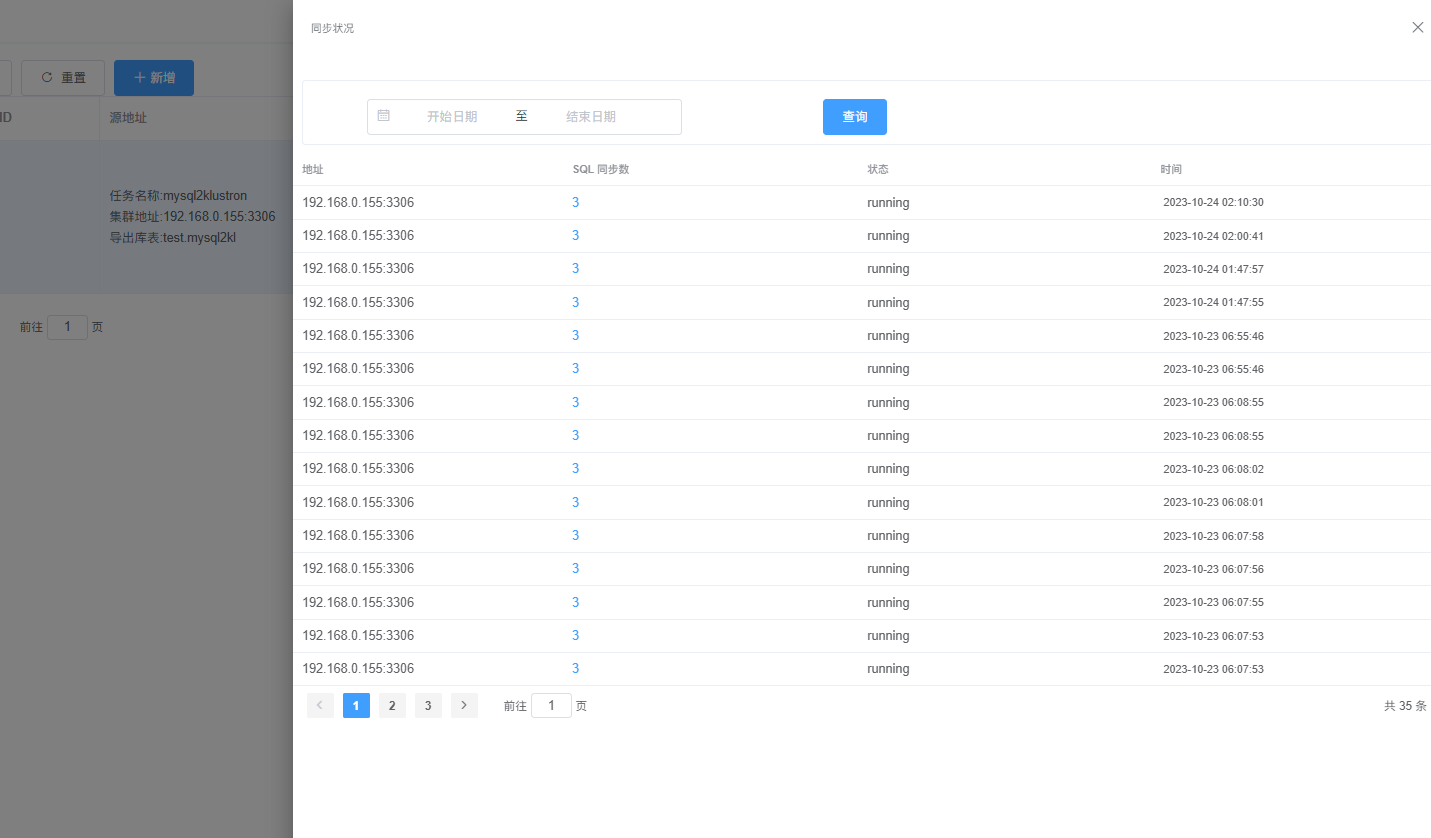


字段记录同步更新成功。

CDC任务在运行过程中，也可以在XPanel界面上查看每个任务的详细执行日志信息，回到XPanel 主界面，点击左侧导航菜单栏中的“CDC任务”，如下所示：



右侧的CDC任务列表中，每一个同步任务的右侧，都有一个“详情”按钮，点击该按钮后，弹出如下窗口，显示同步任务的详细信息：



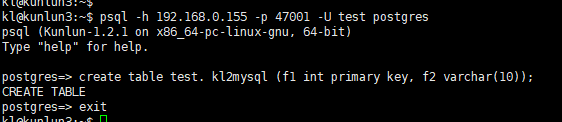
根据需要，可自定义输入不同的“开始日期”和“结束日期”来查看特定时段的任务执行日志。

* 1. 从Klustron向MySQL同步数据

连接Klustron计算节点，建立源库相应的表，按如下指令创建相应对象：

kl@kunlun3:~$ psql -h 192.168.0.155 -p 47001 -U test postgres

create table test. kl2mysql (f1 int primary key, f2 varchar(10));



然后在目标端 MySQL中建立相应的表，按如下指令创建相应对象：

root@kunlun3:/home/kl# mysql -h 192.168.0.155 -P 3306 -u root -p

mysql> create database postgres\_$$\_test ;

