使用 HVR 从本地 MySQL 将数据实时复制到 AWS Aurora

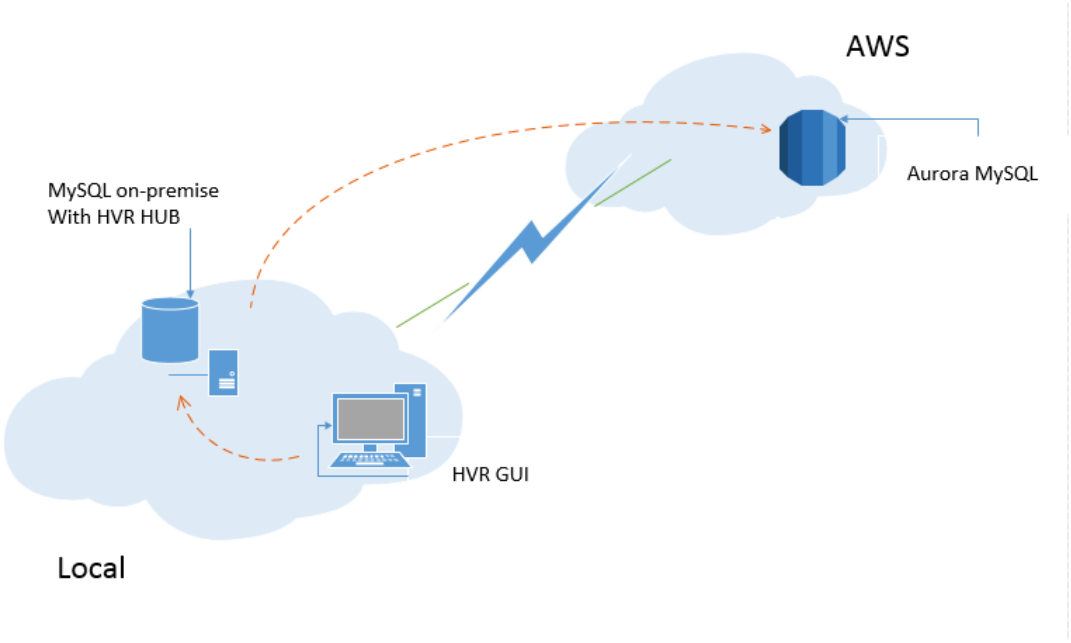
操作手册

目录

一、 **环境介绍** 3
1.1 拓扑图 3
1.2 基础信息 3
二、 **安装 HVR Hub** 3
2.1 配置环境变量 3
2.2 创建相关目录并上传文件 4
三、 **配置源端与 HUB 端数据库** 4
3.1 启用 binlog 4
3.2 配置 Hub 数据库 4
3.3 配置源数据库 4
四、 **配置目标端数据库（Aurora）** 5
4.1 更改 Aurora 数据库集群的时区 5
五、 **配置 HUB** 6
5.1 安装 gui 客户端 6
5.2 注册 HUB 7
5.3 创建 location 8
六、 **创建复制链路及 action** 10
6.1 创建复制链路 10
6.2 创建 location 组 11
七、 **定义需要复制的表** 12
八、 **定义 action** 13
8.1 为源 location 定义 action 13
8.2 为目标 location 定义 action 14
九、 **hvr initialize 生成复制任务** 16
十、 **hvr refresh 复制存量数据** 18
十一、**使用 hvr compare 验证数据** 20
十二、**启用实时复制** 22

一、环境介绍

1.1 拓扑图



AWS Aurora 是一种与 MySQL 和 PostgreSQL 兼容的关系数据库，专为云而打造，既具有传统企业数据库的性能和可用性，又具有开源数据库的简单性和成本效益。
本手册将指导用户如何从本地数据中心自建的 mysql 数据库迁移到 AWS 的 Aurora 数据库。

1.2 基础信息

1.2.1 源端数据库（mysql）

192.168.3.200 centos6.9 mysql 5.6.40

1.2.2 HVR Hub（安装在源端数据库服务器上）

192.168.3.200 centos6.9 mysql 5.6.40

1.2.3 目标端数据库（Aurora）

hvrtest.cxrlbso32bmm.rds.cn-northwest-1.amazonaws.com.cn Aurora 引擎 mysql5.6

1.2.4 图形管理工具

任意一台带图形界面的 windows 或者 linux 客户端

1.2.5 HVR Hub 支持的操作系统

HVR Hub 可以安装在任意的机器上，既可以是源端，也可以是目标端，或者一台独立的服务器。操作系统支持 windows、linux、unix。

二、安装 HVR Hub

2.1 配置环境变量

在源数据库上，以 mysql 用户为例

编辑用户主目录下的.bash_profile，添加如下内容

export HVR_HOME=/backup/hvr/hvr_home
export HVR_CONFIG=/backup/hvr/hvr_config
export HVR_TMP=/tmp
export PATH=$HVR_HOME/bin:$PATH

2.2 创建相关目录并上传文件
mkdir –p /backup/hvr/hvr_home
mkdir /backup/hvr/hvr_config
chown –R mysql:mysql /backup

上传文件到 hvr_home 目录
tar -zxvf hvr-5.3.1_13-linux_glibc2.5-x64-64bit.tar.gz
启动 listener 服务
hvrremotelistener -d -N 4343

上传授权文件 hvr.lic 到/backup/hvr/hvr_home/lib 目录

三、配置源端与 HUB 端数据库
3.1 启用 binlog
如已启用，则跳过此步骤
在源数据库服务器上编辑/etc/my.cnf
在【mysqld】下添加
log-bin = binlog
server-id = 1
binlog_format = row

重新启动 mysql
service mysqld stop
service mysqld start

3.2 配置 Hub 数据库
创建 Hub 的数据库及 hvr 用户，用来存储 Hub 的相关工作信息
mysql> create database `hvrhub` character set utf8;

创建用来连接的用户并授予权限
mysql> grant all on hvrhub.* to hvr@"%" Identified by "hvr";

3.3 配置源数据库
创建用来作为数据源的数据库 source 及 source 用户
mysql> create database `source` character set utf8;
mysql> grant all on source.* to source@"%" Identified by "source";
mysql> grant REPLICATION CLIENT on *.* to source;
mysql> grant REPLICATION SLAVE on *.* to source@"%";

mysql> grant select on *.* to source@"%";

创建若干表并插入一些数据用于测试

```
mysql> show tables;
+------------------+
| Tables_in_source |
+------------------+
| departments      |
| employees        |
+------------------+
```

```
mysql> select * from employees;
+--------+------------+------------+-----------+------------+
| emp_no | birth_date | first_name | last_name | hire_date  |
+--------+------------+------------+-----------+------------+
|      1 | 1956-02-14 | tom        | zhang     | 1995-01-04 |
|      2 | 1976-03-04 | jerry      | yao       | 2005-12-04 |
|      3 | 1996-03-04 | cati       | huang     | 2008-11-24 |
+--------+------------+------------+-----------+------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from departments;
+---------+-----------------+
| dept_no | dept_name       |
+---------+-----------------+
| d002    | Finance         |
| d001    | Human Resources |
+---------+-----------------+
```

四、配置目标端数据库（Aurora）

4.1 更改 Aurora 数据库集群的时区

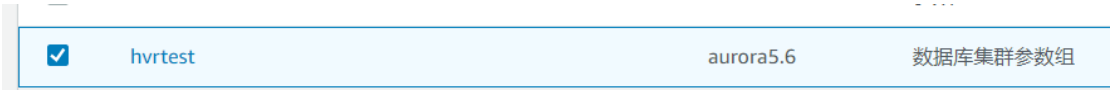
4.1.1 查询当前数据库实例时间，默认是 utc，两端时区应保持一致

```
mysql> select now() as Systemtime;
+---------------------+
| Systemtime          |
+---------------------+
| 2018-08-14 02:37:12 |
+---------------------+
1 row in set (0.06 sec)
```

4.1.2 创建数据库集群参数组

选择系列、类型，输入组名称创建

点击刚创建的参数组进行编辑



搜索 time_zone 参数并编辑成 Asia/Shanghai，然后将该参数组附加到集群。

4.1.3 创建目标端数据库

在 Aurora 上

mysql> create database `test` character set utf8;

