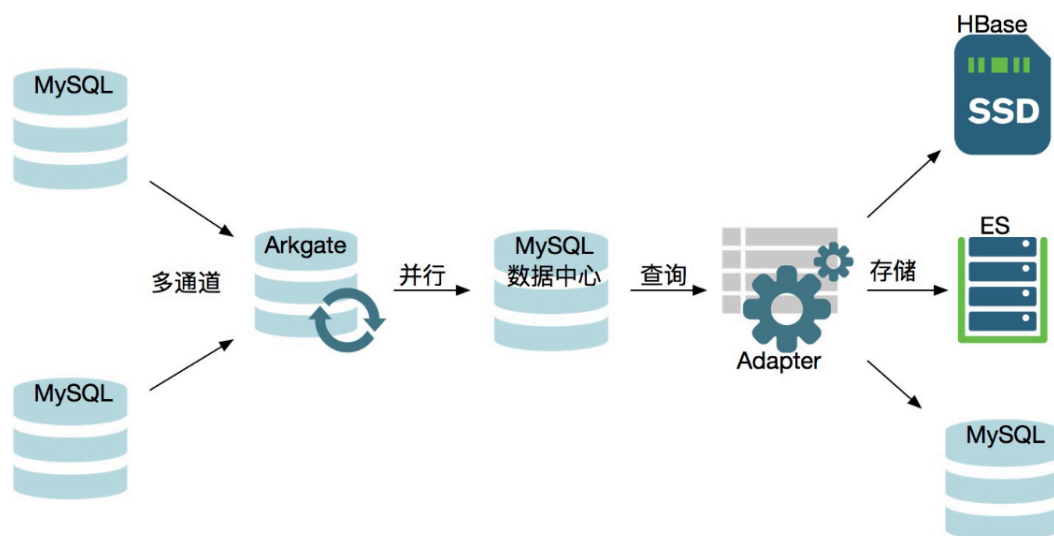


## 1. Arkcontrol 数据同步功能简介

Arkcontrol 数据同步功能是基于极数云舟自主研发的 Arkgate 的数据异构同步工具。Arkgate 是一款 MySQL 的插件，模拟了 MySQL 从库的 IO Thread，不断地 Dump MySQL 的 Binlog Events，实时地将 MySQL 数据库的增量更新同步到目标数据库中。目标数据库支持 HBase、Elasticsearch 及 MySQL 数据库，以解决不同人群及业务的需求。架构如下图所示：



图中各个节点说明如下：

### 在线 MySQL

最左边的 MySQL，可以是一个集群，也可以是单节点，都可以通过同一个 Arkgate 来做数据传输，一个传输任务称为一个通道，也就是可以有多个通道。要求线上 MySQL 必须记录 Binlog 并且设置 binlog\_format 为 ROW 模式。

### Arkgate

Arkgate 是一个 MySQL 插件，在 MySQL 内部不断运行着传输任务。将 Dump 过来的数据 Binlog Events 解析出来，转换为 JSON，并行地存储到 MySQL 中，这是一个数据的中间状态。这个数据库被称为数据中心。

### 数据中心

在这个数据库中，一个数据库对应一个通道，一个通道具有唯一的名称，即 Tunnel Name。在这个数据库中，存储了这个通道相关的所有信息，包括复制主从节点、独立配置参数、从节点对应的位置信息、主节点复制的最新位置、白

/黑名单及 JSON 数据。本系统到此为止也是可以使用的，业务可以自己来定制化地查这里面的数据，而如果为了简单方便，可以部署增强系统，即 Adapter 程序。

## Adapter 程序

已经做了几种目标数据库的 Adapter 程序，包括 Kafka, HBase、ES 及 MySQL（目前页面支持自动部署的为到 MySQL），这里可以通过配置文件实现表列的对应关系，最终实现实时异构的数据转换。

目前 Arkcontrol 中可以在已经搭建好的 gate 集群中添加创建同步任务,监测任务状态，完成同步任务启停，参数修改、配置等操作。订阅版包含分布式哨兵集群,可以监控同步链路状态,确保高可用。