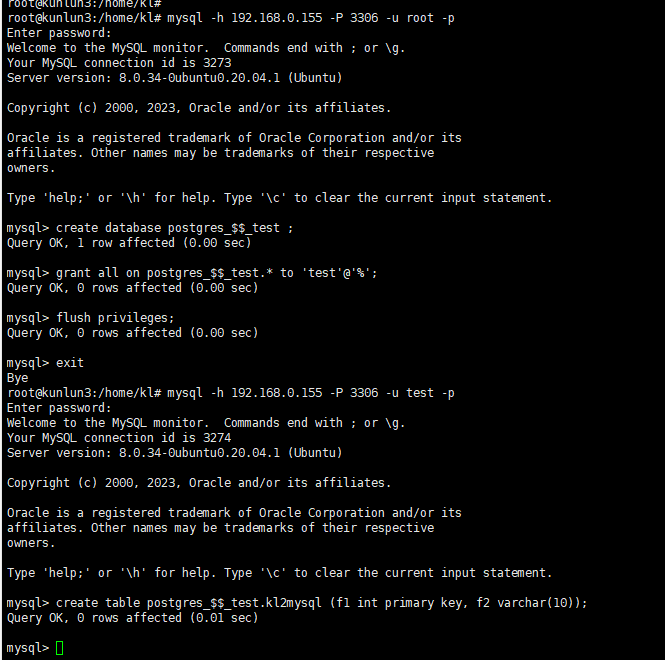
mysql> grant all on postgres\_$$\_test.\* to 'test'@'%';

mysql> flush privileges;

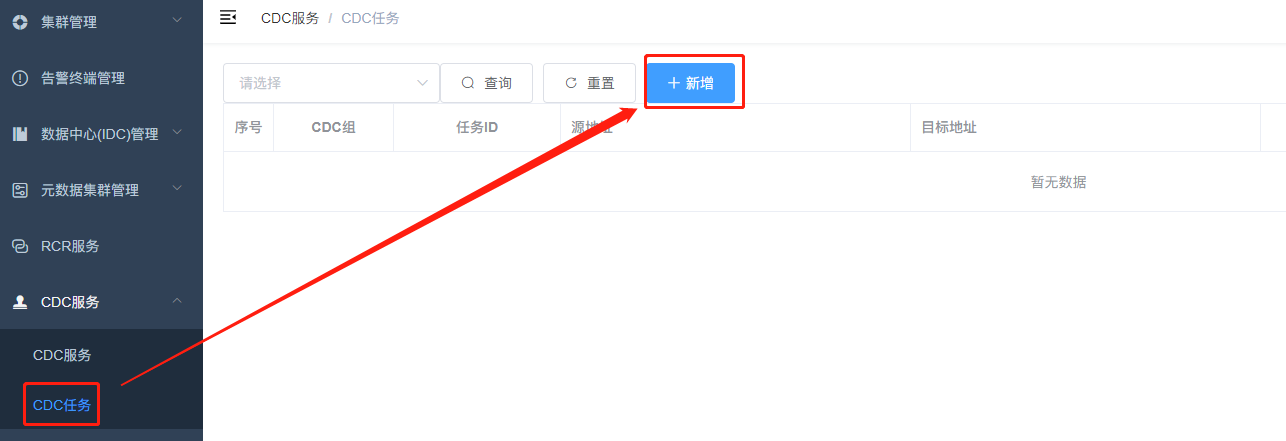
mysql> exit

root@kunlun3:/home/kl# mysql -h 192.168.0.155 -P 3306 -u test -p

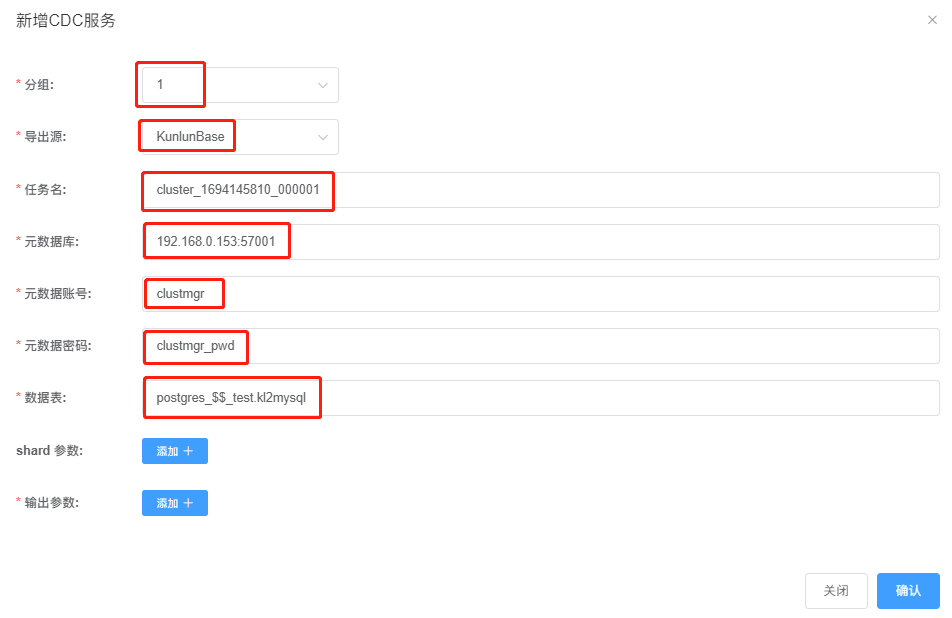
mysql> create table postgres\_$$\_test.kl2mysql (f1 int primary key, f2 varchar(10));



