# [HBase-Elasticsearch数据同步](https://esdoc.bbossgroups.com/" \l "/hbase-elasticsearch?id=hbase-elasticsearch%e6%95%b0%e6%8d%ae%e5%90%8c%e6%ad%a5)

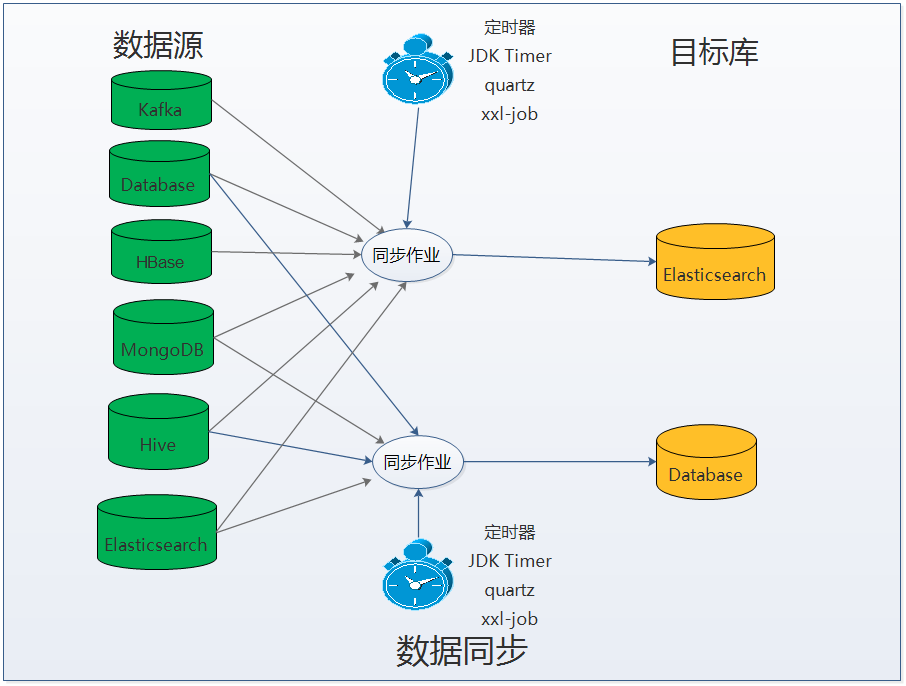
本文介绍HBase-Elasticsearch数据同步功能，HBase-Elasticsearch数据同步案例源码地址：

[**https://github.com/bbossgroups/hbase-elasticsearch**](https://github.com/bbossgroups/hbase-elasticsearch)

# 前言

Bboss is a good elasticsearch Java highlevel rest client api. It operates and accesses elasticsearch like mybatis to db.

bboss提供了不错的数据同步功能，通过bboss可以非常方便地实现各种大数据同步功能：



与logstash类似，bboss数据同步主要功能特点：

1. 支持多种类型数据源数据同步功能
   * 将数据库表数据同步到Elasticsearch
   * 将数据库表数据同步到数据库表
   * 将Elasticsearch数据同步到数据库表
   * 将Elasticsearch数据同步到Elasticsearch
   * 将mongodb数据同步到Elasticsearch
   * 将mongodb数据同步到数据库表
   * 将hbase数据同步到Elasticsearch
   * 从kafka接收数据导入elasticsearch（支持kafka\_2.12-0.10.2.0和kafka\_2.12-2.3.0 系列版本）
2. 支持的导入方式
   * 逐条数据导入
   * 批量数据导入
   * 批量数据多线程并行导入
   * 定时全量（串行/并行）数据导入
   * 定时增量（串行/并行）数据导入
3. 支持的数据库： mysql,maridb，postgress,oracle ,sqlserver,db2,tidb,hive，mongodb等
4. 支持的Elasticsearch版本： 1.x,2.x,5.x,6.x,7.x,+
5. 支持将ip转换为对应的运营商和城市地理位置信息
6. 支持多种定时任务执行引擎：
   * jdk timer （内置）
   * quartz
   * xxl-job分布式调度引擎，基于分片调度机制实现海量数据快速同步能力