## 2. Arkcontrol 同步任务创建

### 2.1 准备工作

Arkcontrol 1.4 版本及以后只需在部署中心部署 Arkgate Cluster 集群即可



之前版本需要手动搭建 Arkgate 集群,用于同步任务

#### 2.1.1 搭建数据库实例

使用 $\${installDir}/arkcontrol/software/arkgate$  下的 `ark-mysql57-5.7.18-1.el7.centos.x86_64.rpm`,搭建单节点实例（127.0.0.1\_3321，用户根据实际情况调整，这里仅举例说明）

```
rpm -ivh ark-mysql57-5.7.18-1.el7.centos.x86_64.rpm
/opt/ark/arkdb/scripts/mysql_install_file --port=3321 --target=mysql57 --installDir=/data
```

#### 2.1.2 拷贝 Arkgate 插件

拷贝  $\${installDir}/ArkControl/software/arkgate$  下的 `arkgate.so` 到部署实例

的 plugin\_dir(登陆实例\${installDir}/ArkControl/software/arkgate 查看)下

拷贝 \${installDir}/ArkControl/software/arkgate 下的 arkgate.key 文件到部署实例的 arkdb/3321/etc 下

### 2.1.3 安装 Arkgate 插件

```
mysql -uroot -P3321 -h127.0.0.1 -A
执行如下语句:
install plugin arkgate soname 'arkgate.so';
install plugin arkgate_datacenter_threads soname 'arkgate.so';
install plugin arkgate_datacenter_full_list soname 'arkgate.so';
install plugin arkgate_fliter_list soname 'arkgate.so';
install plugin arkgate_slave_list soname 'arkgate.so';
install plugin arkgate_datacenter_list soname 'arkgate.so';
install plugin arkgate_datacenter_options soname 'arkgate.so';
install plugin arkgate_datacenter_tables soname 'arkgate.so';
#查看 arkgate 相关参数确认安装成功
mysql> show global variables like '%arkgate%';
```

### 2.1.4 修改配置文件

修改 arkdb/3321/etc/my.cnf 添加

```
[mysqld]
arkgate_datacenter_host = 127.0.0.1 ##dc 地址
arkgate_datacenter_port = 3321  ##dc 端, dc 与 gate 公用一个 MySQL 实例
arkgate_datacenter_user = arkgate ##访问 dc 用户
arkgate_datacenter_password = arkgate_test ##访问 dc 密码
arkgate_license_file = /data/ark/arkdb/multi/3321/etc/arkgate.key ##License 验证
```

### 2.1.5 重启 MySQL 实例

```
/etc/init.d/mysql.server -P 3321 restart
```

### 2.1.6 账号授权

```
mysql -uroot -P3321 -h127.0.0.1 -A
grant all privileges on *.* to 'arkgate'@'%' identified by 'arkgate_test';
```