打开XPanel：http:// 192.168.0.152:40181/KunlunXPanel/#/cdc/worker ，为MySQL向Klustron进行数据同步添加CDC任务：



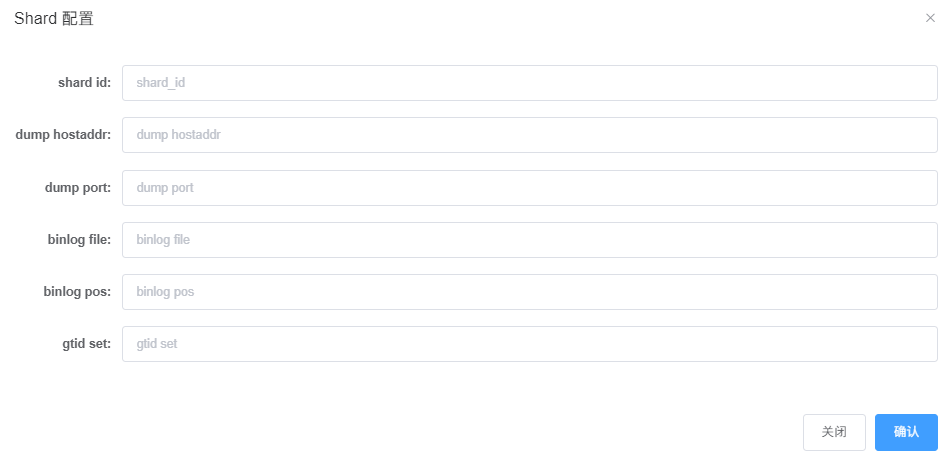
点击“+新增”，输入如下参数：



说明：

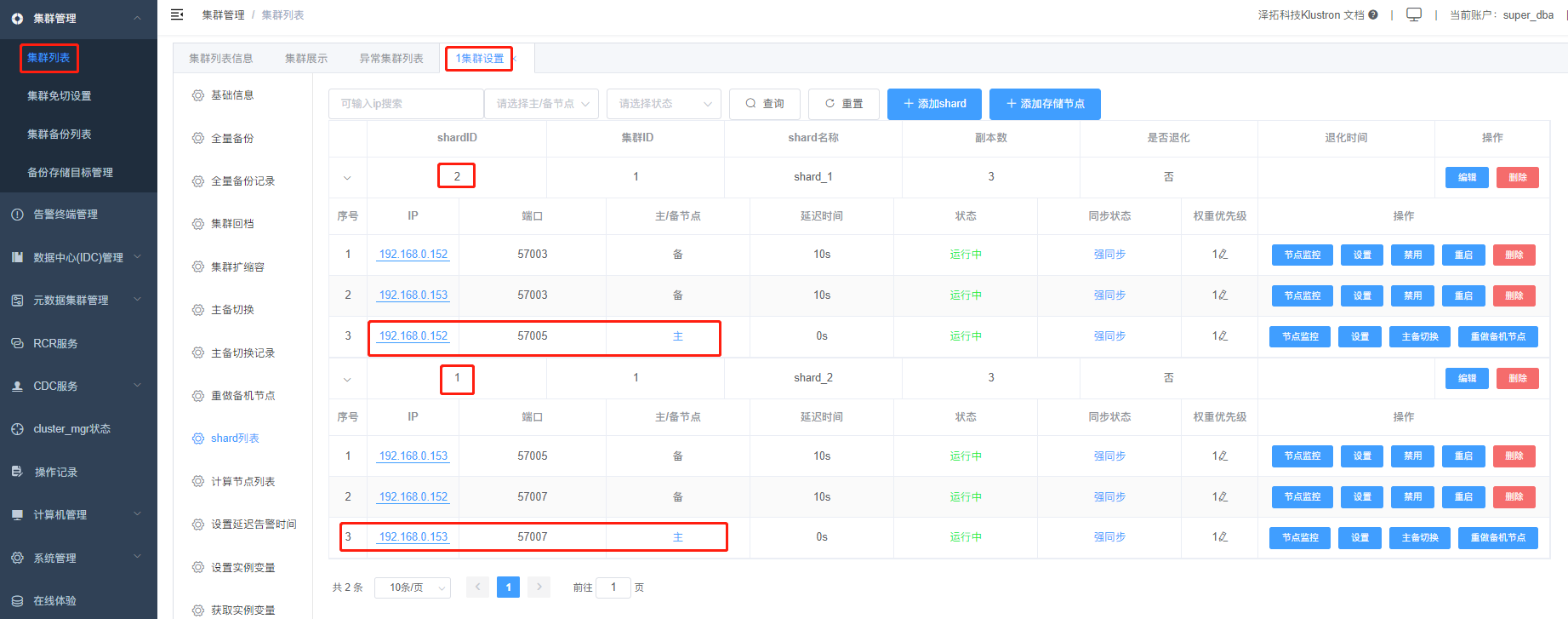
1. 任务名：需要查看Klustron集群cluster1的名称，非随意填入；
2. 元数据库：Klustron元数据库是一个3节点的MGR，需在XPanel中查看Klustron集群环境中MGR的主节点当前是哪一个，在此处填写主节点信息；
3. 数据表：数据表的格式，需要遵循：<database名>\_$$\_<schema名>.<表名>的格式，有多个表要同步的话，中间用“，”隔开。

