kettle介绍和使用-结合日志平台

笔记本：	ElasticSearch		
创建时间：	2020/5/27 9:11	**更新时间：**	2020/6/4 15:31
作者：	jin.zhou@definesys.com		

kettle介绍

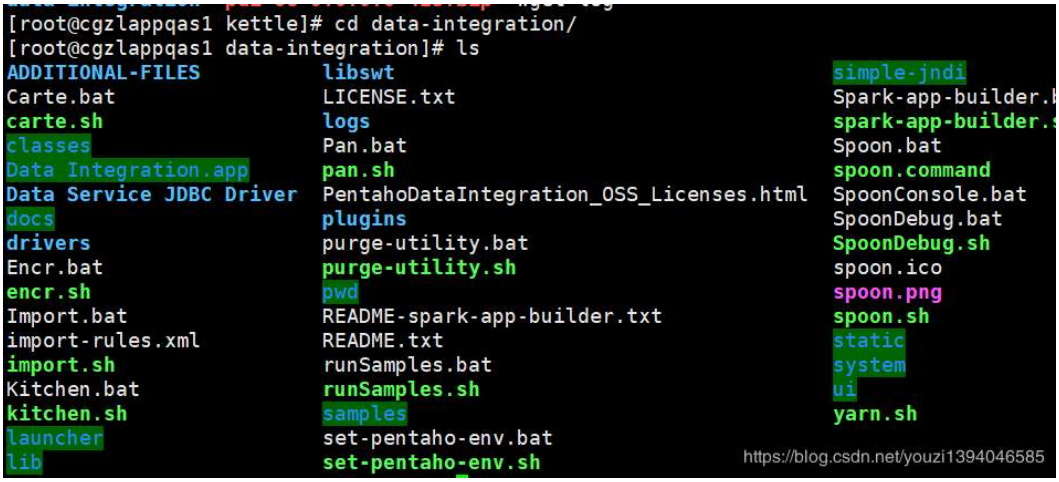
> Kettle 是一款国外开源的 ETL 工具，纯 Java 编写，绿色无需安装，数据抽取高效稳定(数据迁移工具)。Kettle
> 中有两种脚本文件，transformation 和 job，transformation 完成针对数据的基础转换，job
> 则完成整个工作流的控制。 Kettle 中文名称叫水壶，该项目的主程序员MATT希望把各种数据放到一个壶里，然后以一种指定的格式流出。
> Kettle这个ETL工具集，它允许你管理来自不同数据库的数据，通过提供一个图形化的用户环境来描述你想做什么，而不是你想怎么做。
> Kettle家族目前包括4个产品：Spoon、Pan、CHEF、Kitchen。
> **SPOON** 允许你通过图形界面来设计ETL转换过程（Transformation）。
> **PAN** 允许你批量运行由Spoon设计的ETL转换 (例如使用一个时间调度器)。Pan是一个后台执行的程序，没有图形界面。
> **CHEF** 允许你创建任务（Job）。 任务通过允许每个转换，任务，脚本等等，更有利于自动化更新数据仓库的复杂工作。任务通过允许每个转换，任务，脚本等等。任务将会被检查，看看是否正确地运行了。
>
> **KITCHEN** 允许你批量使用由Chef设计的任务 (例如使用一个时间调度器)。KITCHEN也是一个后台运行的程序。

kettle安装

进入下面链接下载：

wget https://nchc.dl.sourceforge.net/project/pentaho/Pentaho%209.0/client-tools/pdi-ce-9.0.0.0-423.zip
大小约为1.5个G，需要搭梯子加速

下载完成之后直接解压就可以使用，不用安装

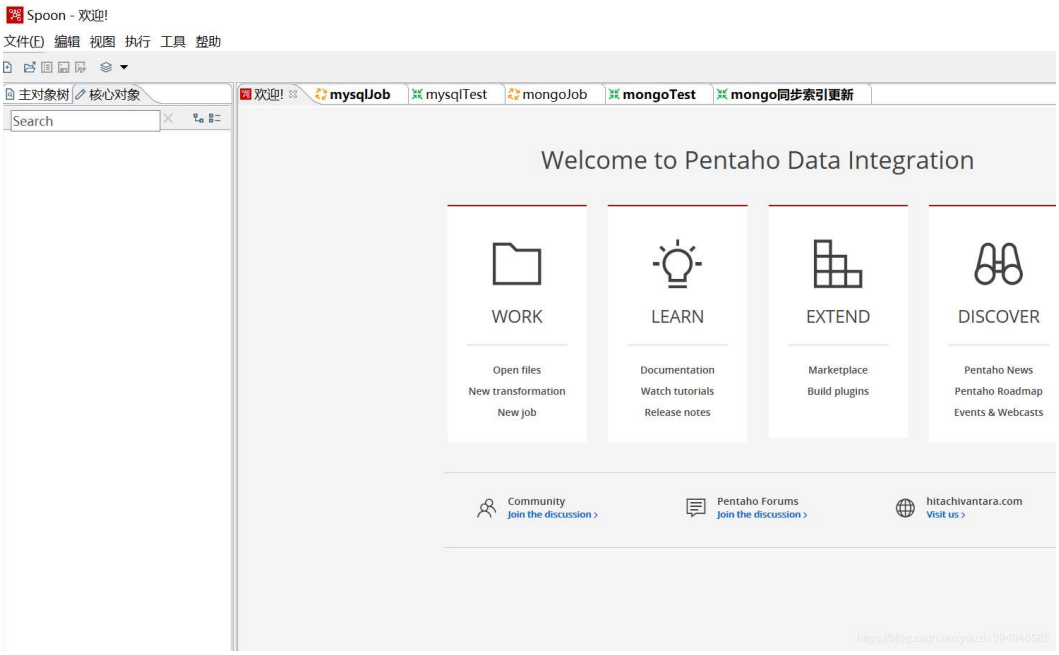


kettle使用

由于kettle是纯java编写，没有web界面，自带了spoon为gui界面，Linux环境下需要安装桌面才能使用图形化界面，建议在Windows下面使用spoon，然后把生成的文件放到服务器上面执行