用户这里使用创建 Aurora 实例时创建的用户 hvr

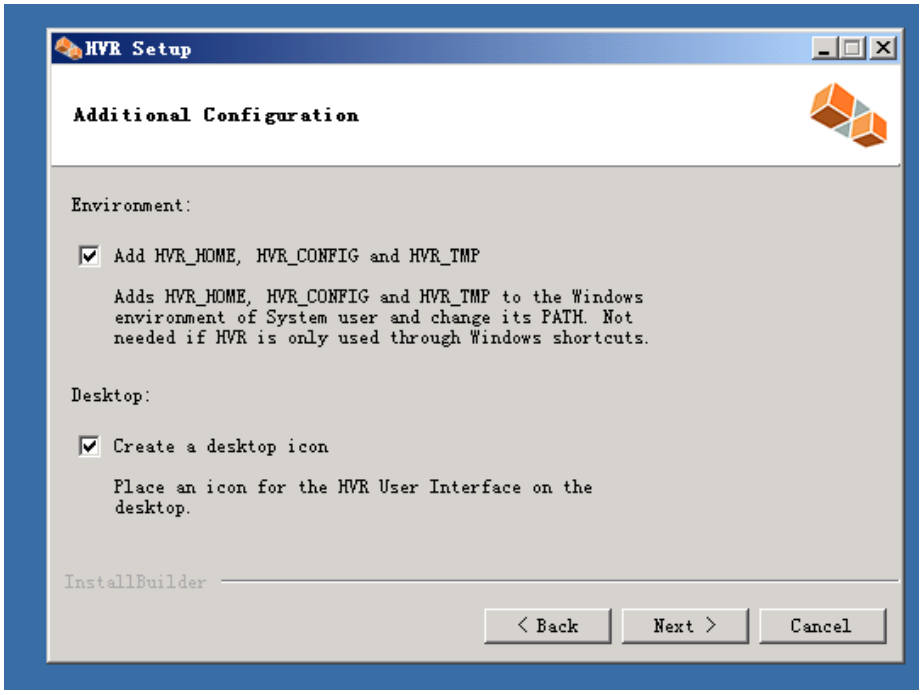
五、配置 HUB

5.1 安装 gui 客户端

使用任意一台带图形界面的客户端，安装 perl 包和 hvr 主程序，勾选如图选项，其余默认

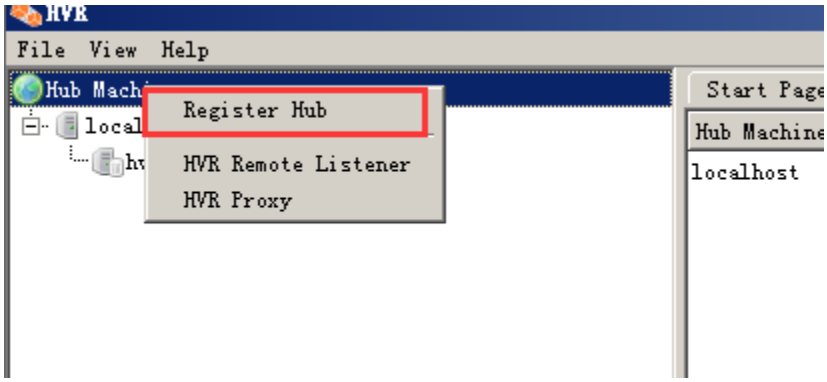
ActivePerl-5.20.2.2001-MSWin32-x64-298913

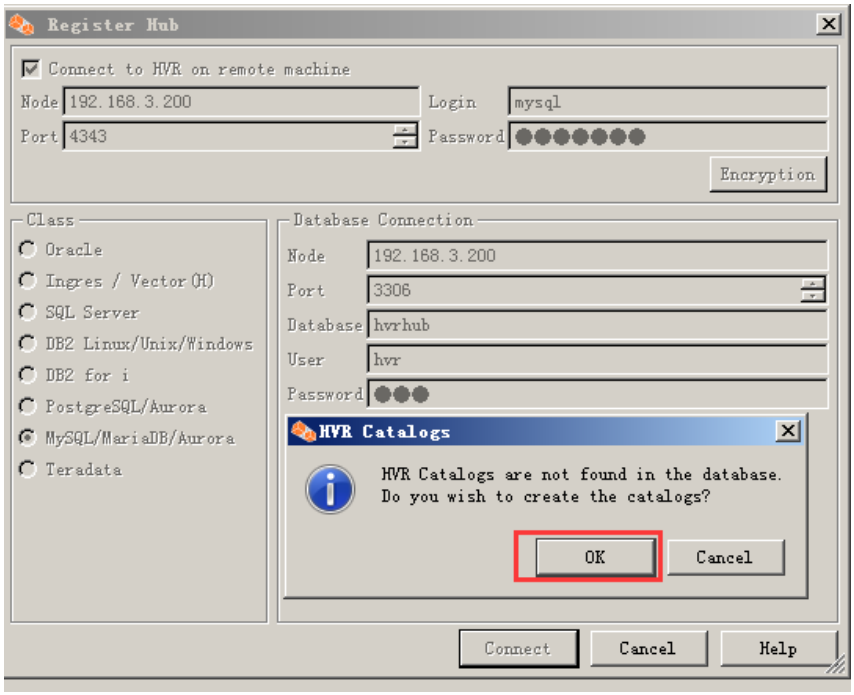
hvr-5.3.1_13-windows-x64-64bit-setup



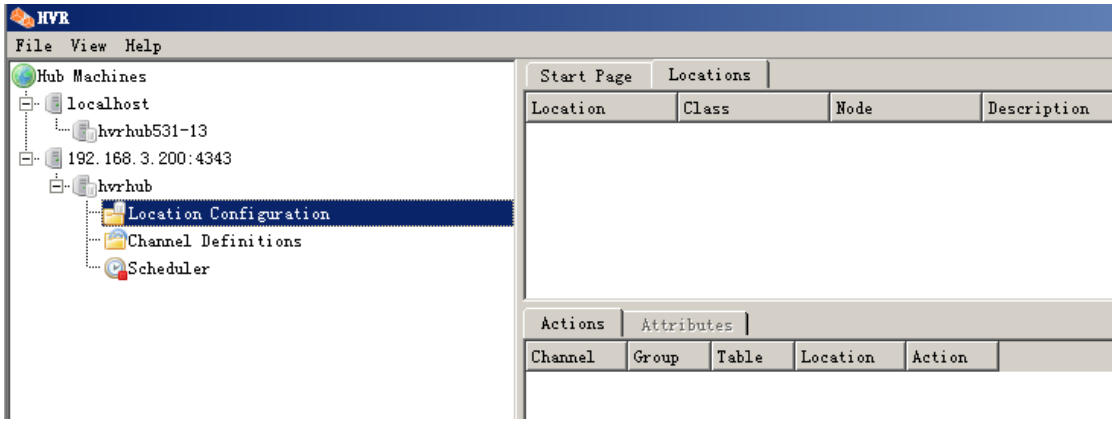
5.2 注册 HUB

5.2.1 运行 hvr 程序，进入程序界面，右键 Hub Machines，Register Hub





如上图设置，连接并创建 catalogs

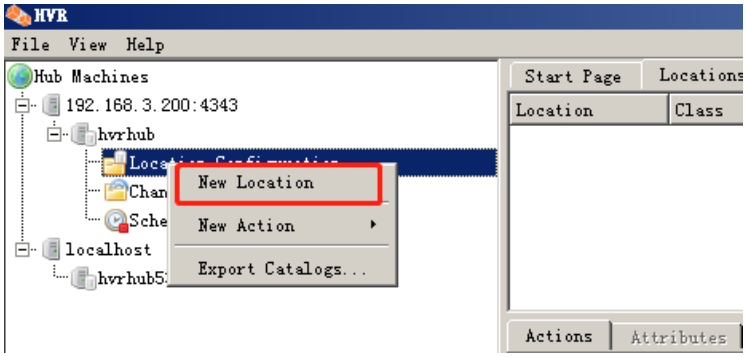


注册完成

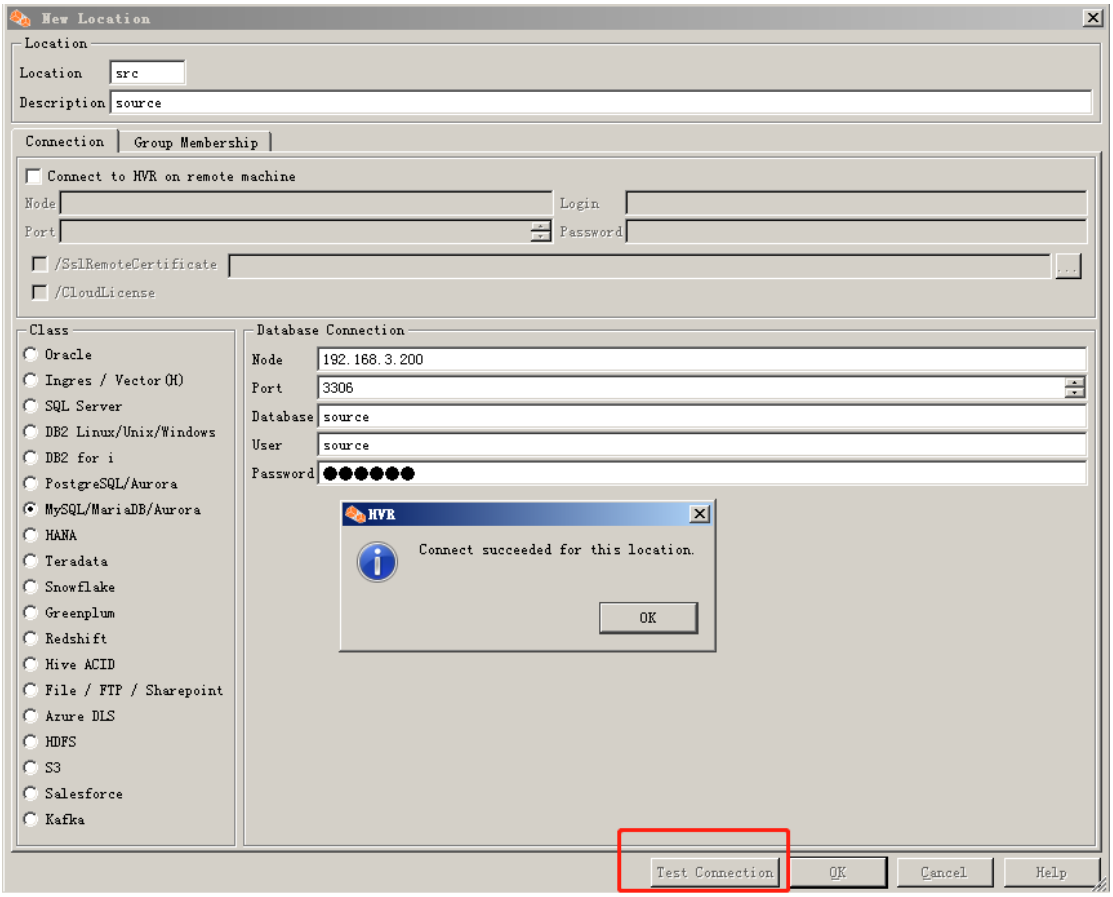
5.3 创建 location

5.3.1 创建名为 src 的源 location

右键点击 location configuration，选择 new location。（可以把 location 理解为 db host）



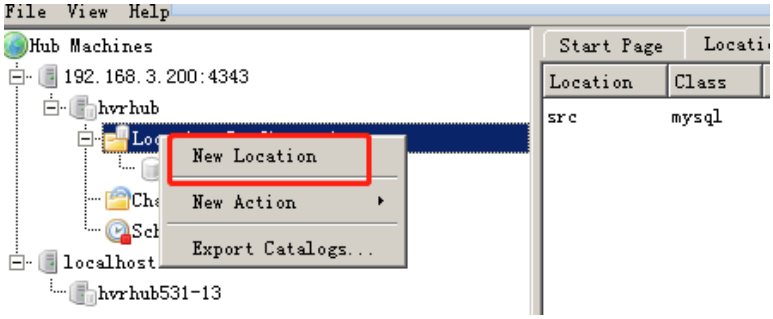
输入 location 的名称，数据库类型，IP 地址，用户密码等信息。因为这里 hub 与源数据库在同一主机上，所以无需勾选 connect to hvr on remote machine