### 2.1.7 初始化 ETL 配置库表

```
CREATE DATABASE IF NOT EXISTS `etl_config`;

CREATE TABLE IF NOT EXISTS `etl_config`.`task_info` (

  `id` int(11) NOT NULL AUTO_INCREMENT,

  `task_name` varchar(128) NOT NULL DEFAULT "",

  `dc_ip` varchar(16) NOT NULL DEFAULT '127.0.0.1',

  `dc_port` int(11) NOT NULL DEFAULT '9991',

  `dc_dbname` varchar(32) NOT NULL DEFAULT "",

  `dc_user` varchar(32) NOT NULL DEFAULT "",

  `dc_pass` varchar(32) DEFAULT "",

  `dst_ip` varchar(16) NOT NULL DEFAULT '127.0.0.1',

  `dst_port` int(11) NOT NULL DEFAULT '3310',

  `dst_user` varchar(32) NOT NULL DEFAULT "",

  `dst_pass` varchar(32) NOT NULL DEFAULT "",

  `start_id` int(11) NOT NULL DEFAULT '1',

  `thread_num` int(11) DEFAULT '8',

  `limit_query_rows` int(11) NOT NULL DEFAULT '4000',

  `buffer_size` int(11) NOT NULL DEFAULT '8000',

  `exec_retry_num` int(11) NOT NULL DEFAULT '3',

  `exec_retry_interval` int(11) NOT NULL DEFAULT '3',

  `logger_file` varchar(128) NOT NULL DEFAULT 'mysql_etl.log',

  `double_active` tinyint(1) NOT NULL DEFAULT '0' COMMENT '双活开关，false 为关闭',

  `exec_mode` int(11) NOT NULL DEFAULT '0' COMMENT '执行 SQL 报错时的控制参数,默认值是 0.(0 跳过,1 覆盖执行,2 业务方提供异常处理接口,3 程序退出);',

  `biz_conflict_api` varchar(256) NOT NULL DEFAULT "",

  `rename_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '修改表名开关',

  `filter_opt_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '操作过滤开关',

  `filter_col_flag` tinyint(1) NOT NULL DEFAULT '0' COMMENT '列过滤开关',

  `create_time` timestamp NOT NULL DEFAULT '1970-01-01 09:00:00',

  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,

  PRIMARY KEY (`id`),
```

```

        UNIQUE KEY `uniq_taskname` (`task_name`)
    )    ENGINE=InnoDB    AUTO_INCREMENT=1    DEFAULT    CHARSET=utf8mb4
COMMENT='ETL 任务基准表';

CREATE TABLE `etl_config`.`etl_state` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `task_name` varchar(512) NOT NULL DEFAULT '' COMMENT '任务名称，不能重复',
    `is_consistent` tinyint(1) NOT NULL DEFAULT '0',
    `delay_time` int(11) NOT NULL DEFAULT -1 COMMENT 'etl 相对于 DC 的延迟时间，单位是秒',
    `qps` int(11) NOT NULL DEFAULT '0' COMMENT '不够精确的 qps',
    `checkpoint` bigint(20) NOT NULL DEFAULT '0',
    `binlog_ctime` timestamp NOT NULL DEFAULT '1973-01-01 08:00:00',
    `heart_beat` int(11) NOT NULL DEFAULT '0' COMMENT '心跳线程活着时，每隔一秒种执行 heart_beat++',
    `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (`id`),
    UNIQUE KEY `uniq_taskname` (`task_name`)
)    ENGINE=InnoDB    AUTO_INCREMENT=1    DEFAULT    CHARSET=utf8mb4
COMMENT='ETL 状态表';

CREATE TABLE IF NOT EXISTS `etl_config`.`rename_rule` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `task_name` varchar(128) NOT NULL DEFAULT '',
    `original_db` varchar(64) NOT NULL DEFAULT '',
    `original_table` varchar(64) NOT NULL DEFAULT '',
    `new_db` varchar(64) NOT NULL DEFAULT '',
    `new_table` varchar(64) NOT NULL DEFAULT '',
    `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE

```

```
CURRENT_TIMESTAMP,  
    PRIMARY KEY (`id`)  
    ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='修改表名';
```

```
CREATE TABLE IF NOT EXISTS `etl_config`.`rename_regexp` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `task_name` varchar(128) NOT NULL DEFAULT "",  
    `db_pattern` varchar(64) NOT NULL DEFAULT "",  
    `table_pattern` varchar(64) NOT NULL DEFAULT "",  
    `new_db` varchar(64) NOT NULL DEFAULT "",  
    `new_table` varchar(64) NOT NULL DEFAULT "",  
    `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
    PRIMARY KEY (`id`)  
    ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='修改表名,正则表达式来匹配原表名';
```

```
CREATE TABLE IF NOT EXISTS `etl_config`.`filter_opt` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `task_name` varchar(128) NOT NULL DEFAULT "",  
    `dbname` varchar(64) NOT NULL DEFAULT "",  
    `tablename` varchar(64) NOT NULL DEFAULT "",  
    `optype` varchar(16) NOT NULL DEFAULT "",  
    `action` varchar(128) NOT NULL DEFAULT "" COMMENT '多个动作作用英文逗号分割,如:  
ADD INDEX,DROP INDEX',  
    `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,
```

```

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COMMENT='操作过滤表';

CREATE TABLE IF NOT EXISTS `etl_config`.`filter_col` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `task_name` varchar(128) NOT NULL DEFAULT "",
  `dbname` varchar(64) NOT NULL DEFAULT "",
  `tablename` varchar(64) NOT NULL DEFAULT "",
  `column_names` varchar(64) NOT NULL DEFAULT "" COMMENT '要过滤多个字段,用英文逗号分割',
  `create_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `update_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COMMENT='字段过滤';

INSERT INTO `etl_config`.`task_info` (
  id, task_name, dc_ip, dc_port, dc_dbname, dc_user, dc_pass, dst_ip, dst_port, dst_user, dst_pass,
  start_id,
  thread_num, limit_query_rows, buffer_size, exec_retry_num, exec_retry_interval, logger_file,
  double_active, exec_mode,
  biz_conflict_api, rename_flag, filter_opt_flag, filter_col_flag) VALUES (
  1, 'etl_beijing', '118.190.89.67', 9991, 'dc_beijing', 'test_etl', 'test_etl', '118.190.89.67', 3311,
  'test_etl',
  'test_etl', 1, 8, 2000, 2000, 3, 3, 'mysql_etl.log', 0, 0, "", 1, 1, 1
);

# rename_rule 测试数据
#只改表名
INSERT INTO `etl_config`.`rename_rule` (id, task_name, original_db, original_table, new_table)