使用kettle同步关系型数据库数据（MySQL示例）

安装好了kettle之后再Windows下我们执行spoon.bat ，然后等待一段时间，会出现如下界面

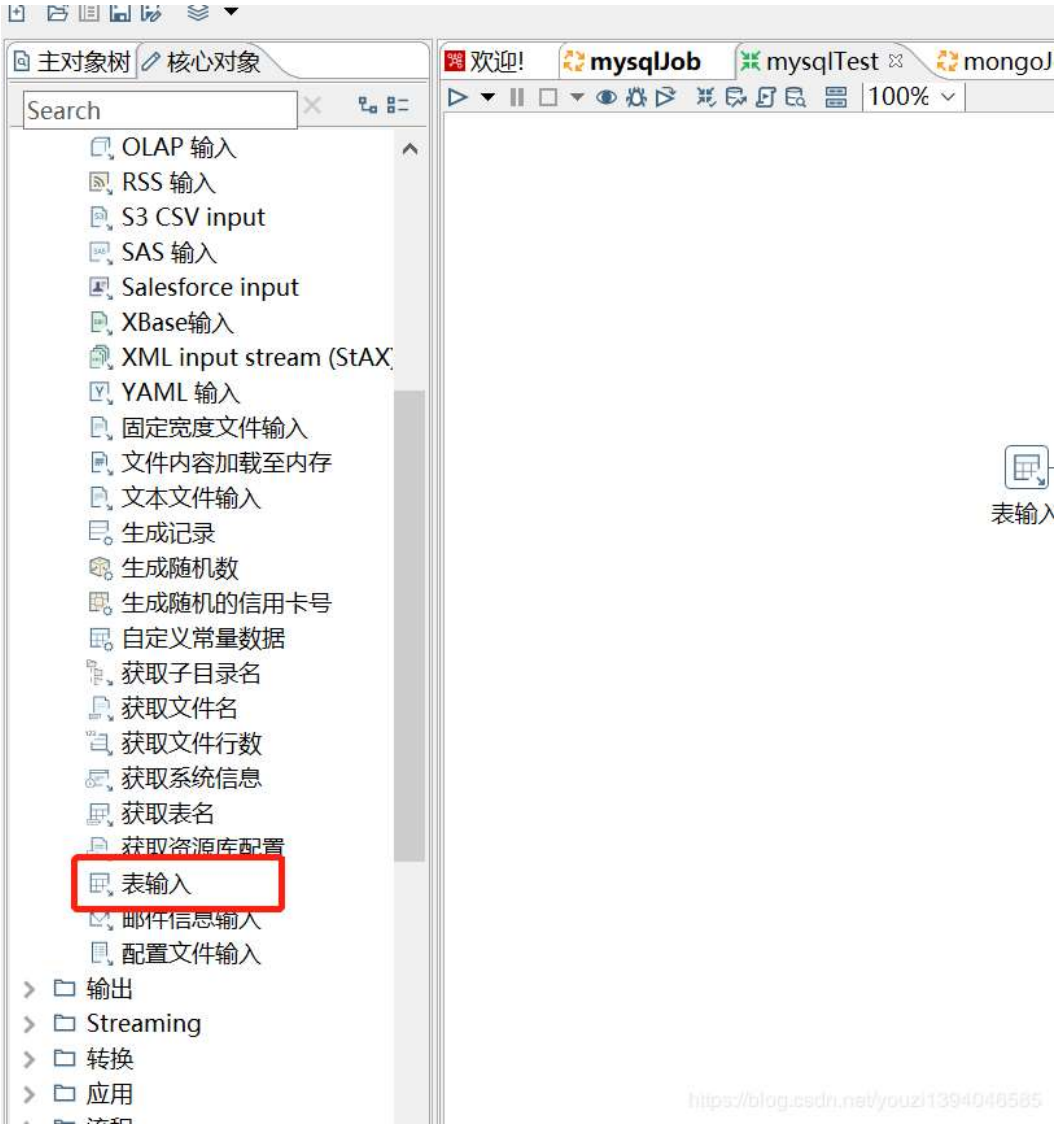


1. 创建一个转换

在欢迎界面选择菜单栏 文件--->新建---> 转换

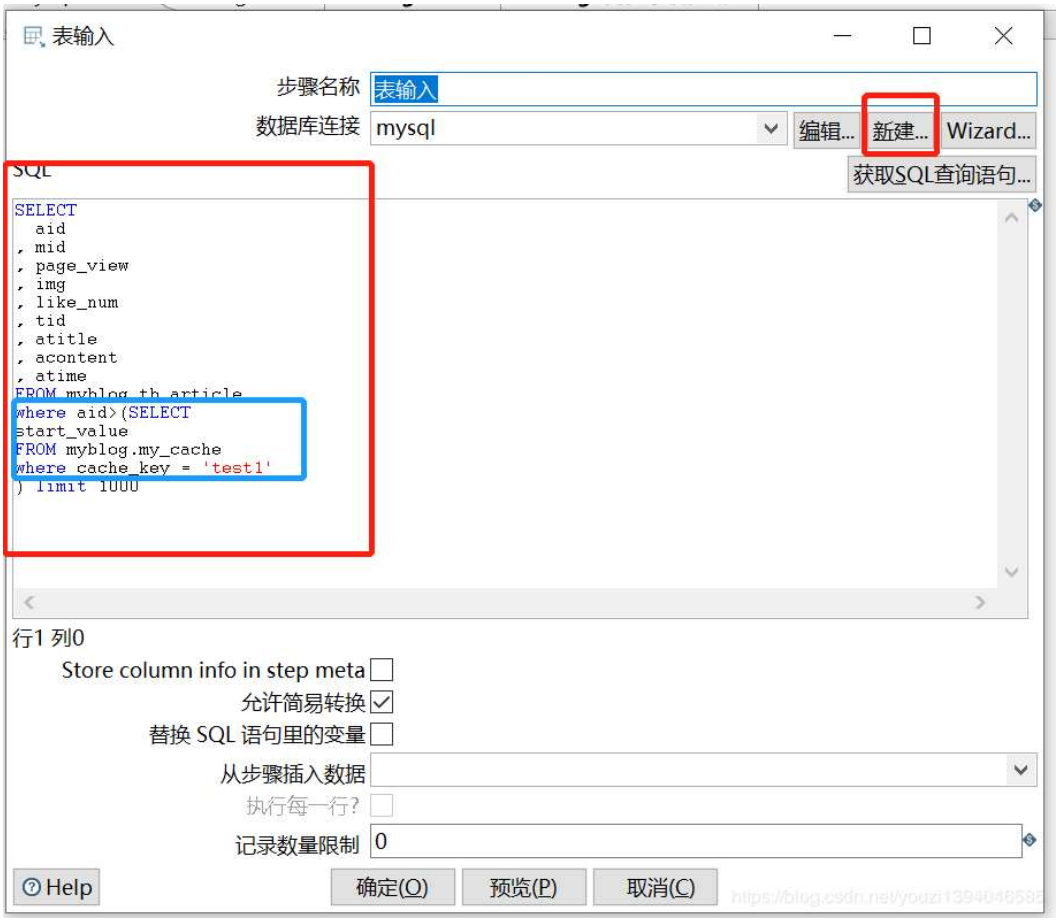
2. 选择表输入

在这之前你也许应该要先下载一个MySQL的数据库连接驱动，然后放到安装目录的lib目录下面，其他数据库同理。
接下来在左边的核心对象里面选择表输入，拖到右侧面板，如图：