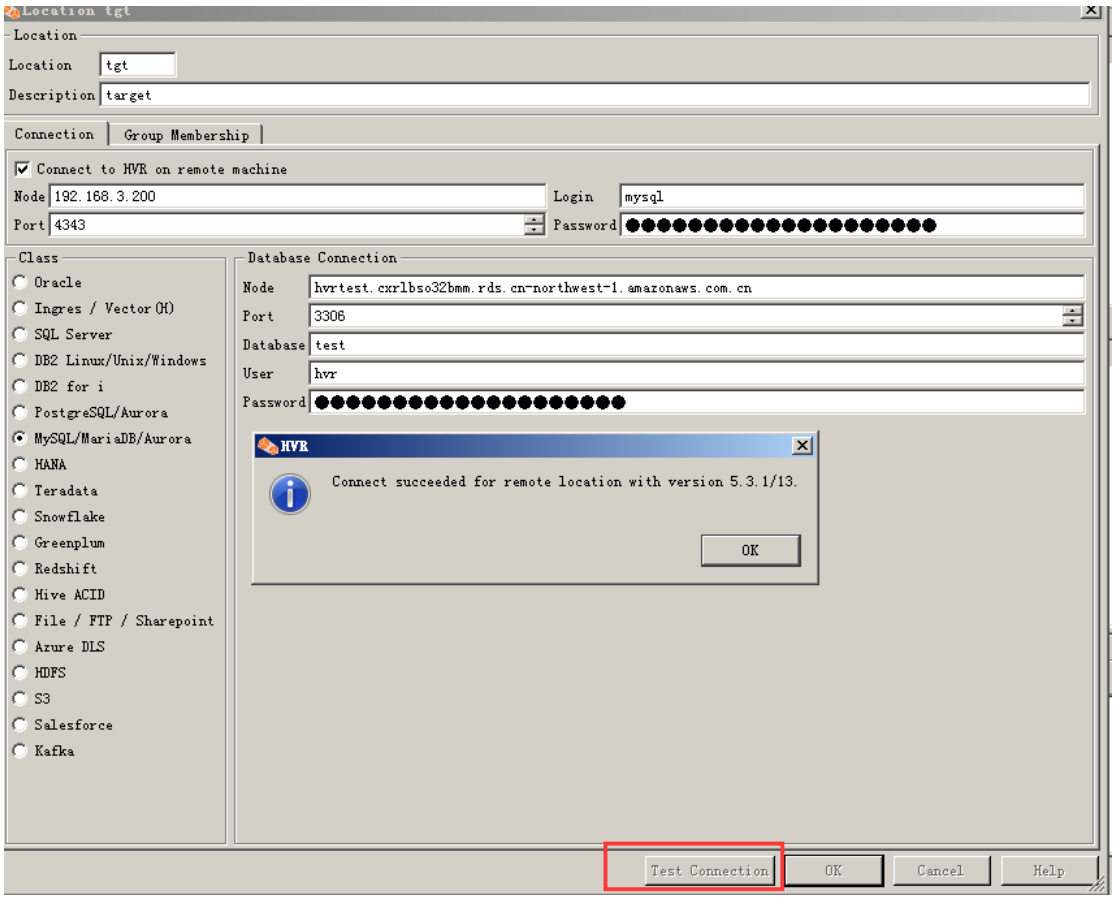


点击 test connection 测试连接，ok 则完成创建

5.3.2 创建名为 tgt 的目标 location

右键点击 location configuration，选择 new location。



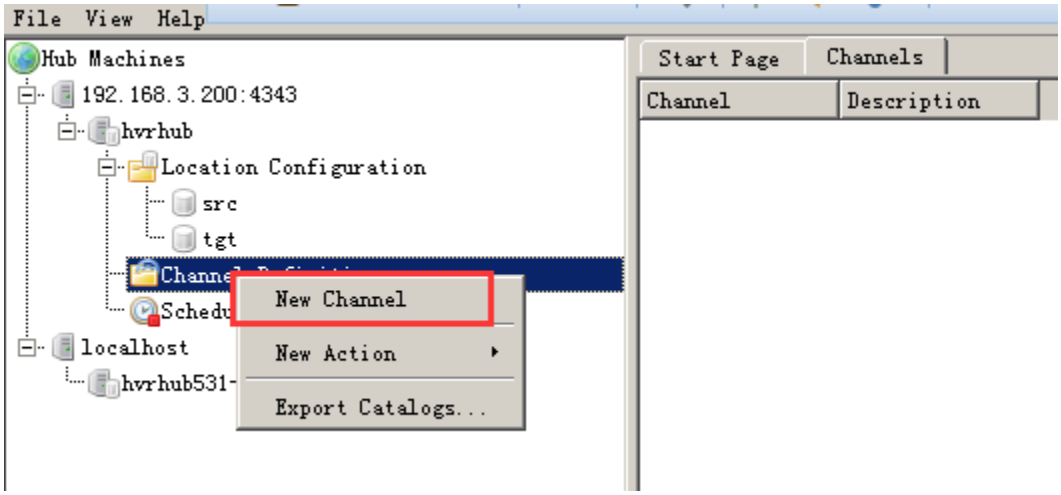


需要 192.168.3.200 这台 host 能具有访问 rds 的网络权限，勾选 connect to hvr on remote machine，输入 hub 所在机器 ip，端口号（默认 4343），启动 hvrremotelistener 的用户和密码。