然后，点击 “shard参数:“右侧的“添加+”，弹出如下对话框：



这里，需要连接到存储节点的MySQL分片，获取两个分片各自的binlog相关的信息，按如下指令操作：

分片信息：

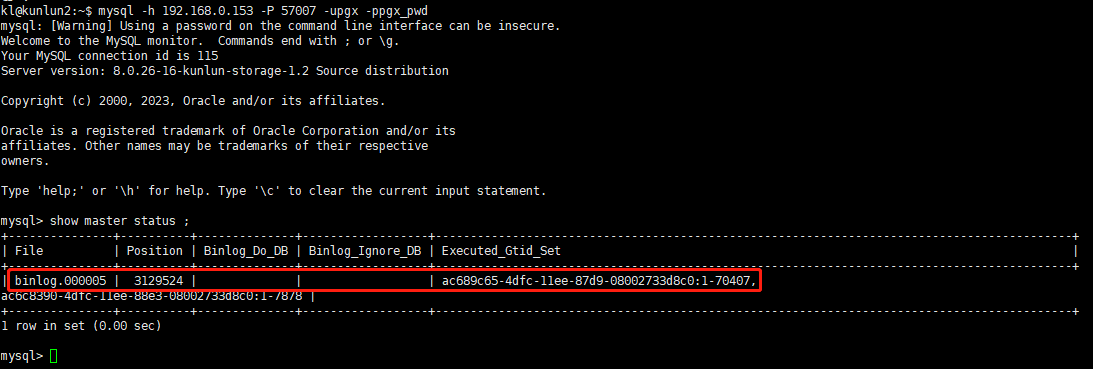


分片1：

kl@kunlun2:~$ mysql -h 192.168.0.153 -P 57007 -upgx -ppgx\_pwd

mysql> show master status ;