双击表输入弹出如下界面。
在下面的界面里面首先需要新建一个数据库连接，创建完成之后再编写SQL语句。

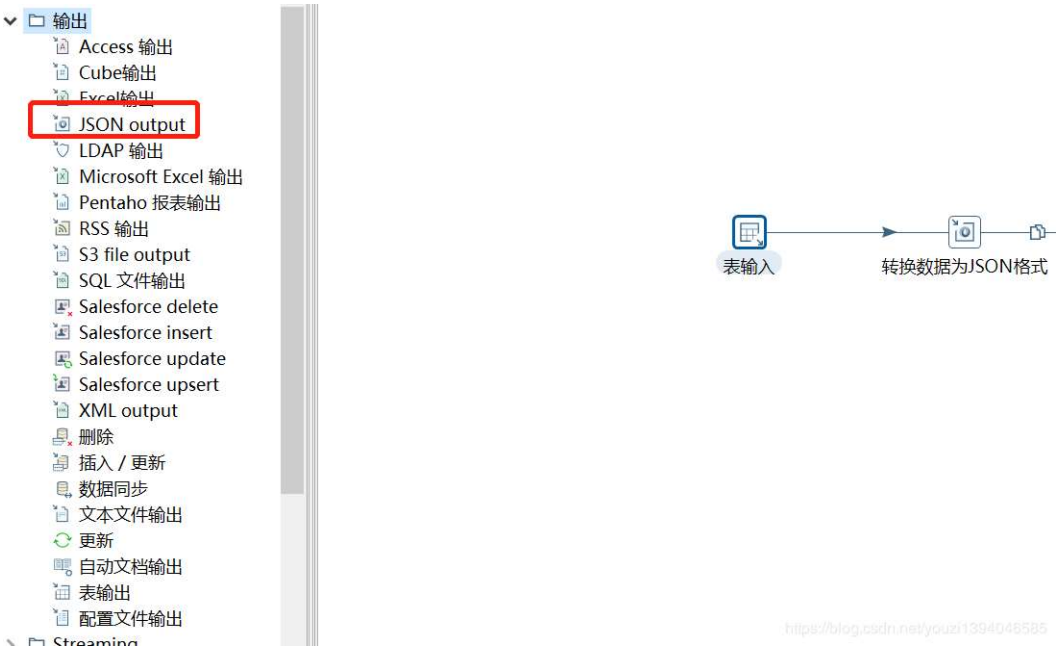
在下图中蓝色方框里面是一个记录表，记录了当前同步的记录的ID



3. 格式转换

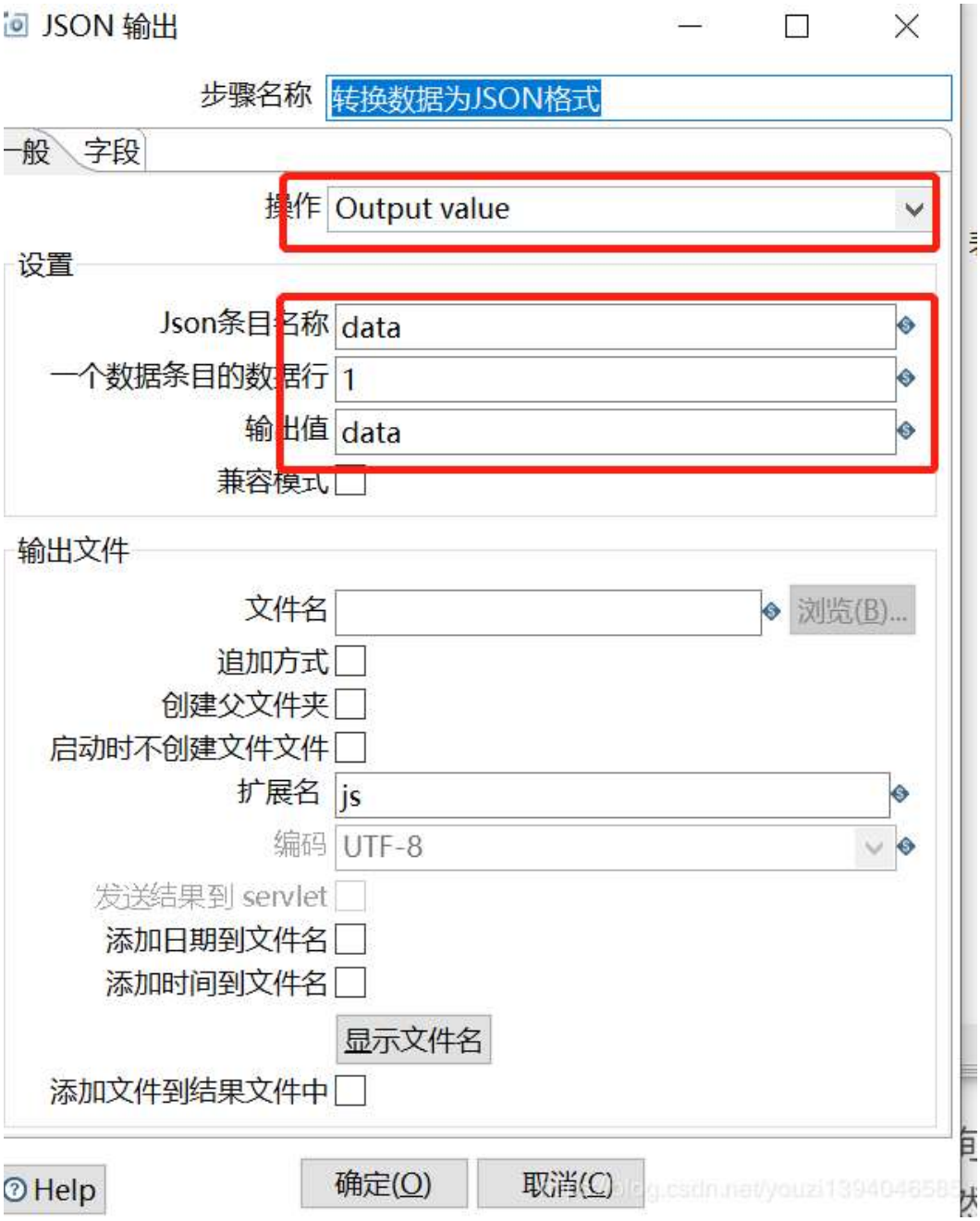
再上一个步骤里面创建好了转换之后我们需要把查询结果转换为JSON格式（可选）。

同理，我们再左侧输出里面选中一个JSON output 然后放到右边面板



接着双击刚刚创建的JSON输出，操作选择 output value，把结果输出到下一个步骤，同时，JSON的字段名为data，其输出格式如图所示

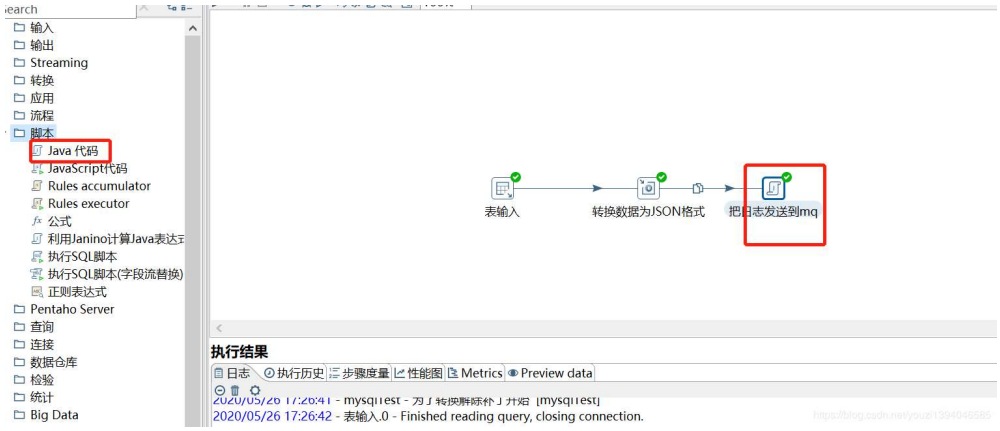
- {"data":[{"like_num":1,"acontent":"测试","img":null,"atime":"2020\/03\/04 10:15:35.000000



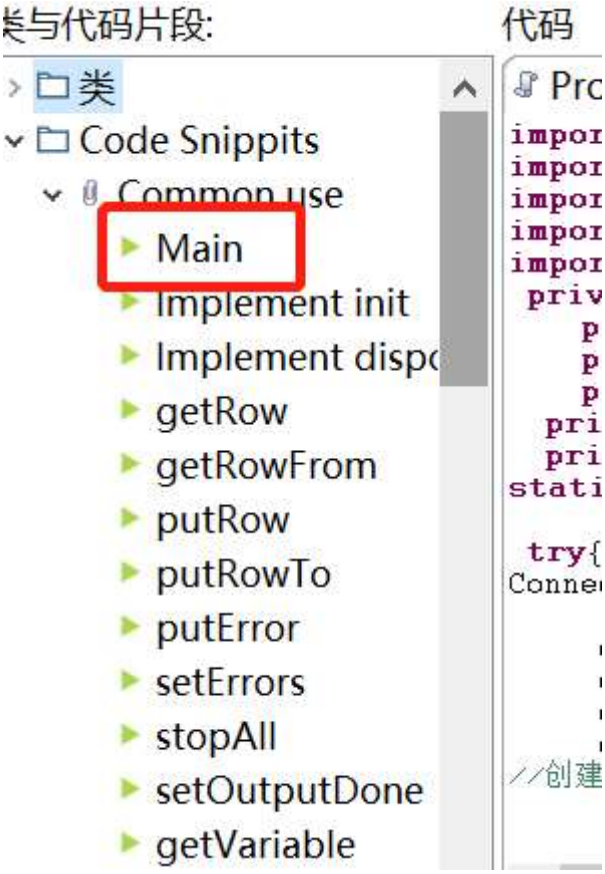
4. 执行脚本

kettle为我们提供了多种脚本的支持，这里我们使用java脚本，把刚刚查询的数据发送到mq。

- 创建脚本
  在左侧的核心对象选择脚本--->java脚本，拖放到右侧面板



- 选择代码模板
  双击我们刚刚创建的脚本，在弹出的选择框里面我们可以快速生成一个代码模板，如图：



- 编写代码
  在选择了代码模板之后，我们开始编写脚本代码，不过在这之前需要先把脚本运行所需要的依赖放到安装目录下面的lib目录下，例如我的脚本就需要kafka的客户端驱动。

```
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;
    private static final KafkaProducer<String, String> producer;
    //需要到日志管理平台申请一个主题和分区
        private static final String TOPIC="testLogTopic";
```

```
    //需要到日志管理平台申请一个主题和分区
        private static final int PARTITION=1;
        private static final String
SERVER_PATH="172.16.161.51:9002,172.16.161.51:9003,172.16.161.51:9004";
static{
        /**
        * 一般这里的属性不用更改
        */
                Properties props = new Properties();
        props.put("bootstrap.servers", SERVER_PATH);//xxx服务器ip