# [环境要求](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_1%e7%8e%af%e5%a2%83%e8%a6%81%e6%b1%82)

## 2.1 软件要求

JDK requirement: JDK 1.7+

Elasticsearch version requirements: 1.x,2.X,5.X,6.X,7.x,+

Spring booter 1.x,2.x,+

hbase 1.x,hbase 2.x

bboss 6.0.1

## [2.2 maven坐标](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_6%e5%af%bc%e5%85%a5maven%e5%9d%90%e6%a0%87)

<dependency>

<groupId>com.bbossgroups.plugins</groupId>

<artifactId>bboss-elasticsearch-rest-hbase</artifactId>

<version>6.0.1</version>

<scope>compile</scope>

</dependency>

<!--

hbase shaded client的版本号与hbase的版本相关，请根据hbase的版本调整hbase shaded client的版本号

-->

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-shaded-client</artifactId>

<version>2.2.3</version>

</dependency>

除了**[org.frameworkset.elasticsearch.imp.HBase2ESScrollTimestampDemo223](https://github.com/bbossgroups/hbase-elasticsearch/blob/master/src/main/java/org/frameworkset/elasticsearch/imp/HBase2ESScrollTimestampDemo223.java" \t "_blank)**基于hbase 2.2.3开发，其他案例都基于hbase 1.3.0版本开发，所以选择的是1.2.4的客户端，具体的client版本号可以根据hbase版本自行选择：

[**https://search.maven.org/artifact/org.apache.hbase/hbase-shaded-client**](https://search.maven.org/artifact/org.apache.hbase/hbase-shaded-client)

compile([group: 'org.apache.hbase', name: 'hbase-shaded-client', version: "1.2.4", transitive: true])

# [3.HBase-Elasticsearch 数据同步工具demo](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_2hbase-elasticsearch-%e6%95%b0%e6%8d%ae%e5%90%8c%e6%ad%a5%e5%b7%a5%e5%85%b7demo)

使用本demo所带的应用程序运行容器环境，可以快速编写，打包发布可运行的数据导入工具，包含现成的示例如下：

## [3.1 jdk timer定时全量同步](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_21-jdk-timer%e5%ae%9a%e6%97%b6%e5%85%a8%e9%87%8f%e5%90%8c%e6%ad%a5)

[**org.frameworkset.elasticsearch.imp.HBase2ESFullDemo**](https://github.com/bbossgroups/hbase-elasticsearch/blob/master/src/main/java/org/frameworkset/elasticsearch/imp/HBase2ESFullDemo.java)

## [3.2 jdk timer定时增量同步](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_22-jdk-timer%e5%ae%9a%e6%97%b6%e5%a2%9e%e9%87%8f%e5%90%8c%e6%ad%a5)

[**org.frameworkset.elasticsearch.imp.HBase2ESScrollTimestampDemo**](https://github.com/bbossgroups/hbase-elasticsearch/blob/master/src/main/java/org/frameworkset/elasticsearch/imp/HBase2ESScrollTimestampDemo.java)

## [3.3 jdk timer定时增量同步（简化demo，hbase1.x,hbase2.x都可以跑）](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_23-jdk-timer%e5%ae%9a%e6%97%b6%e5%a2%9e%e9%87%8f%e5%90%8c%e6%ad%a5%ef%bc%88%e7%ae%80%e5%8c%96demo%ef%bc%8chbase1xhbase2x%e9%83%bd%e5%8f%af%e4%bb%a5%e8%b7%91%ef%bc%89)

[**org.frameworkset.elasticsearch.imp.HBase2ESScrollTimestampDemo223**](https://github.com/bbossgroups/hbase-elasticsearch/blob/master/src/main/java/org/frameworkset/elasticsearch/imp/HBase2ESScrollTimestampDemo223.java)

## [3.4 jdk timer定时带条件同步](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_24-jdk-timer%e5%ae%9a%e6%97%b6%e5%b8%a6%e6%9d%a1%e4%bb%b6%e5%90%8c%e6%ad%a5)

[**org.frameworkset.elasticsearch.imp.HBase2ESFullDemoWithFilter**](https://github.com/bbossgroups/hbase-elasticsearch/blob/master/src/main/java/org/frameworkset/elasticsearch/imp/HBase2ESFullDemoWithFilter.java)

## [3.5 quartz定时全量同步](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_25-quartz%e5%ae%9a%e6%97%b6%e5%85%a8%e9%87%8f%e5%90%8c%e6%ad%a5)

[**org.frameworkset.elasticsearch.imp.QuartzHBase2ESImportTask**](https://github.com/bbossgroups/hbase-elasticsearch/blob/master/src/main/java/org/frameworkset/elasticsearch/imp/QuartzHBase2ESImportTask.java)

## 3.6 [本案例支持的hbase版本](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_3%e6%94%af%e6%8c%81%e7%9a%84%e6%95%b0%e6%8d%ae%e5%ba%93%ef%bc%9a)

HBase 1.x,2.x到elasticsearch数据同步

## 3.7 本案例[支持的Elasticsearch版本](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_4%e6%94%af%e6%8c%81%e7%9a%84elasticsearch%e7%89%88%e6%9c%ac)