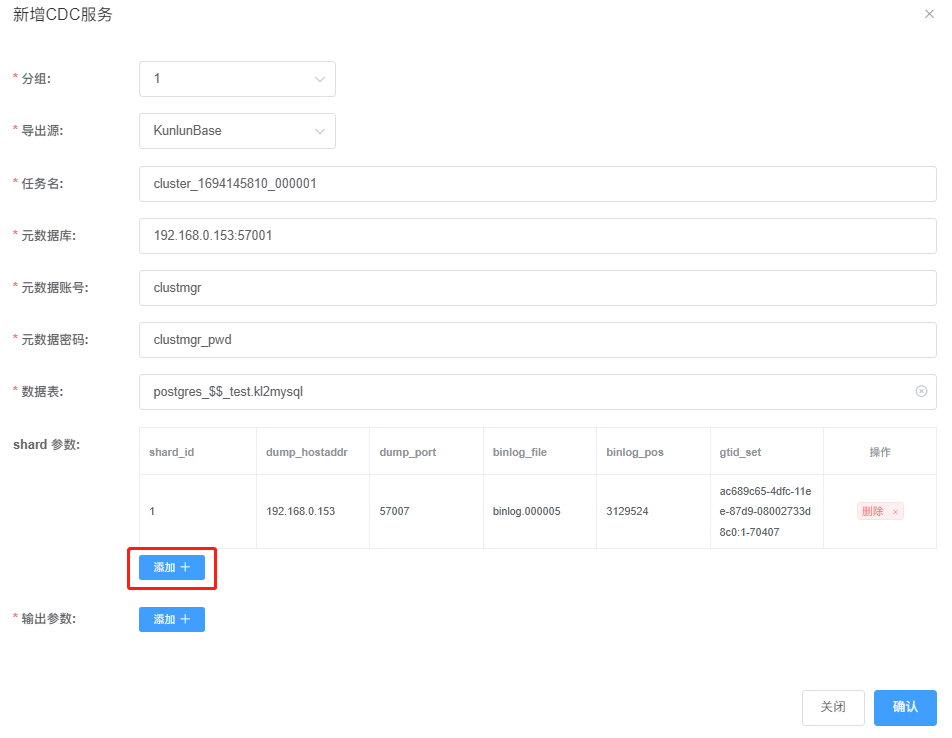
输出信息如下所示：



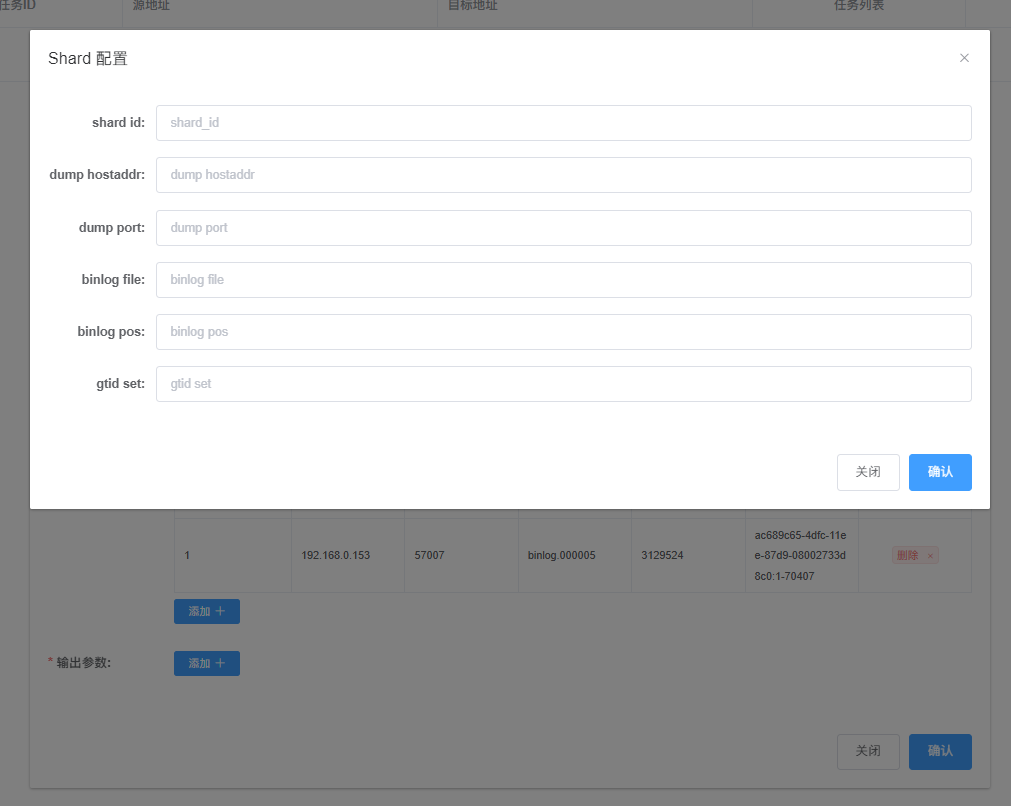
将 上图中file, position, executed\_gtid\_set 等信息填写至前述shard 配置信息对话框中，如下所示：



点击“确认”完成第一个分片信息配置，显示如下：



点击“添加”进行分片2信息的配置，



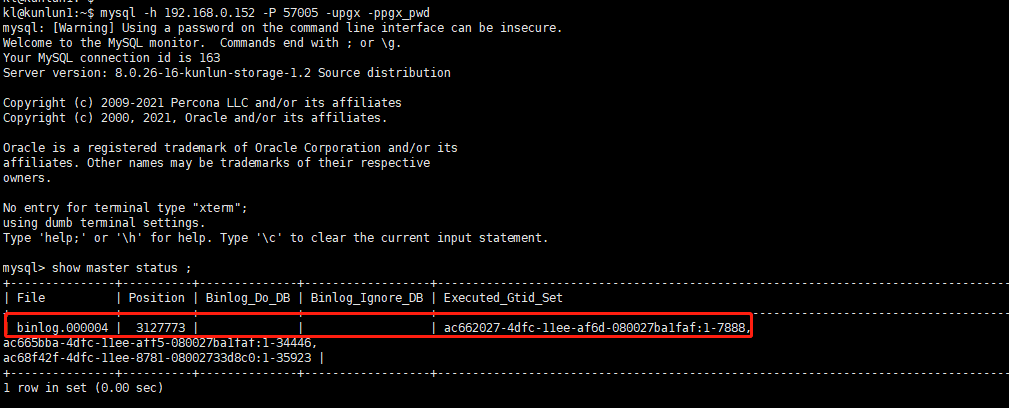
操作如下：

分片2：

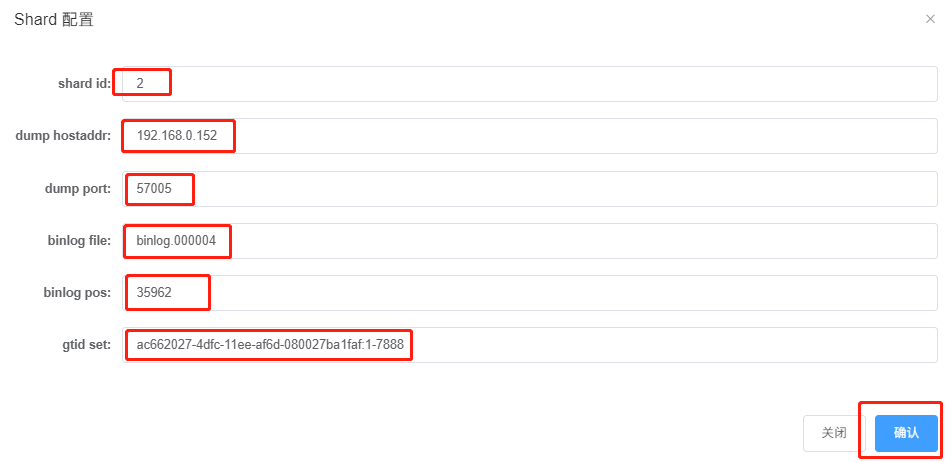
kl@kunlun1:~$ mysql -h 192.168.0.152 -P 57005 -upgx -ppgx\_pwd

mysql> show master status ;