        props.put("acks", "all");//所有follower都响应了才认为消息提交成
功，即"committed"
        props.put("retries", 0);//retries = MAX 无限重试，直到你意识到出
现了问题:)
        props.put("batch.size", 16384);//producer将试图批处理消息记录，
以减少请求次数.默认的批量处理消息字节数
        //batch.size当批量的数据大小达到设定值后，就会立即发送，不顾下
面的linger.ms
        props.put("linger.ms", 1);//延迟1ms发送，这项设置将通过增加小的
延迟来完成--即，不是立即发送一条记录，producer将会等待给定的延迟时间以
允许其他消息记录发送，这些消息记录可以批量处理
        props.put("buffer.memory", 33554432);//producer可以用来缓存数据
的内存大小。
        props.put("key.serializer",

"org.apache.kafka.common.serialization.IntegerSerializer");
        props.put("value.serializer",

"org.apache.kafka.common.serialization.StringSerializer");
        producer = new KafkaProducer<String, String>(props);
}



public boolean processRow(StepMetaInterface smi, StepDataInterface sdi)
throws KettleException {
  Object[] r = getRow();
  if (r == null) {
    setOutputDone();
    return false;
  }
  // It is always safest to call createOutputRow() to ensure that your
output row's Object[] is large
  // enough to handle any new fields you are creating in this step.
  r = createOutputRow(r, data.outputRowMeta.size());
  String res = get(Fields.In, "data").getString(r);
try{
        //String val = new String(res.getBytes(),"utf-8");
```