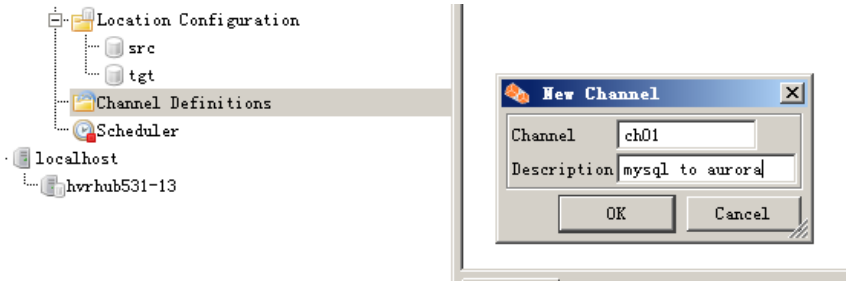
六、创建复制链路及 action

6.1 创建复制链路

如图右键选择 new channel



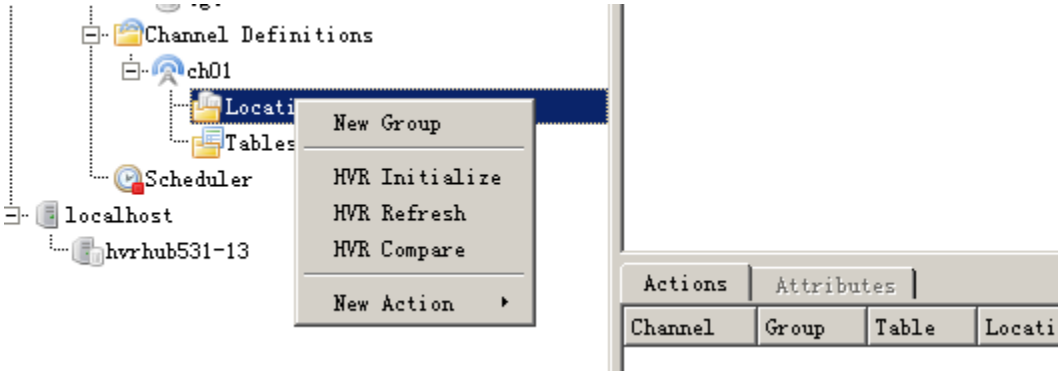
输入链路名称，描述，ok 完成



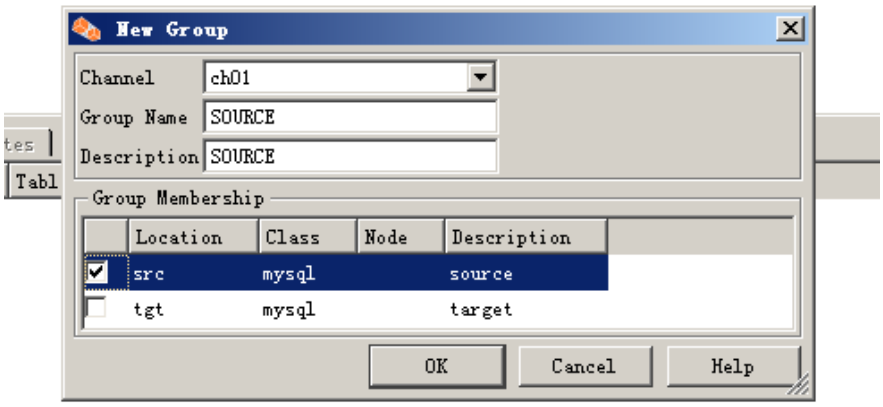
6.2 创建 location 组

6.2.1 创建源 location 组

展开 ch01，右键 location groups，选择 new group

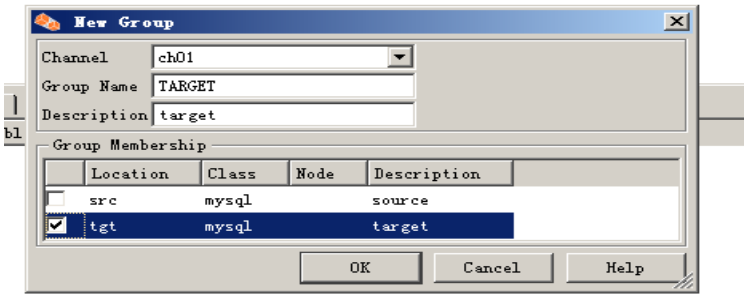


输入组名，描述，下方组成员中勾选之前在 location 里创建的 src，ok 完成



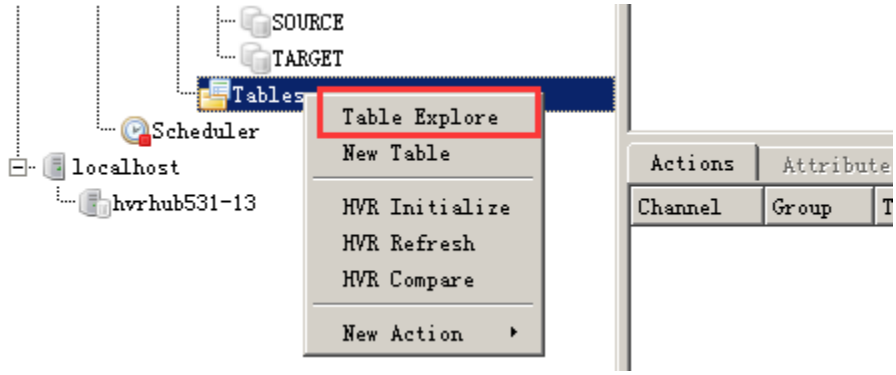
6.2.2 创建目标 location 组

继续 new group，输入组名，描述，下方勾选之前创建的 tgt

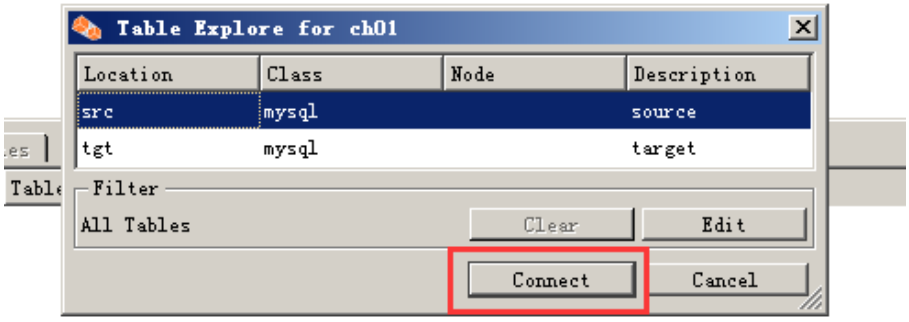


七、定义需要复制的表

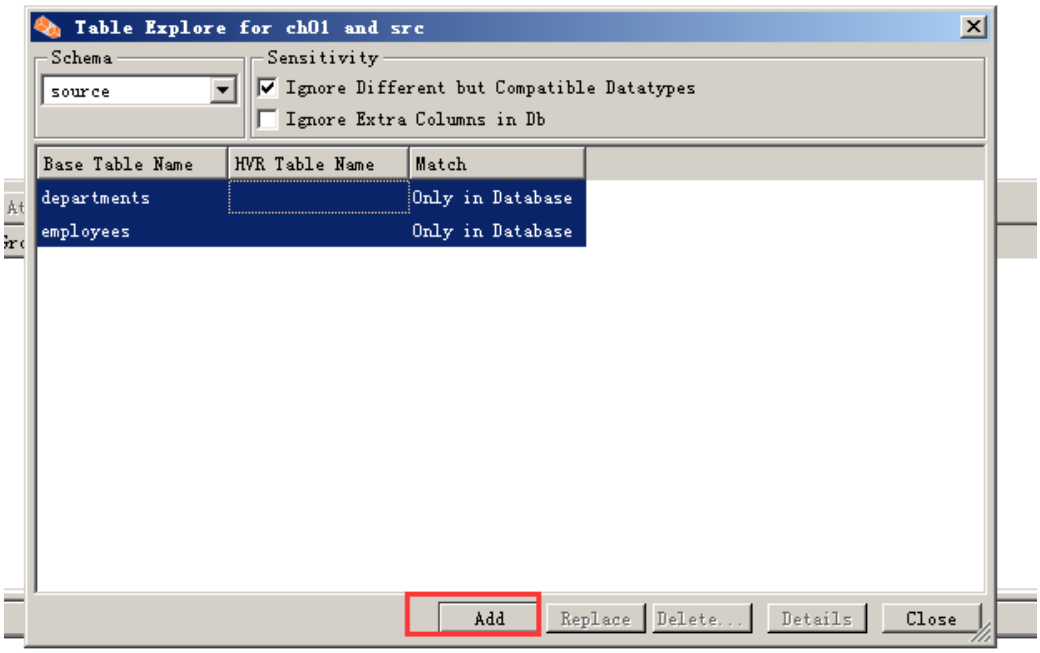