```

```

VALUES (1, 'etl_beijing', 'da1', 'ta1', 'ta');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, new_table) VALUES (2,
'etl_beijing', 'db', 'tb');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_table, new_table) VALUES (3,
'etl_beijing', 'tc1', 'tc');

#只改库名

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, original_table, new_db)
VALUES (4, 'etl_beijing', 'dd1', 'td1', 'dd');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, new_db) VALUES (5,
'etl_beijing', 'de1', 'de');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_table, new_db) VALUES (6,
'etl_beijing', 'tf1', 'df');

#库名表名都修改为指定名称

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, new_db, new_table)
VALUES (7, 'etl_beijing', 'dg1', 'dg', 'tg');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_table, new_db, new_table)
VALUES (8, 'etl_beijing', 'th1', 'dh', 'th');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, original_table, new_db,
new_table) VALUES (9, 'etl_beijing', 'di1', 'ti1', 'di', 'ti');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, original_table, new_db,
new_table) VALUES (10, 'etl_beijing', 'di1', 'ti2', 'di', 'ti');

INSERT INTO `etl_config`.`rename_rule`(id, task_name, original_db, original_table, new_db,
new_table) VALUES (11, 'etl_beijing', 'di2', 'ti1', 'di', 'ti');


#rename_regexp

INSERT INTO `etl_config`.`rename_regexp`(id, task_name, table_pattern, new_table) VALUES
(1, 'etl_beijing', '^rta', 'rta');

INSERT INTO `etl_config`.`rename_regexp`(id, task_name, db_pattern, new_db) VALUES (2,
'etl_beijing', '^rdb', 'rdb');

INSERT INTO `etl_config`.`rename_regexp`(id, task_name, db_pattern, table_pattern, new_db,
new_table) VALUES (3, 'etl_beijing', '^rdc', '^rtc[a-z]*rtcc$', 'rdc', 'rtc');

```



#filter\_opt 测试数据

```
INSERT INTO `etl_config`.`filter_opt`(id, task_name, dbname, tablename, optype, action)
VALUES
```

```
(1, 'etl_beijing', 'animal', 'dog', 'ALERTABLE', 'ADD COLUMN, ADD INDEX'),(2,
'etl_beijing', 'animal', 'dog', 'ALERTABLE', 'DROP COLUMN, DROP INDEX');
```

```
INSERT INTO `etl_config`.`filter_opt`(id, task_name, dbname, optype) VALUES (3, 'etl_beijing',
'animal1', 'CREATETABLE');
```

```
INSERT INTO `etl_config`.`filter_opt`(id, task_name, tablename, optype) VALUES (4, 'etl_beijing',
'cat', 'TRUNCATE');
```

```
INSERT INTO `etl_config`.`filter_opt`(id, task_name, optype) VALUES (5, 'etl_beijing',
'DROPTABLE');
```

#filter\_col 测试数据

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, tablename, column_names)
VALUES (1, 'etl_beijing', 'school', 'teacher', 'hobby,address'); #1.普通字段过滤
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, tablename, column_names)
VALUES (2, 'etl_beijing', 'school', 'teacher', 'column'); #2.特殊字段过滤;12.多个过滤规则作用在
同一张表上的字段过滤
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, tablename, column_names)
VALUES (3, 'etl_beijing', 'school', 'student', 'math, english'); #4.过滤多个字段
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, tablename, column_names)
VALUES (4, 'etl_beijing', 'school', 'student', 'idxcol'); #5.要过滤的字段上有索引
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, tablename, column_names)
VALUES (5, 'etl_beijing', 'school', 'student', 'xingming'); #6.要过滤的字段有发生了 change 字段
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, column_names) VALUES (6,
'etl_beijing', 'school', 'telephone'); #7.过滤一个库下所有表的该字段
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, tablename, column_names) VALUES (7,
'etl_beijing', 'all_table', 'col1, col2'); #8.过滤所有表名同为 ta 的一个或多个字段
```