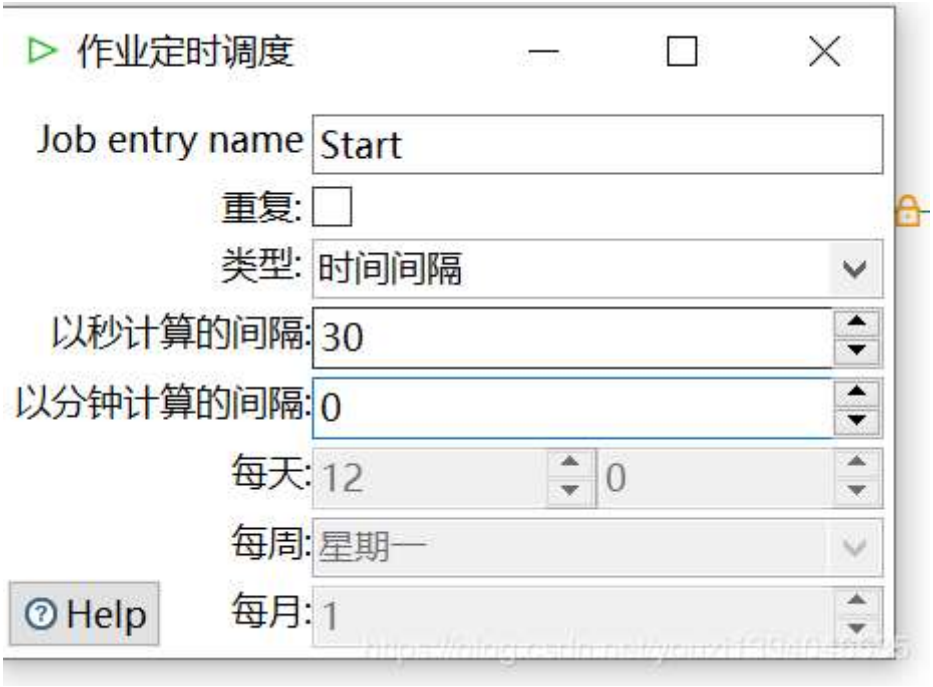
```
            //发送消息
            producer.send(new ProducerRecord<String, String>
    (TOPIC,PARTITION,null, res));
            //logBasic(val);
    }catch(Exception e){
    logError(e.getMessage(), e);

    }

      // Send the row on to the next step.
      putRow(data.outputRowMeta, r);

      return true;
    }
```

5. 创建job

在转换创建完成之后需要创建一个job用来运行我们创建的转换
菜单栏--->文件--->新建--->作业
如图，创建好job之后再左侧的核心对象里面选择start，转换，成功等组件拖放到右侧面板