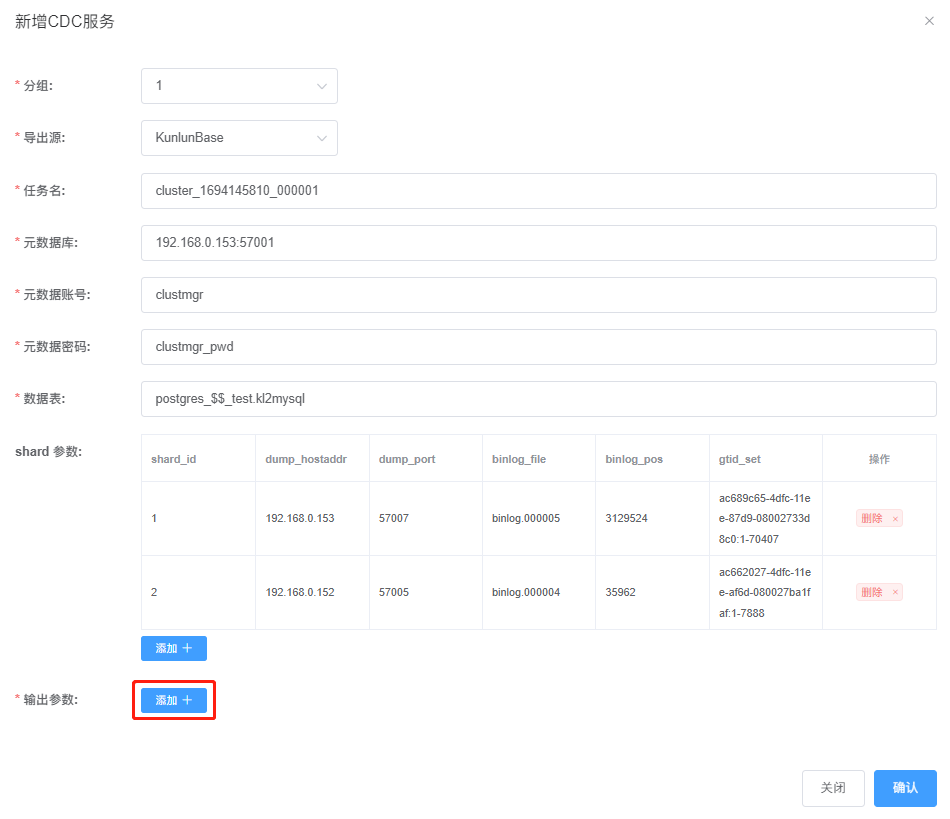
输出信息如下所示：



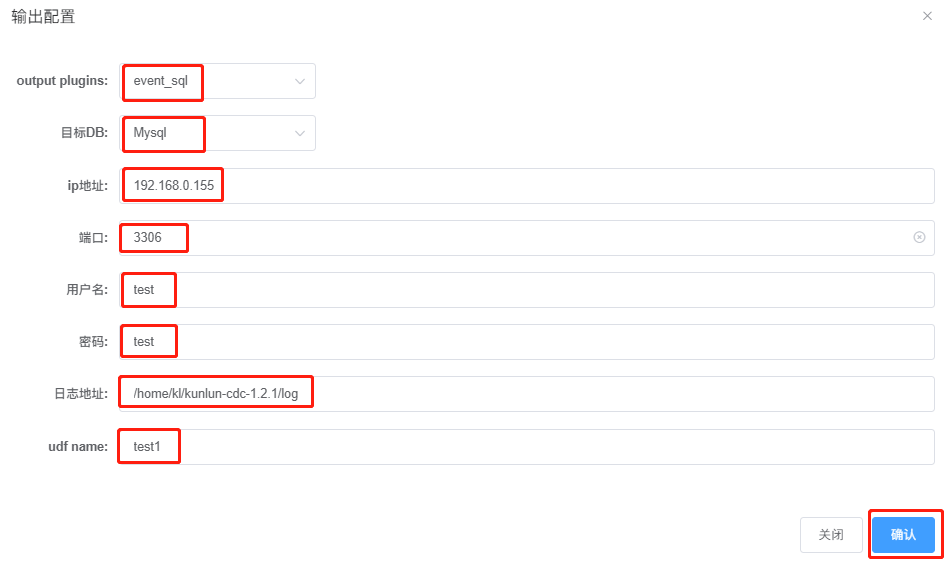
将 file, position, executed\_gtid\_set 等信息填写至前述shard 配置信息对话框中，如下所示：