右键 tables，选择 table explore



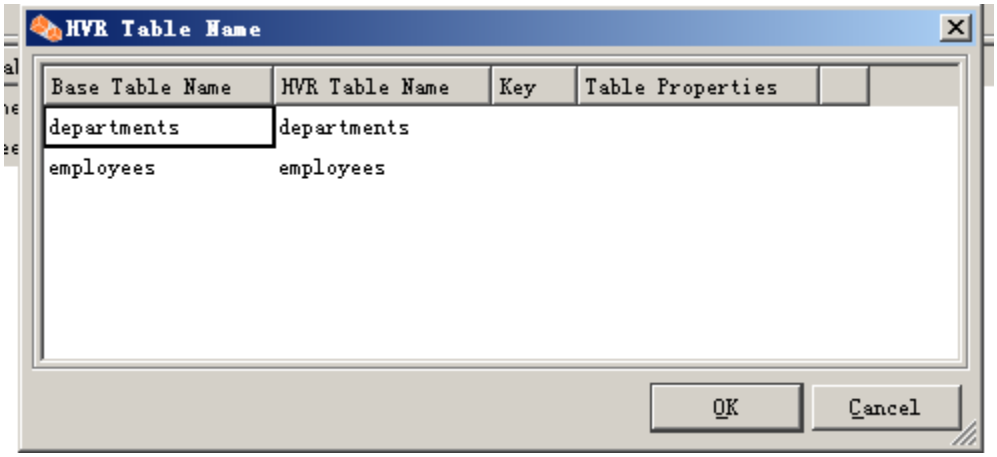
选中源 location src，点击 connect



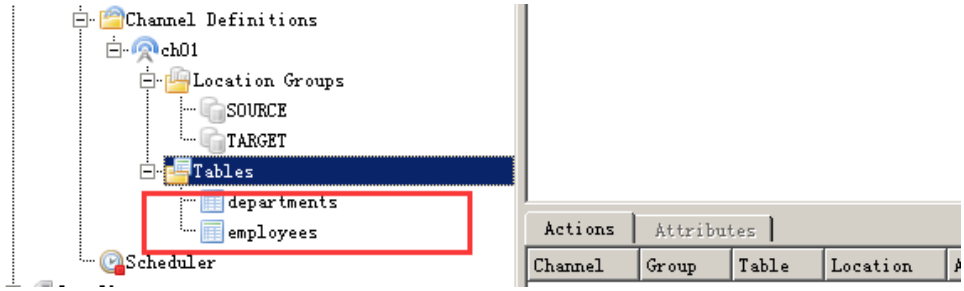
选中需要复制的表，点击 add



点击 ok 继续



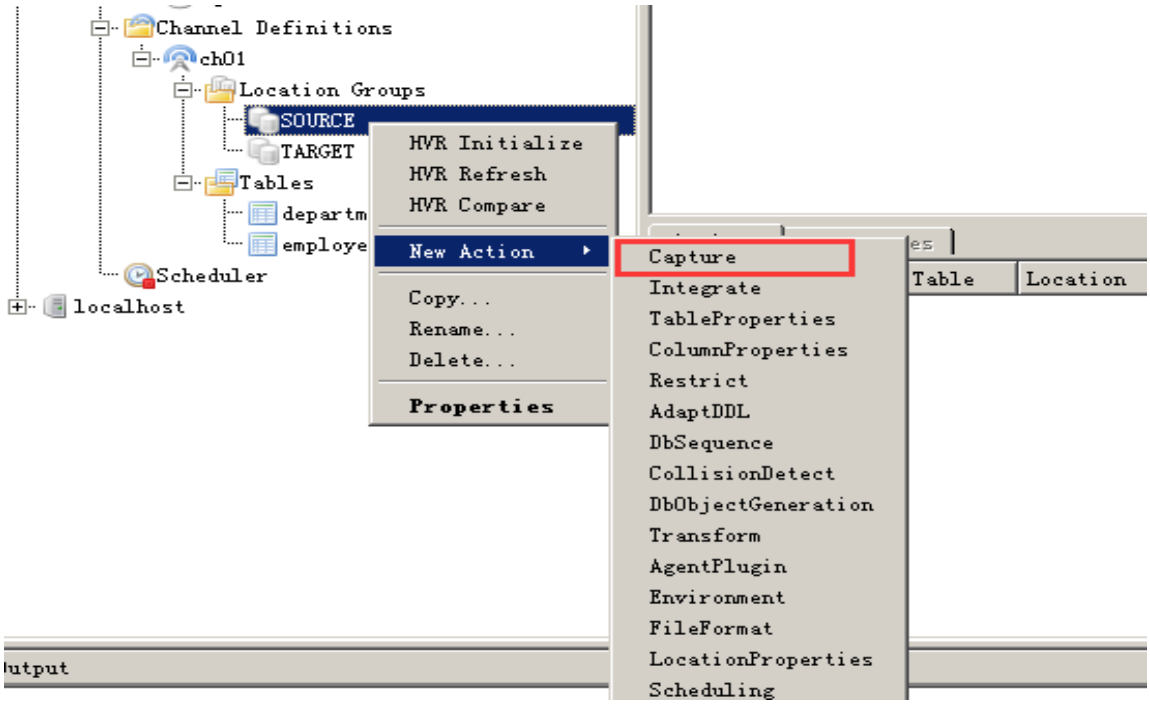
添加后如下图



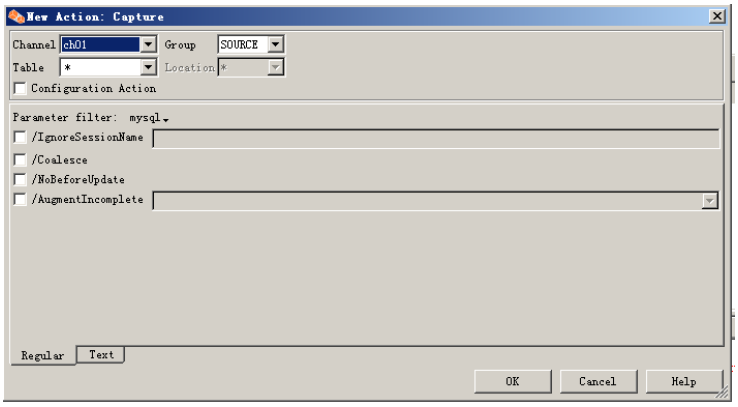
八、定义 action

8.1 为源 location 定义 action

右键 source location 组，new action 选择 capture

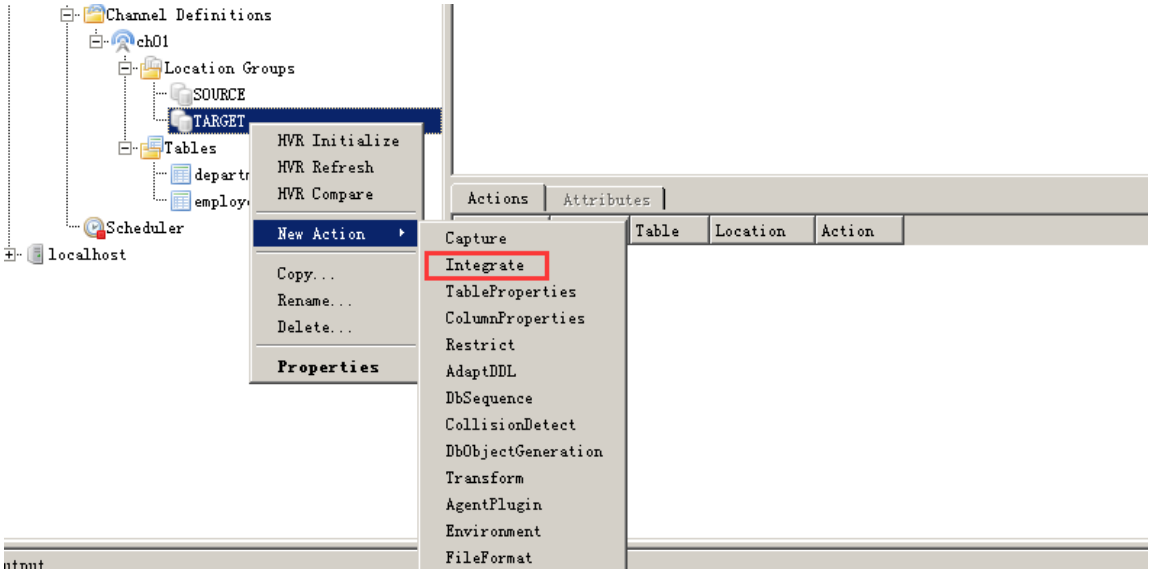


默认选项，ok 完成

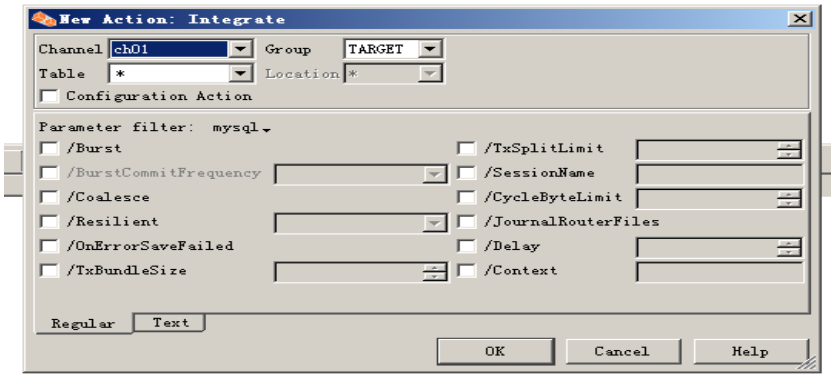


8.2 为目标 location 定义 action

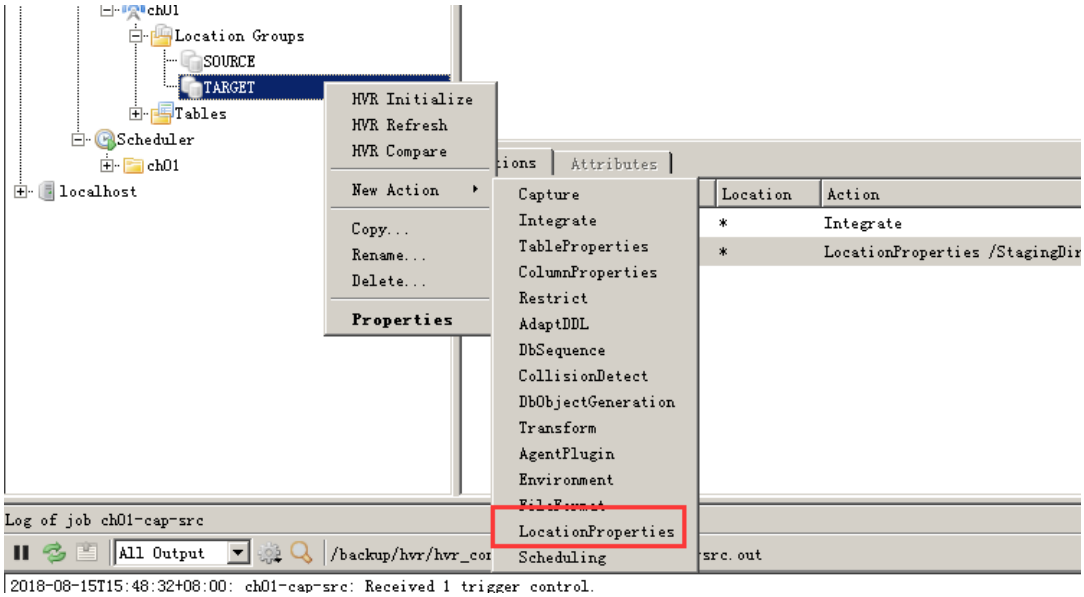
8.2.1 右键 target location 组，new location 组选择 integrate