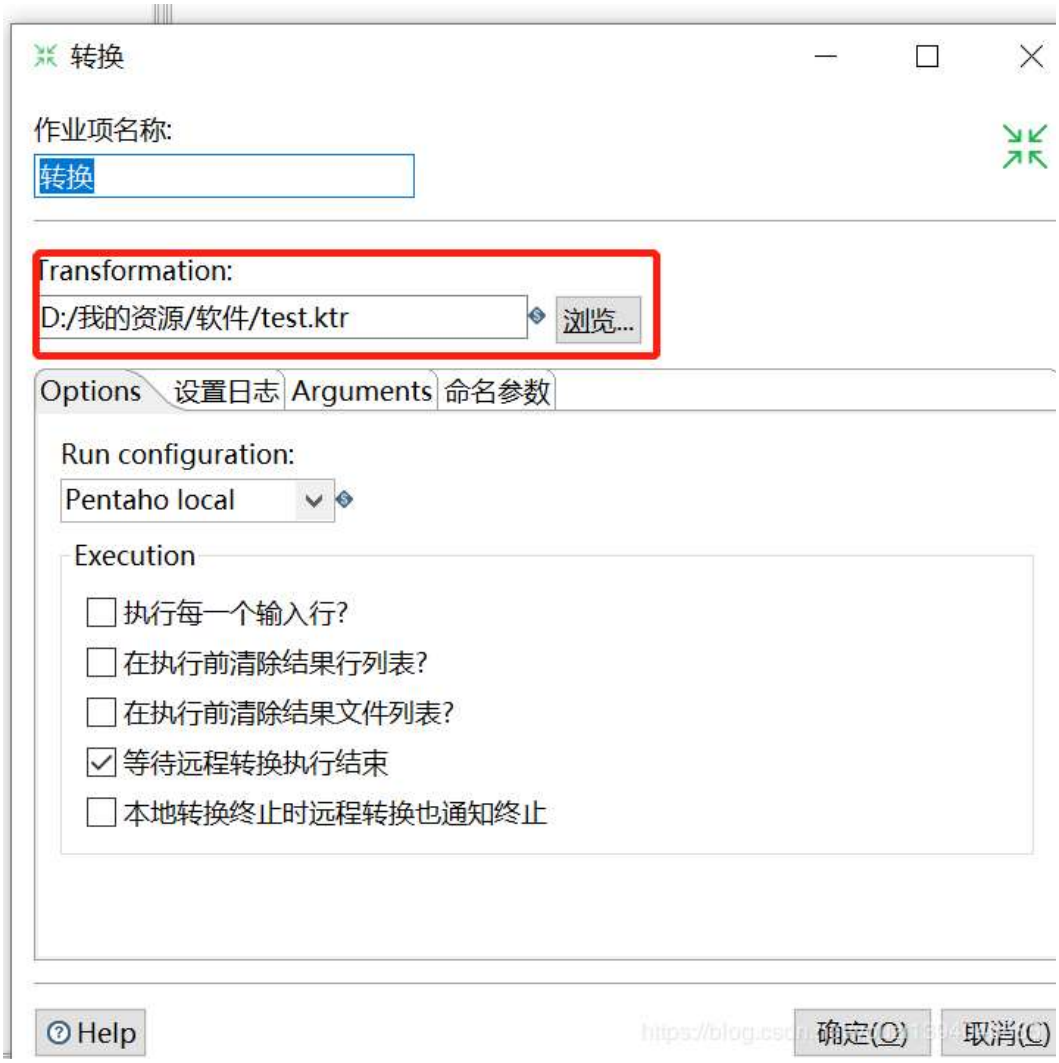
在这个job中：

start
start标记着本次任务的开始，双击start可以设置该job的执行方案，比如重复执行等等



转换

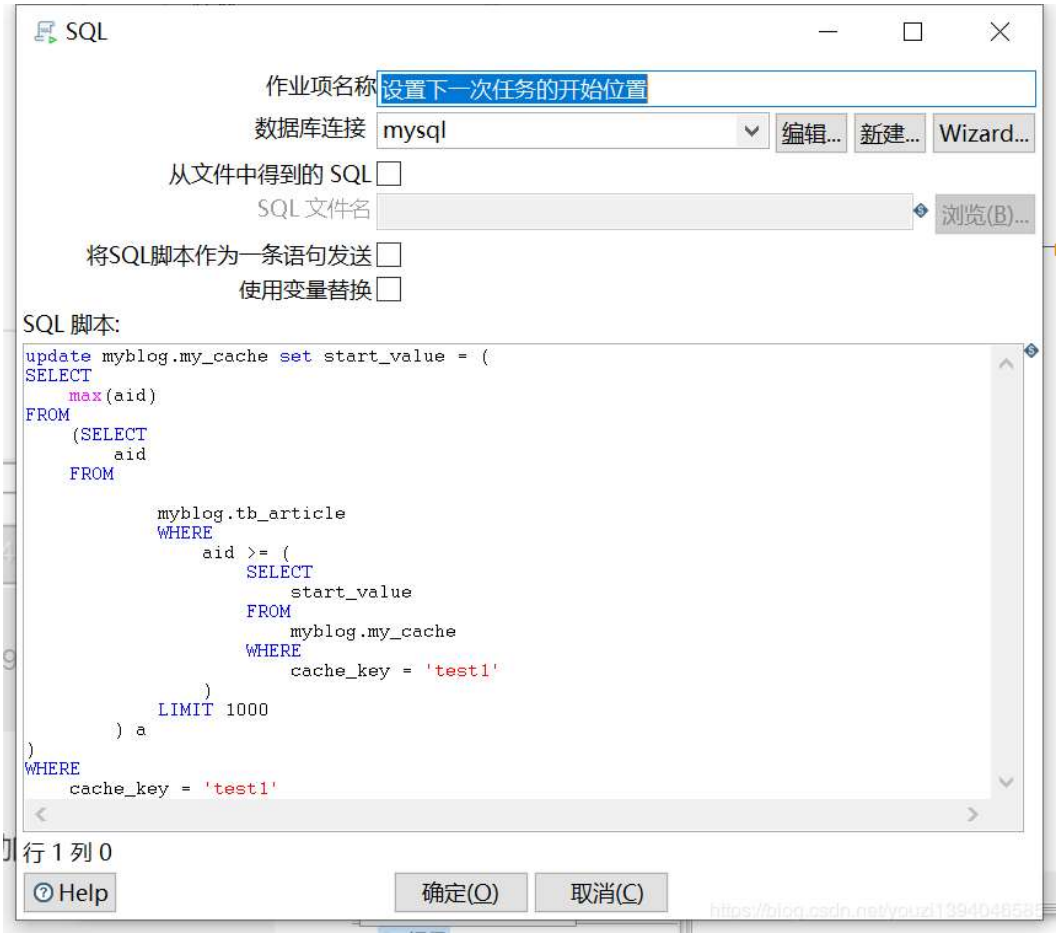
双击转换可以选择我们在之前创建的转换文件



脚本

转换步骤完成之后我们需要更新一下下一次开始同步的时候需要的ID的值，所以我们

可以添加一个脚本来执行更新操作



使用kettle同步NoSql数据（MongoDB示例）

这里创建作业和转换的步骤和前面大同小异，就不详细介绍了，主要说明几点区别

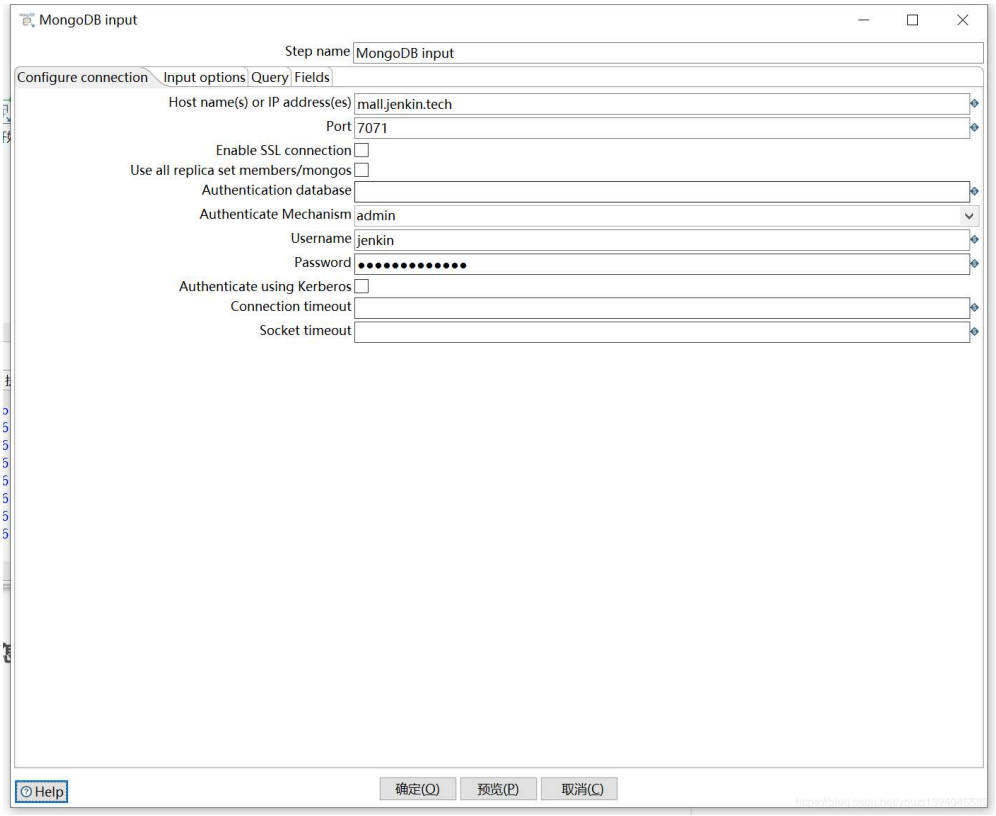
创建转换时的注意事项

由于kettle里面转换时并发执行的，在编写转换的时候需要注意有些步骤坑需要阻塞，如图：
在下面的转换里面第一步查询到了一个开始ID之后然后把结果设置到一个变量里面，然后在MongoDBinput里面就可以使用刚刚设置的变量，但是在这三个步骤里面由于是并行执行的，所以如果不添加阻塞步骤，可能会导致MongoDBinput这个步骤获取