点击“确认”保存，显示如下：



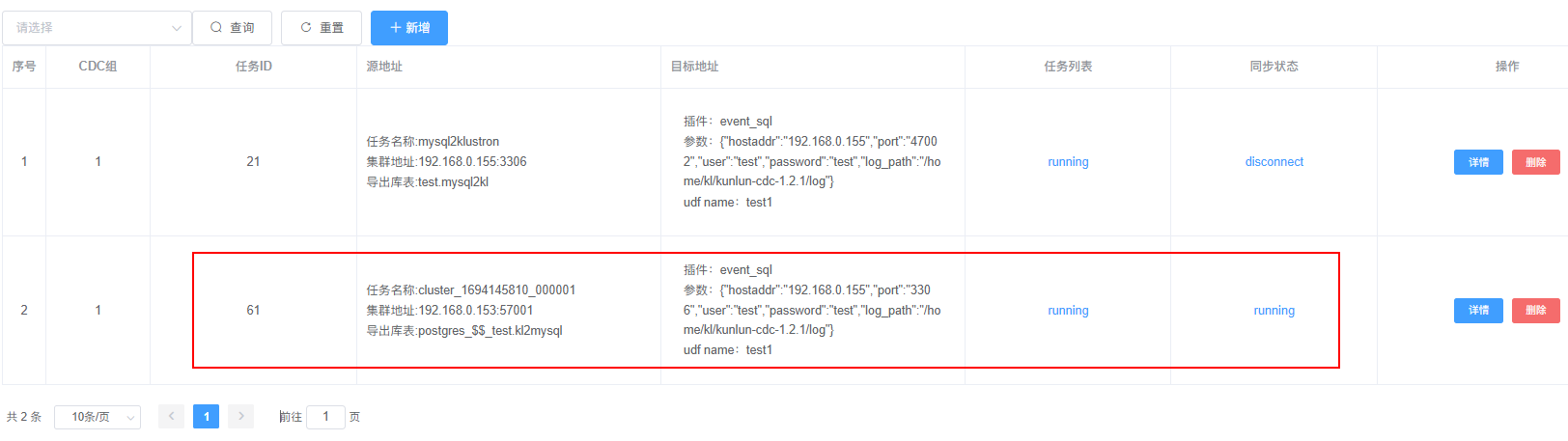
点击“添加”，完成输出参数配置：



点击“确认”，完成CDC 参数配置，显示如下：



确认无误后，点击“确认”，完成CDC任务上报，显示如下：



以上Klustron到MySQL的同步任务的创建，也可以通过调用Rest API的方式来完成，任意一台能连通192.168.0.152的机器上，执行如下调用指令即可：

curl -d '

{

"version":"1.0",

"job\_id":"",

"job\_type":"add\_dump\_table",

"timestamp":"1435749309",

"user\_name":"test",

"paras":{

"meta\_db":"192.168.0.153:57001",

"meta\_user":"clustmgr",

"meta\_passwd":"clustmgr\_pwd",

"cluster\_name":"cluster\_1694145810\_000001",

"dump\_tables":"postgres\_$$\_test.kl2mysql",

"shard\_params":[

{

"shard\_id":"1",

"dump\_hostaddr":"192.168.0.153",

"dump\_port":"57007",

"binlog\_file":"binlog.000005",

"binlog\_pos":"3129524",

"gtid\_set":"ac689c65-4dfc-11ee-87d9-08002733d8c0:1-70407"

},{

"shard\_id":"2",

"dump\_hostaddr":"192.168.0.152",

"dump\_port":"57005",

"binlog\_file":"binlog.000004",

"binlog\_pos":"3127773",

"gtid\_set":"ac662027-4dfc-11ee-af6d-080027ba1faf:1-7888"

}

],

"output\_plugins":[

{

"plugin\_name":"event\_file",

"plugin\_param":"/home/kl/kunlun-cdc-1.2.1/log/event.log",

"udf\_name":"test1"

},

{

"plugin\_name":"event\_sql",

"plugin\_param":"{\"hostaddr\":\"192.168.0.155\",\"port\":\"3306\",\"user\":\"test\",\"password\":\"test\",\"log\_path\":\"/home/kl/kunlun-cdc-1.2.1/log\", \"is\_kunlun\": \"0\"}",

"udf\_name":"test1"

}

]

}

}' -X POST http://192.168.0.153:18012/kunlun\_cdc

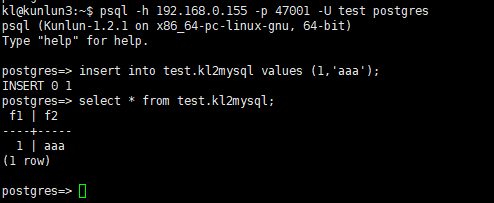
注意：本调用示例对比前述XPanel配置CDC任务的方式，多了一个event\_file插件的使用，意即对于CDC任务，对于捕获的日志，输出时可以有多个目标，且可以同时是多个类型的输出目标。

如果同步任务的状态正常，则我们可以接下来进行数据同步测试，连接到Klustron集群的计算实例，对kl2mysql表插入记录：

kl@kunlun3:~$ psql -h 192.168.0.155 -p 47001 -U test postgres

postgres=> insert into test.kl2mysql values (1,'aaa');

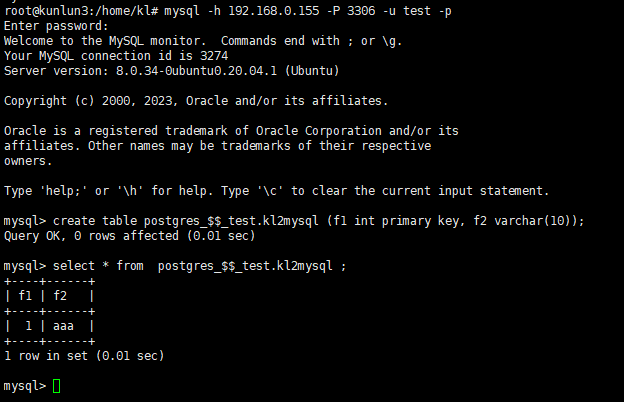
postgres=> select \* from test.kl2mysql;insert into mysql2kl values(1,'aaa');



再连接到MySQL，检查目标表的同步情况：

root@kunlun3:/home/kl# mysql -h 192.168.0.155 -P 3306 -u test -p

mysql> select \* from postgres\_$$\_test.kl2mysql ;



可以看到，目标库的同名表中已有之前在Klustron中插入的记录，证明同步任务执行正常。

* 1. 其他CDC相关的API说明及使用示例

在完成一些CDC任务创建后，想通过Rest API来获得所有的任务信息，通过如下调用完成；

kl@kunlun3:~$ curl -d '