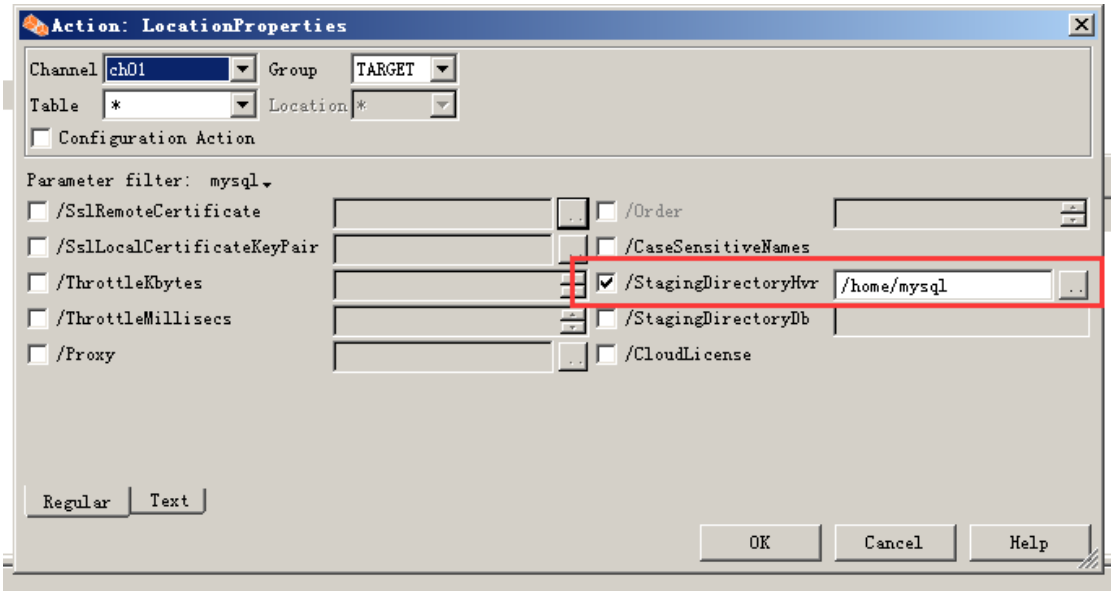
参数默认



8.2.2 继续添加 action，new action 选择 LocationProperties

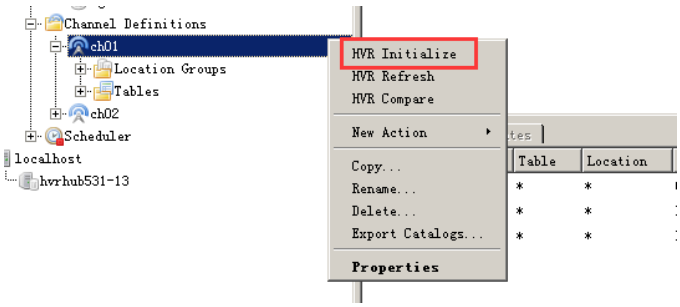


勾选 StagingDirectoryHvr，设定一个缓存目录，需要有读写权限。这里设为 hub 本机的 /home/mysql 目录。

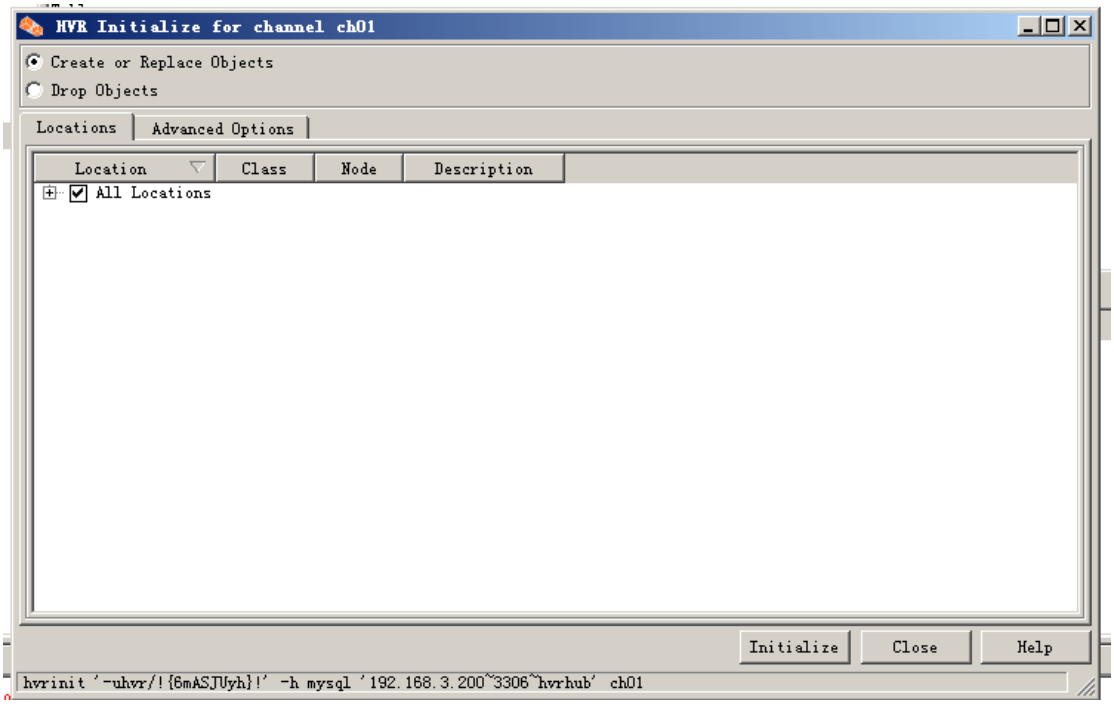


九、hvr initialize 生成复制任务

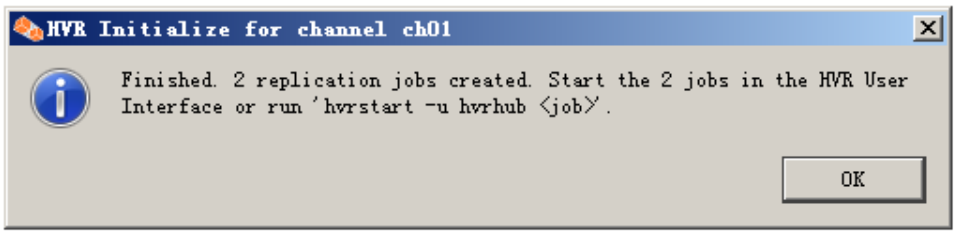
右键复制链路 ch01，选择 HVR initialize



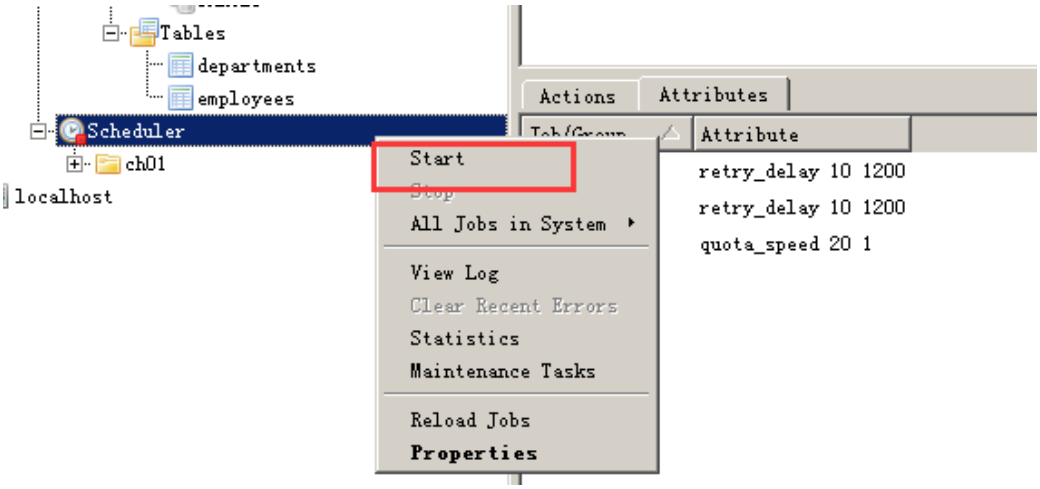
点击 initialize



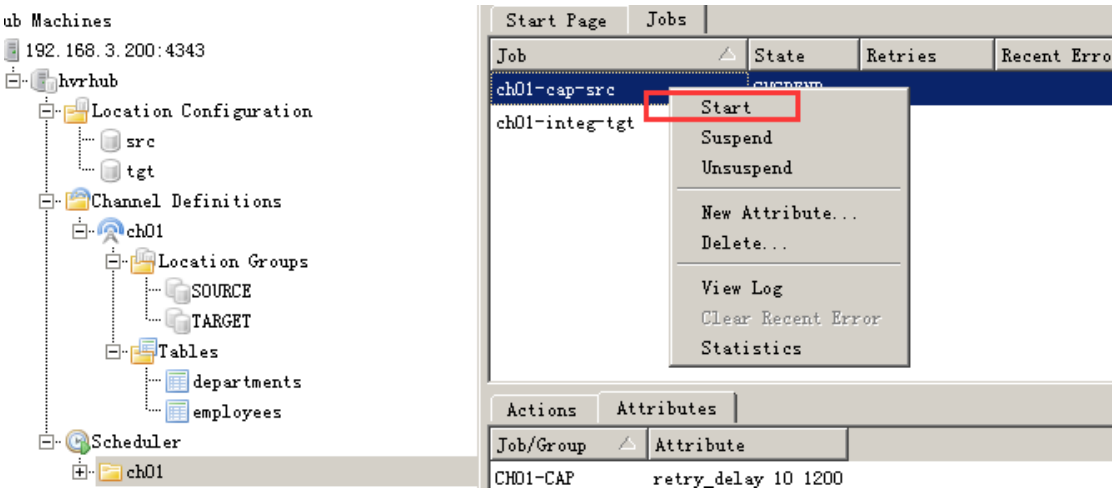
创建完成



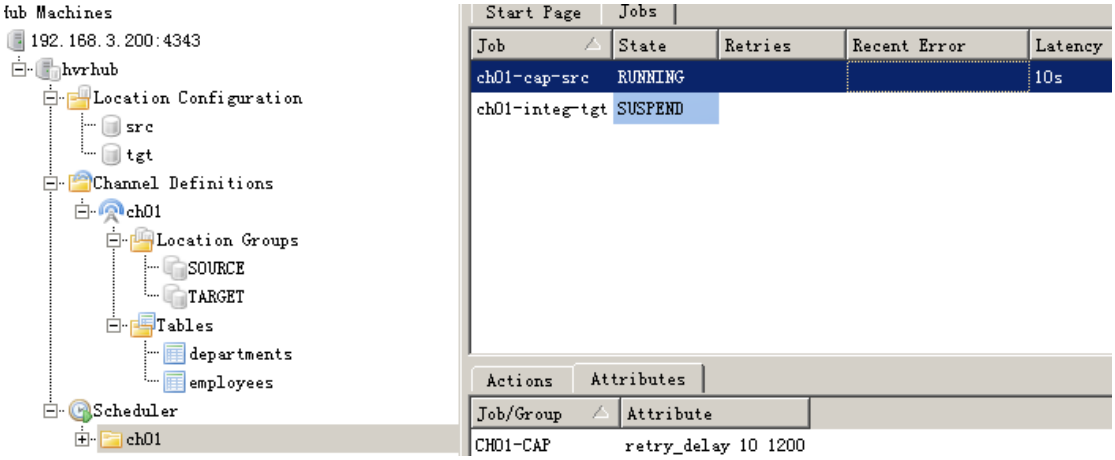
右键点击 scheduler，start 启动调度



点击 ch01，启动 capture 任务