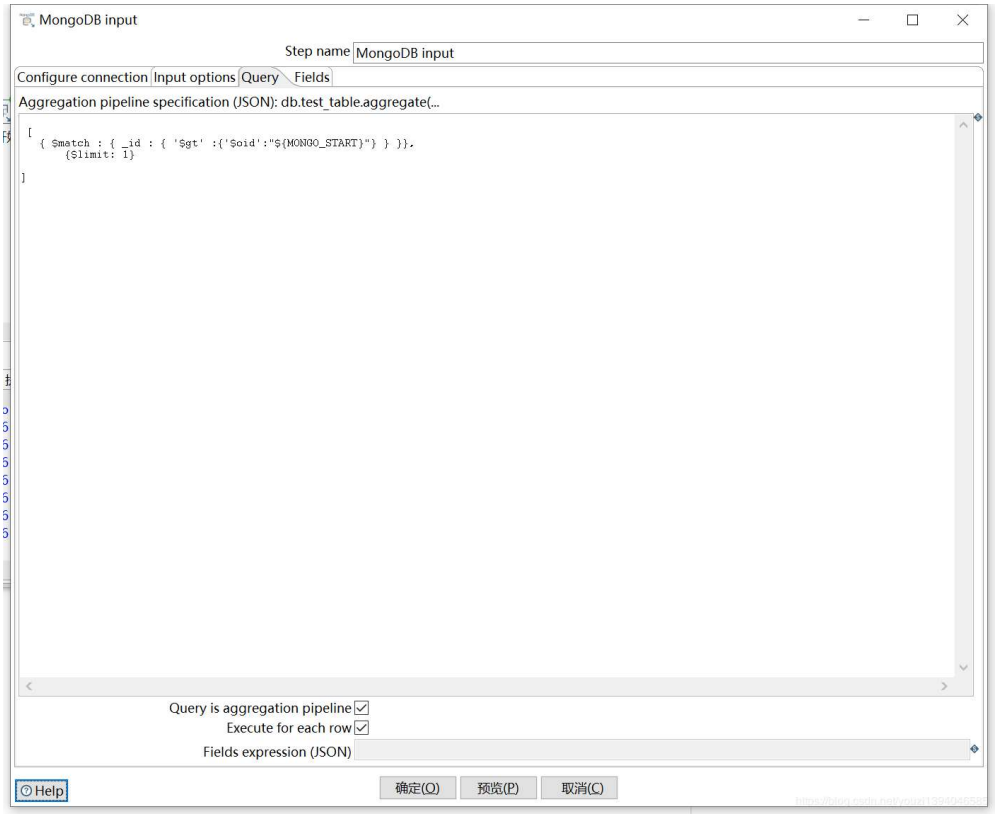
不到最新的变量

MongoDBInput怎么创建

1. 连接创建

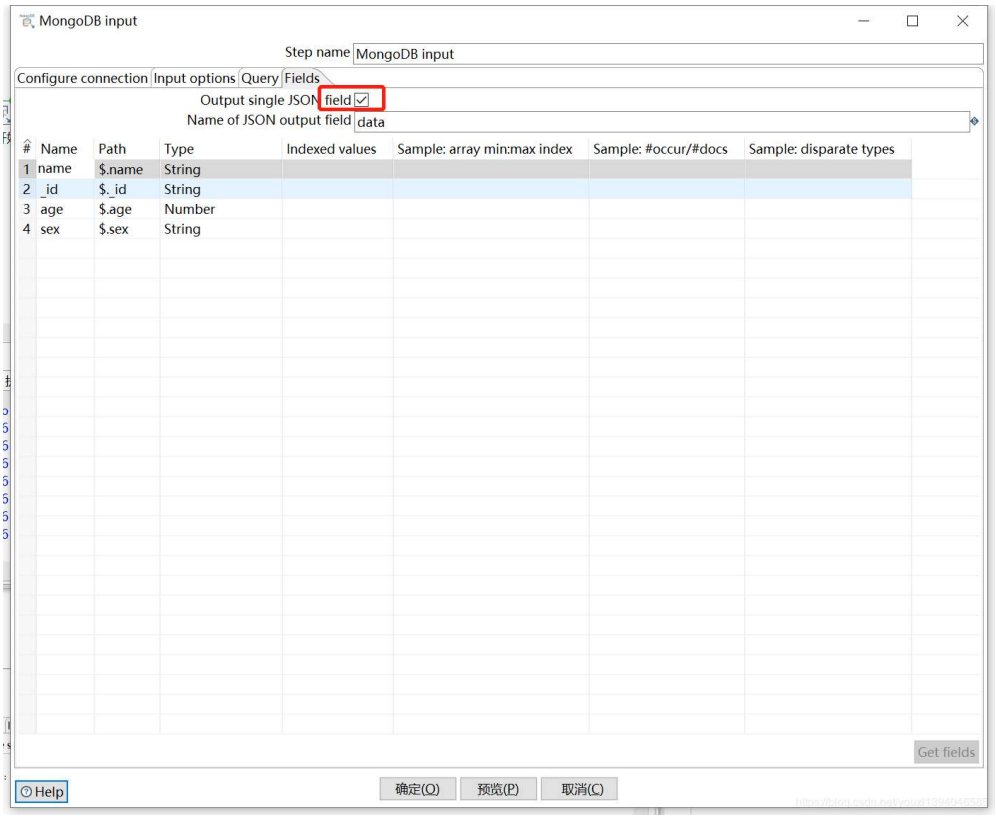


2. 查询编写



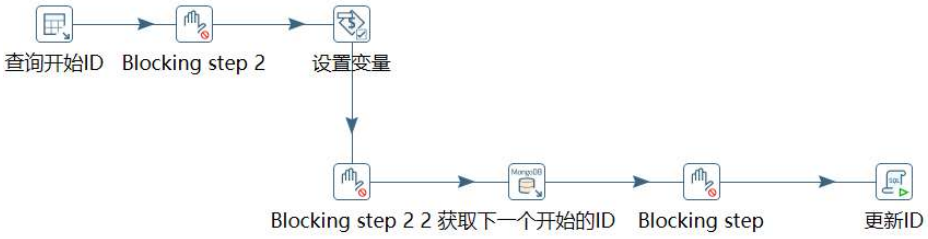
3. 字段映射

下图中的复选框标识是否将结果以一个字段返回，如果是的话，结果将会是一个名为data的JSON字符串



job的创建

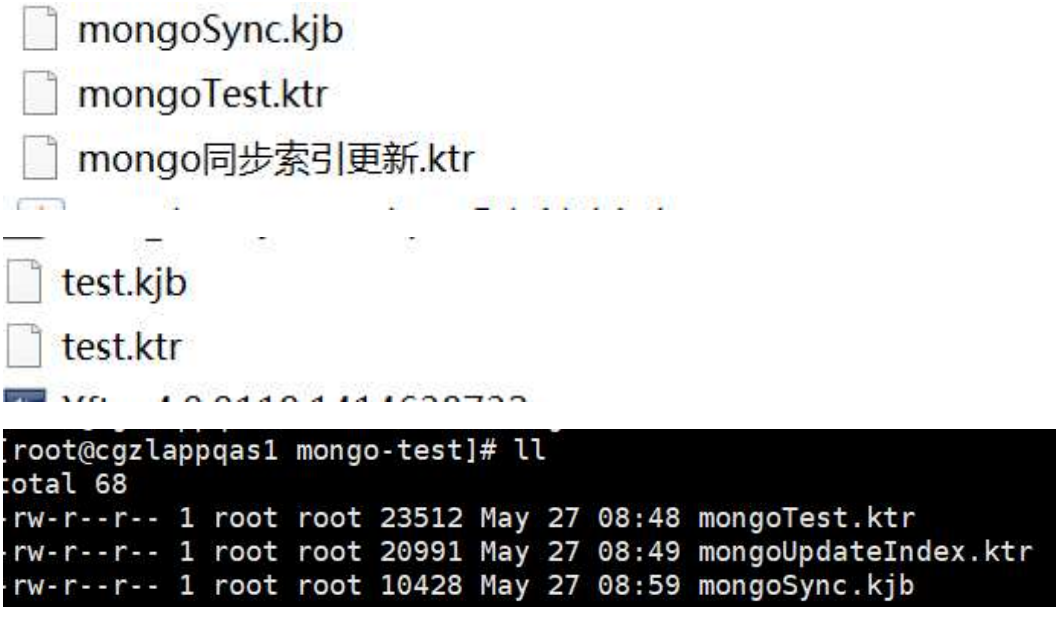
在job里面有两个转换，一个是同步数据的转换，另一个是更新同步位置的转换，如下图所示，这个步骤是为了在一次任务同步完成之后更新下一次任务开始时的ID