```
INSERT INTO `etl_config`.`filter_col`(id, task_name, dbname, tablename, column_names)
VALUES (8, 'etl_beijing', 'company', 'all_table', 'col3, col3'); #9.过滤所有表名同为 ta 的一个或
```

多个字段

```
INSERT INTO `etl_config`.filter_col(id, task_name, dbname, tablename, column_names)
VALUES (9, 'etl_beijing', 'company', 'order', 'col100'); #10.如果要过滤的字段不再表中是否有影响
```

```
INSERT INTO `etl_config`.filter_col(id, task_name, dbname, tablename) VALUES (10,
'etl_beijing', 'company', 'library'); #13.column_names 为空,不过虑
```

```
INSERT INTO `etl_config`.filter_col(id, task_name, column_names) VALUES (11, 'etl_beijing',
'money');
```

## 2.2 创建任务

### 2.2.1 填写任务的源库和目的库信息

数据同步 > 创建同步

创建同步

○ 库源及目的库 > ○ 通道配置 > ○ 任务配置

数据源信息

\* 任务名称: 请输入任务名称, 最大长度为20位 **任务名称,方便用户标示**

\* 实例来源: **已录入实例** **未录入实例**

\* 实例名称: qq13-3\_3307 **需要同步的源数据库**

\* 数据库账号:

\* 数据库密码:

\* Binlog起始文件: mysql-bin.000001 **需要用户指定合法的binlog位置**

\* Binlog起始位置: 194

目的库信息

\* 目的端类型: **MySQL** **Kafka** **Redis** **ES** **Oracle**

\* 实例来源: **已录入实例** **未录入实例** **需要同步数据到目的数据库 目前web只支持MySQL**

\* 实例名称: qq13-3\_3307

\* 数据库账号:

\* 数据库密码:

### 2.2.2 通道配置(读源库配置)

高级配置可以指定修改同步参数,以及黑白名单等

收起配置

通道-参数配置

同步线程:

5

配置Arkgate的同步并发线程数

工作队列长度:

10000

源库架构类型:

原生MySQL | 阿里云RDS

是否包含表结构的全部信息:

包含 | 不包含

默认包含, 用于同步到MySQL场景。不包含适用于HBase、ES等场景

数据保留时长(小时):

168

Arkgate集群中数据保留时长以及清理策略

删除数据分配大小(行数):

5000

删除数据线程数:

5

通道-过滤配置

白名单: 只同步符合要求的表数据。黑名单: 忽略符合要求的表数据。两者只能配置一种

类型:

黑名单 | 白名单

数据库名称

database name

表名称

table name

删除

添加

通道-数据源高可用配置

备注: 数据源高可用配置, 用于当同步源数据库出现宕机后, 或者别的原因导致不可用的时候, 如果配置了高可用, 通道可以自动的切换到可用的从库, 继续同步, 确保同步正常。

\* 实例来源:

已录入实例 | 未录入实例

\* 实例名称:

请选择

删除

添加源从库

### 2.2.3 任务配置(写目的库配置)

仅展示用户设置的通道名称

数据同步 > 创建同步

创建同步

库源及目的库

通道配置

任务配置

预检查

任务基本信息

通道名称:

arkgate\_demo

仅展示用户配置的通道名称

高级配置

上一步

预检查

高级设置可以配置写目的库时的并发,表重命名,操作过滤等策略

收起配置

任务高级信息

工作线程:

8

写入目的库时并发数

开始位置:

1