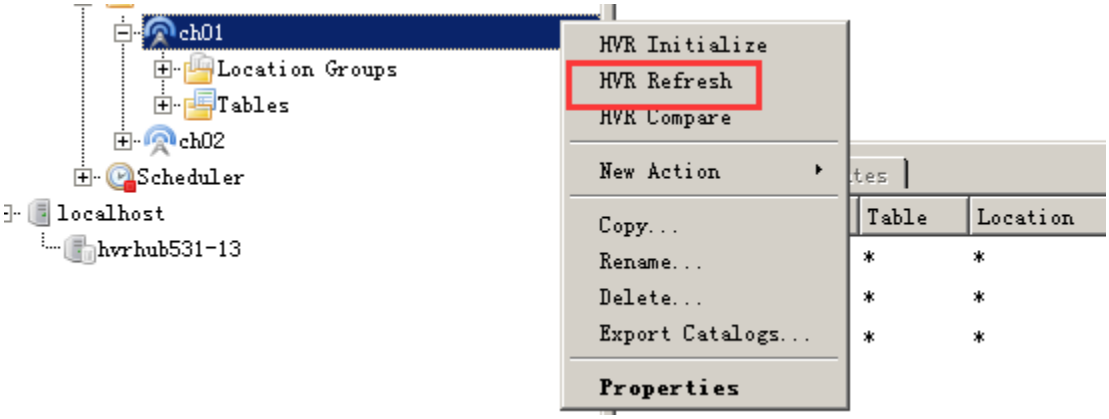


任务状态

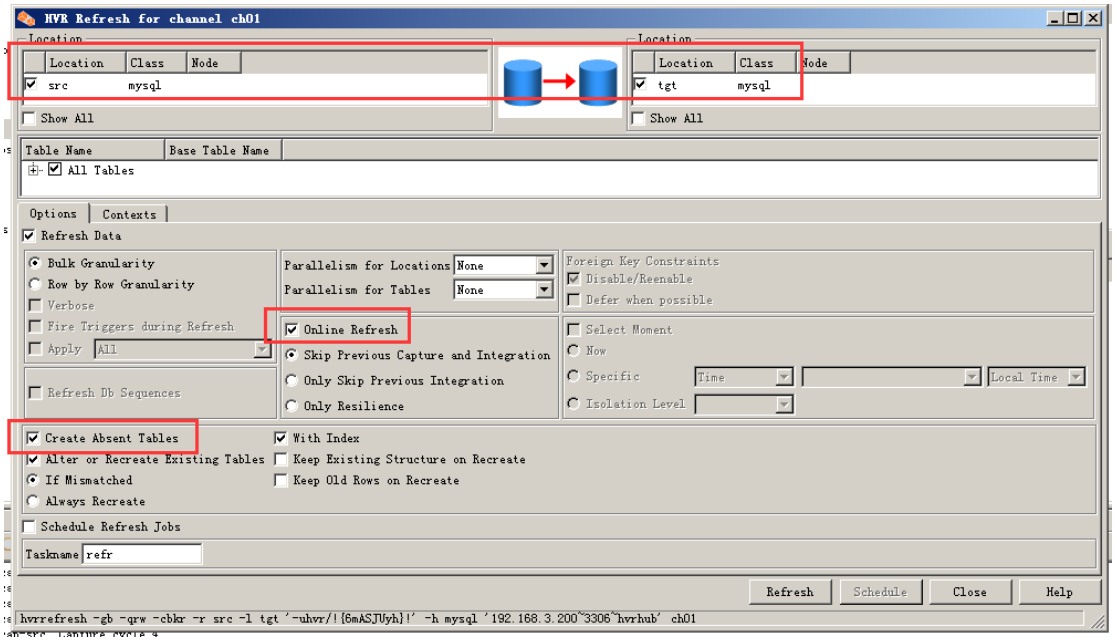


十、hvr refresh 复制存量数据

右键复制链路 ch01，选择 HVR refresh

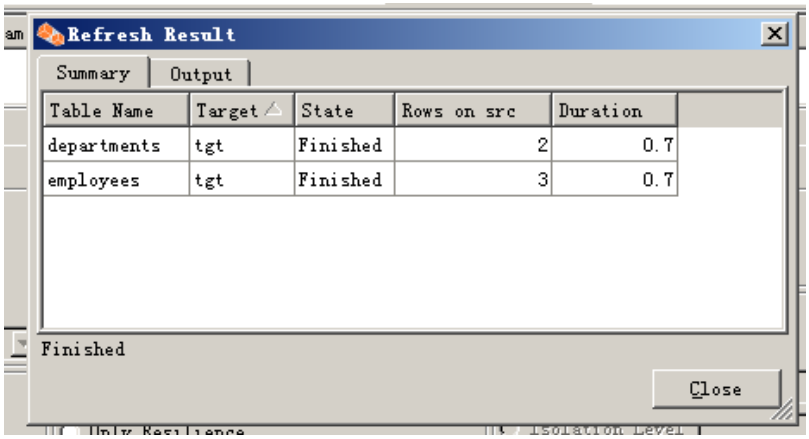


确认数据复制方向，勾选 create absent tables 与 online refresh，点击 refresh



yes，启动复制

复制完成



在目标端验证数据

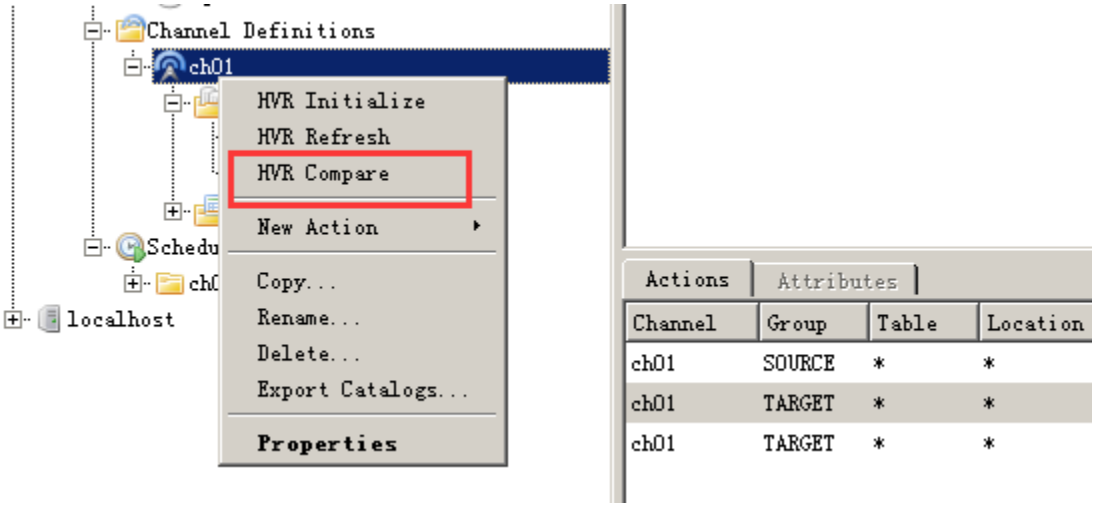
```
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| departments     |
| employees       |
| hvr_stbuch01_tgt |
| hvr_stinch01_tgt |
| hvr_stisch01_tgt |
| hvr_strrch01_tgt |
| hvr_strsch01_tgt |
+-----------------+
7 rows in set (0.05 sec)
```