{

"version":"1.0",

"job\_id":"",

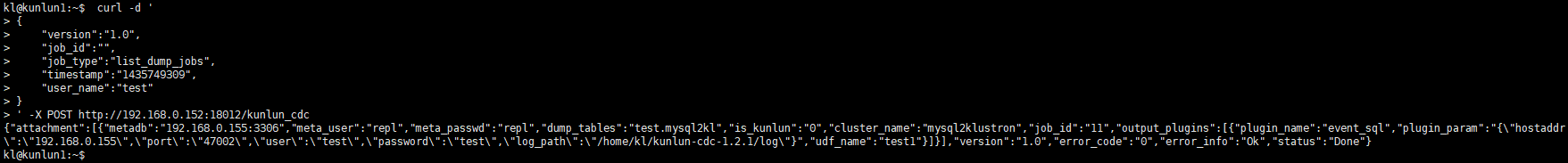
"job\_type":"list\_dump\_jobs",

"timestamp":"1435749309",

"user\_name":"test"

}

' -X POST http://192.168.0.152:18012/kunlun\_cdc



如果想要通过Rest API了解某个CDC任务的状态，通过如下调用完成：

kl@kunlun3:~$ curl -d '

{

"version":"1.0",

"job\_id":"11", --- job\_id 由 list\_dump\_jobs 调用获得或在XPanel上查看

"job\_type":"get\_job\_state",

"timestamp":"1435749309",

"user\_name":"repl",

"paras":{

"meta\_db":"192.168.0.155:3306",

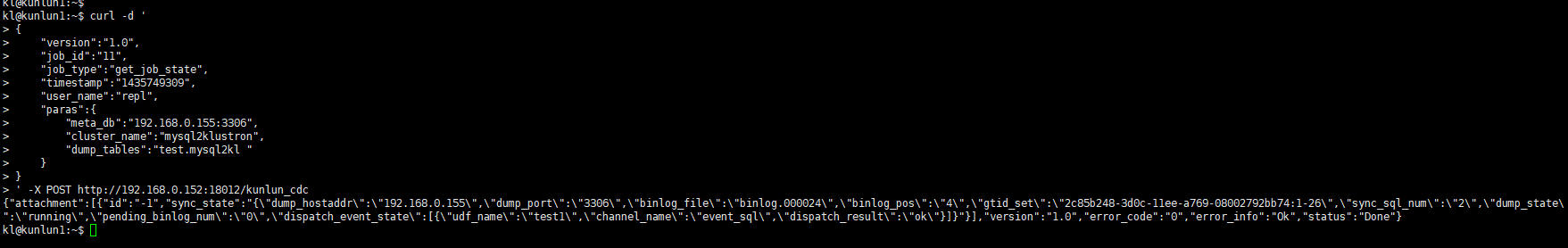
"cluster\_name":"mysql2klustron",

"dump\_tables":"test.mysql2kl "

}

}

' -X POST <http://192.168.0.152:18012/kunlun_cdc>



1. **关于插件**
   1. 关于缺省插件的说明

event\_file插件直接将CDC捕获的数据转成json写入文件。开发自定义插件时可以参考event\_file插件输出。

event\_file 插件输入参数为json内容具体写入的文件。添加dump任务时output\_plugins字段中输入参数，例如：

"plugin\_name":"event\_file", -- 插件名称

"plugin\_param":"/xx/event.log", -- 表示json内容写入该event.log文件

"udf\_name":"test1" -- 扩展字段

event\_sql支持将CDC捕获的数据转换成sql，写入目标db。目标db可以为Klustron集群或者开源mysql。默认写入Klustron集群，如果需要开源mysql需要在plugin\_param中添加is\_kunlun=0。event\_sql插件输入参数，添加dump任务时output\_plugins字段中输入参数，例如：

"plugin\_name":"event\_sql", --- 插件名称

"plugin\_param":"{\"hostaddr\":\"172.0.0.2\",\"port\":\"24002\",\"user\":\"abc\",\"password\":\"abc\",\"log\_path\":\"../log\"}",---event\_sql插件输入参数

"udf\_name":"test2" --- 扩容字段

特别说明：如果使用event\_sql将源数据写入Klustron节点，plugin\_param里面配置为Klustron计算节点mysql 端口，不能为pg端口。

* 1. 自定义插件开发以及挂载方法

1、需要在 Klustron 官网下载 Klustron CDC软件，在软件包的 include子目录下找到开发插件依赖的头文件dispatch\_event.h

2、继承CDispatchEvent类，实现init/execute/close三个虚函数接口，在dispatch\_event.h中有详细说明各个函数使用方法

3、讲代码在Linux操作系统中编译成 so 文件，放到kunlun\_cdc安装包的plugin目录下。

4、在kunlun\_cdc安装包的conf目录下kunlun\_cdc.cnf中的plugin\_so标签下添加新开发的插件名。

例如开发插件so名为event\_test.so，那么在kunlun\_cdc.cnf文件plugin\_so标签下添加内容如下：

plugin\_so = event\_file,event\_sql,event\_test

特别注意：插件名之间用逗号隔开

5、修改完配置文件，需要重启Klustron CDC集群所有CDC进程才可以生效

由于插件由用户自己开发，插件通常会需要输入参数，则在添加dump任务时在output\_plugins标签下输入该插件需要的参数。

【End】