如何在Linux上面运行已经创建好的job?

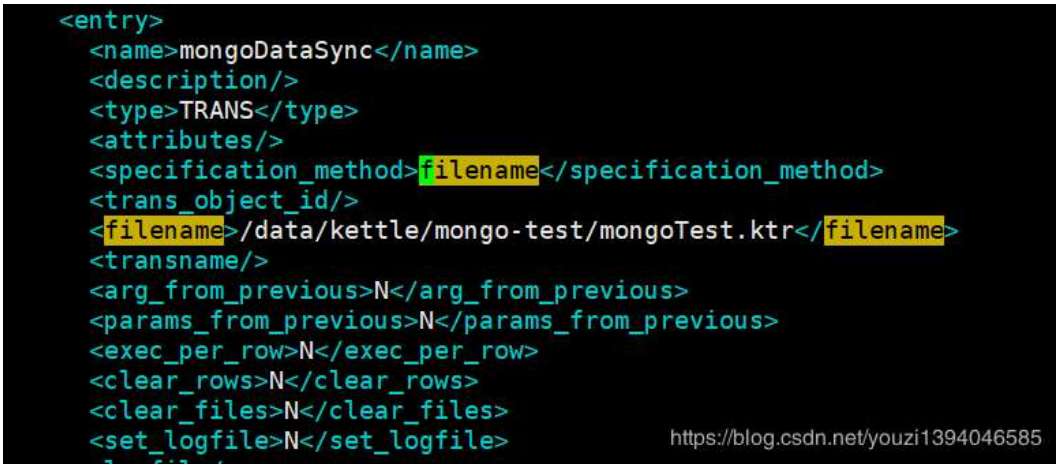
在这之前需要把我们用到的依赖jar包先放到Linux机器上面的kettle安装目录下面的lib目录里面去，比如MySQL的驱动等等。

1、把创建好的job传输到Linux机器上面



2、编辑kjb文件

编辑我们创建好的kjb文件，由于job文件中的转换路径发生了变化，所以需要修改为正确的路径
如图，搜索filename关键字可以找到对应的需要修改的位置，把路径修改为目前Linux里面的路径就可以了。



3、执行任务

执行下面的代码就可以运行我们创建好的job了

```
../data-integration/kitchen.sh -file=mongoSync.kjb
```