与源端数据一致

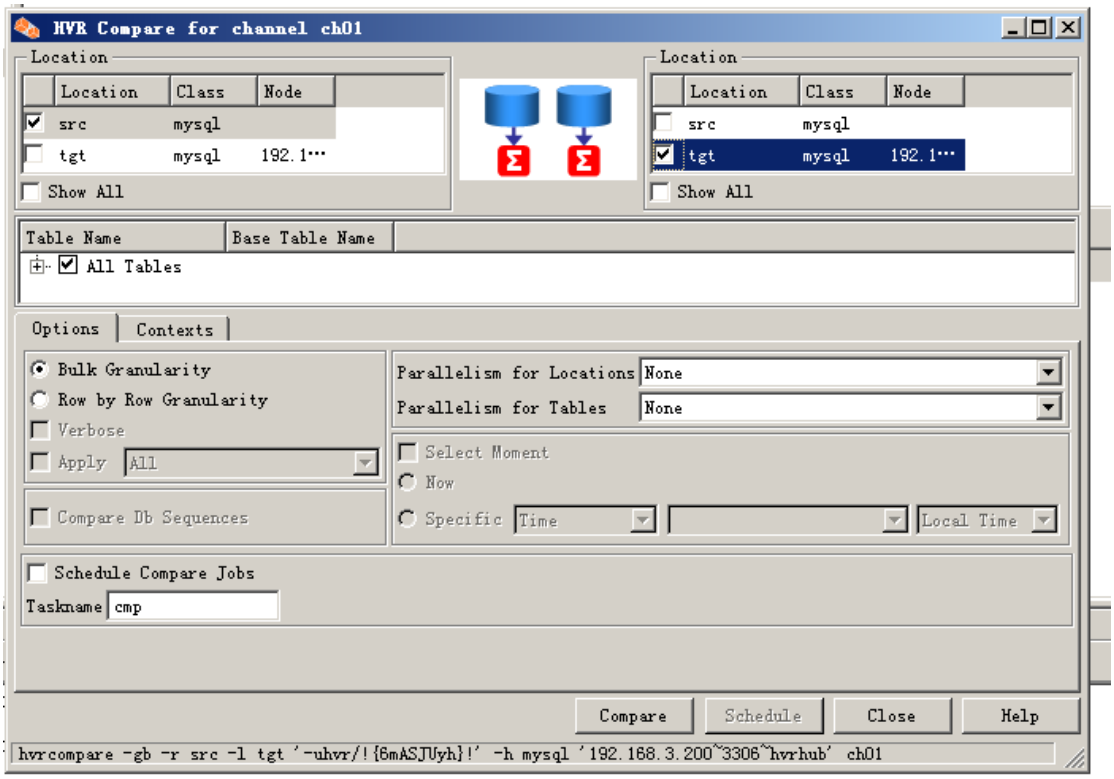
```
mysql> select * from departments;
+---------+-----------------+
| dept_no | dept_name       |
+---------+-----------------+
| d001    | Human Resources |
| d002    | Finance         |
+---------+-----------------+
2 rows in set (0.04 sec)

mysql> select * from employees;
+--------+------------+------------+-----------+------------+
| emp_no | birth_date | first_name | last_name | hire_date  |
+--------+------------+------------+-----------+------------+
|      1 | 1956-02-14 | tom        | zhang     | 1995-01-04 |
|      2 | 1976-03-04 | jerry      | yao       | 2005-12-04 |
|      3 | 1996-03-04 | cati       | huang     | 2008-11-24 |
+--------+------------+------------+-----------+------------+
3 rows in set (0.06 sec)
```

十一、使用 hvr compare 验证数据

在数据量大的情况下，一条条去验证数据显然不现实，可以使用此命令来验证两端数据是否一致



选择需要对比的数据库与表



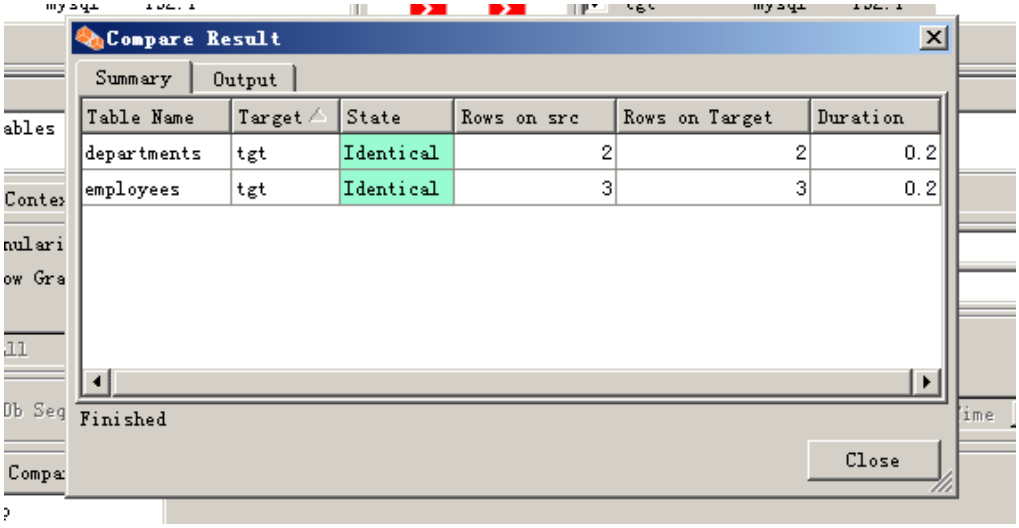
对比选项

Bulk Granularity　全表比对，适合数据差异较大的数据，只比对数据总量。

Row by Row Granularity 逐行比对，适合数据差异较少的数据，会有差异的明细（insert、update、delete 明细差异）

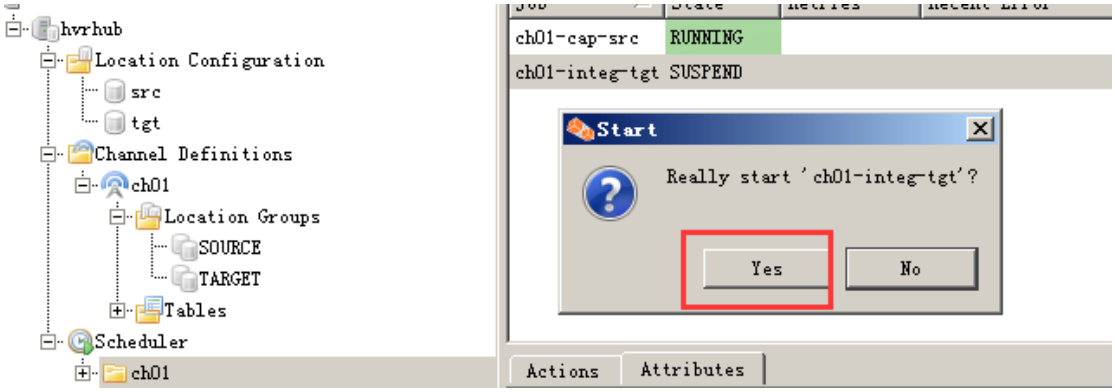
Parallellism 并行数选择的越多运行越快，但会占用服务器资源从而影响到业务性能

点击 compare 开始比对，结果如下图

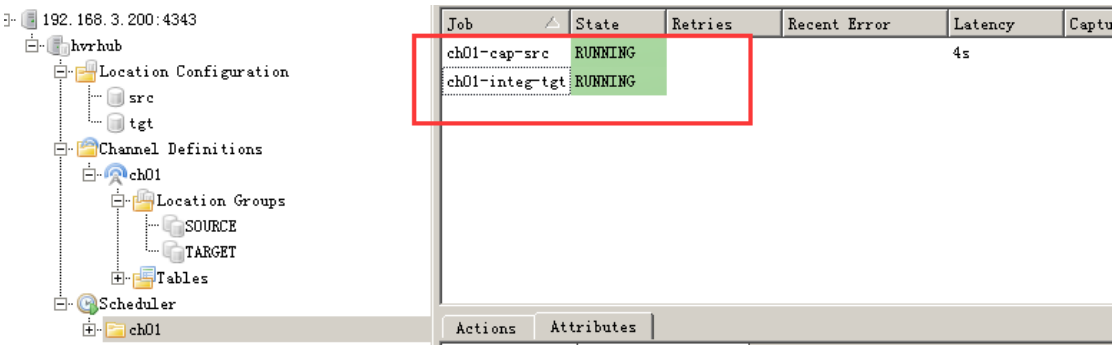


十二、启用实时复制

点击 ch01，在右侧项目栏中启动 integrate



服务状态



在源数据库上进行增删改的操作

```
mysql> insert into employees values(4,19820716,'helln','tan',20030428);
Query OK, 1 row affected (0.05 sec)

mysql> delete from employees where emp_no=2;
Query OK, 1 row affected (0.06 sec)

mysql> update employees set birth_date=19660214 where emp_no=1;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees;
+--------+------------+------------+-----------+------------+
| emp_no | birth_date | first_name | last_name | hire_date  |
+--------+------------+------------+-----------+------------+
|      1 | 1966-02-14 | tom        | zhang     | 1995-01-04 |
|      3 | 1996-03-04 | cati       | huang     | 2008-11-24 |
|      4 | 1982-07-16 | helln      | tan       | 2003-04-28 |
+--------+------------+------------+-----------+------------+
3 rows in set (0.06 sec)
```

在目标端验证

```
mysql> select * from employees;
+--------+------------+------------+-----------+------------+
| emp_no | birth_date | first_name | last_name | hire_date  |
+--------+------------+------------+-----------+------------+
|      1 | 1966-02-14 | tom        | zhang     | 1995-01-04 |
|      3 | 1996-03-04 | cati       | huang     | 2008-11-24 |
|      4 | 1982-07-16 | helln      | tan       | 2003-04-28 |
+--------+------------+------------+-----------+------------+
```

两端完全一致

从 hvr 日志中可以看到相关捕获的记录

```
2018-08-15T16:51:56+08:00: ch01-cap-src: Scanned 1 transaction (0 bytes) from 0 seconds ago containing 1 row (1 ins) for 1 table in 27.15 seconds.
2018-08-15T16:51:56+08:00: ch01-cap-src: Routed 285 bytes (compression=30.1%) from 'src' into 1 location.
2018-08-15T16:51:56+08:00: ch01-cap-src: Capture cycle 3.
2018-08-15T16:52:25+08:00: ch01-cap-src: Scanned 1 transaction (0 bytes) from 0 seconds ago containing 1 row (1 del) for 1 table in 29.14 seconds.
2018-08-15T16:52:26+08:00: ch01-cap-src: Routed 284 bytes (compression=30.4%) from 'src' into 1 location.
2018-08-15T16:52:26+08:00: ch01-cap-src: Capture cycle 4.
2018-08-15T16:53:04+08:00: ch01-cap-src: Scanned 1 transaction (0 bytes) from 0 seconds ago containing 2 rows (1 upd) for 1 table in 8.54 seconds.
```