从数据中心拉数据的开始位置,1表示从头拉取

工作步长:

4000

每次从数据中心拉取数据行数

缓存大小:

8000

双活开关:

OFF

用于双活模式,具体使用咨询技术云舟客服

出错处理:

跳过 | 重试执行 | API模式 | 退出

双活模式下,多写冲突处理方式

规则配置-表重命名配置

数据库名称(源库):

database name

表名称(源库):

table name

=>

数据库名称(目的库):

database name

表名称(目的库):

table name

删除

添加

规则配置-列过滤

备注: 多个列名, 请按逗号分隔

数据库名称:

database name

表名称:

table name

列名称列表 (源库):

多个列名, 请按逗号分隔

删除

添加

规则配置-DDL过滤

库名称:

表名称:

DDL类型:

DDL具体类型:

### 2.2.4 预检查及提交

配置完成后点击预检查



如果检测不通过,可以点击失败,显示详情



如果检查通过则可以点击提交,完成任务创建



提交后等待几秒钟(创建同步任务比较耗时),会自动跳转到详情页面

数据同步 > 同步列表 > 同步详情

基本信息		
同步任务名称: arkgate_demo	通道名称: arkgate_demo	通道状态: 正常
同步延迟时间(秒): 0	同步工作集群: 10.0.0.107_9991	任务出错时间:
任务异常信息:		
源库信息		
实例名称: sk1_3306	数据库地址: 10.0.0.131	数据库端口: 3306
Binlog起始文件: mysql-bin.000023	Binlog起始位置: 54921397	
目的库信息		
实例名称: sk3_3306	数据库地址: 10.0.0.133	数据库端口: 3306
实例类型: MySQL		
通道信息		
同步工作集群	黑白名单	同步参数配置
同步工作集群:	10.0.0.107_9991	
数据库地址:	10.0.0.107	
数据库端口:	9991	
数据库账号:	arkgate	
任务-规则信息		
基本信息	表库命名	列过滤
DDL过滤		

### 3 Arkcontrol 同步任务管理

#### 3.1 进入数据同步，查看同步列表

导入中心  
部署中心  
主机管理  
MySQL管理  
性能调优  
系统巡检  
备份中心  
迁移中心  
数据同步  
新建同步  
同步列表  
用户管理  
产品规划

数据同步 > 同步列表

任务名称	通道名称	源库地址	源库端口	目的库地址	目的库端口	操作
arkgate_demo	arkgate_demo	10.0.0.131	3306	10.0.0.133	3306	详情
eee	eee	10.0.0.131	3306	10.0.0.133	3306	详情
cccc	cccc	10.0.0.131	3306	10.0.0.131	3306	详情
qcj_gate_demo7	qcj_gate_demo7	10.0.0.87	3307	10.0.0.204	3306	详情
12	niceshi	10.0.0.131	3306	10.0.0.131	3306	详情
第十次	xpten	10.0.0.131	3306	10.0.0.131	3306	详情
qcj_gate_demo	qcj_gate_demo5	10.0.0.87	3307	10.0.0.204	3306	详情
bbbb	bbbb	10.0.0.131	3306	10.0.0.131	3306	详情
aaaa	aaaa	10.0.0.131	3306	10.0.0.131	3306	详情
qcj_demo	qcj_gate_ext_test	10.0.0.87	3307	10.0.0.88	3306	详情

共有108条, 每页显示: 10条

1 2 3 4 5

#### 3.2 点击详情进入详情页面

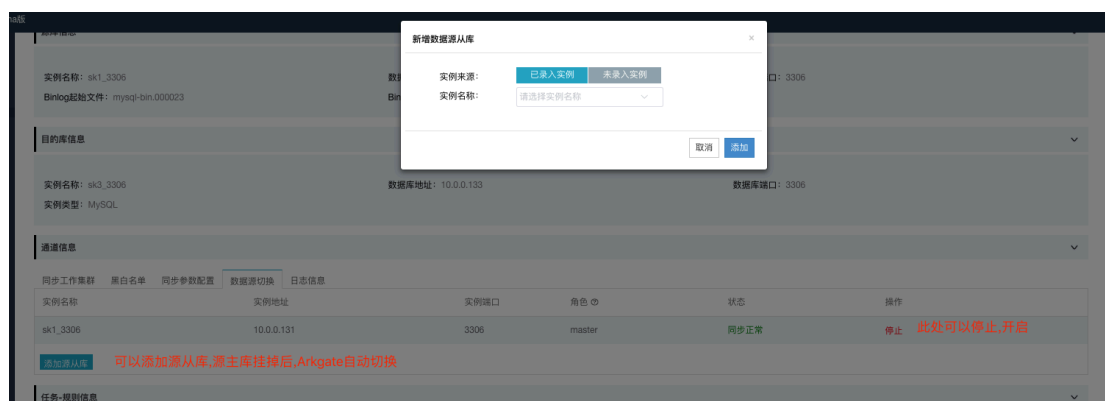
#### 3.3 修改通道信息(读取源库配置)

##### 3.3.1 添加黑白名单



### 3.3.2 数据源切换

数据源切换, 可以开始, 关闭同步任务



### 3.4 修改任务-规则信息(写入目的库配置)