Elasticsearch 1.x,2.x,5.x,6.x,7.x,+

# [4.hbase-elasticsearch同步作业代码和配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_7hbase-elasticsearch%e5%90%8c%e6%ad%a5%e4%bd%9c%e4%b8%9a%e4%bb%a3%e7%a0%81%e5%92%8c%e9%85%8d%e7%bd%ae)

以示例HBase2ESScrollTimestampDemo223作业进行讲解。

## [4.1 hbase参数配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_71-hbase%e5%8f%82%e6%95%b0%e9%85%8d%e7%bd%ae)

HBaseExportBuilder importBuilder = new HBaseExportBuilder();

importBuilder.setBatchSize(1000) //设置批量写入目标Elasticsearch记录数

.setFetchSize(10000); //设置批量从源Hbase中拉取的记录数,HBase-0.98 默认值为为 100，HBase-1.2 默认值为 2147483647，即 Integer.MAX\_VALUE。Scan.next() 的一次 RPC 请求 fetch 的记录条数。配置建议：这个参数与下面的setMaxResultSize配合使用，在网络状况良好的情况下，自定义设置不宜太小， 可以直接采用默认值，不配置。

// importBuilder.setHbaseBatch(100) //配置获取的列数，假如表有两个列簇 cf，info，每个列簇5个列。这样每行可能有10列了，setBatch() 可以控制每次获取的最大列数，进一步从列级别控制流量。配置建议：当列数很多，数据量大时考虑配置此参数，例如100列每次只获取50列。一般情况可以默认值（-1 不受限），如果设置了scan filter也不需要设置

// .setMaxResultSize(10000l);//客户端缓存的最大字节数，HBase-0.98 无该项配置，HBase-1.2 默认值为 210241024，即 2M。Scan.next() 的一次 RPC 请求 fetch 的数据量大小，目前 HBase-1.2 在 Caching 为默认值(Integer Max)的时候，实际使用这个参数控制 RPC 次数和流量。配置建议：如果网络状况较好（万兆网卡），scan 的数据量非常大，可以将这个值配置高一点。如果配置过高：则可能 loadCache 速度比较慢，导致 scan timeout 异常

// 参考文档：https://blog.csdn.net/kangkangwanwan/article/details/89332536

/\*\*

\* hbase参数配置

\*/

String hbaseZookeeperQuorum = CommonLauncher.getProperty("hbase.zookeeper.quorum","192.168.137.133");//同时指定了默认值

String hbaseZookeeperPort = CommonLauncher.getProperty("hbase.zookeeper.property.clientPort","2183");//同时指定了默认值

String zookeeperZnodeParent = CommonLauncher.getProperty("zookeeper.znode.parent","/hbase");//同时指定了默认值

importBuilder.addHbaseClientProperty("hbase.zookeeper.quorum",hbaseZookeeperQuorum) //hbase客户端连接参数设置，参数含义参考hbase官方客户端文档

.addHbaseClientProperty("hbase.zookeeper.property.clientPort",hbaseZookeeperPort)

.addHbaseClientProperty("zookeeper.znode.parent",zookeeperZnodeParent)

.addHbaseClientProperty("hbase.ipc.client.tcpnodelay","true")

.addHbaseClientProperty("hbase.rpc.timeout","10000")

.addHbaseClientProperty("hbase.client.operation.timeout","10000")

.addHbaseClientProperty("hbase.ipc.client.socket.timeout.read","20000")

.addHbaseClientProperty("hbase.ipc.client.socket.timeout.write","30000")

.setHbaseClientThreadCount(100) //hbase客户端连接线程池参数设置

.setHbaseClientThreadQueue(100)

.setHbaseClientKeepAliveTime(10000l)

.setHbaseClientBlockedWaitTimeout(10000l)

.setHbaseClientWarnMultsRejects(1000)

.setHbaseClientPreStartAllCoreThreads(true)

.setHbaseClientThreadDaemon(true)

.setHbaseTable("AgentInfo") //指定需要同步数据的hbase表名称

;

## [4.2 Elasticsearch参数配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_72-elasticsearch%e5%8f%82%e6%95%b0%e9%85%8d%e7%bd%ae)

/\*\*

\* es相关配置

可以通过addElasticsearchProperty方法添加Elasticsearch客户端配置，

也可以直接读取application.properties文件中设置的es配置,两种方式都可以，案例中采用application.properties的方式

\*/

//importBuilder.addElasticsearchProperty("elasticsearch.rest.hostNames","192.168.137.1:9200");//设置es服务器地址

importBuilder.setTargetElasticsearch("targetElasticsearch");//设置目标Elasticsearch集群数据源名称，在application.properties文件中配置

importBuilder.setIndex("hbase2esdemo") //全局设置要目标elasticsearch索引名称

.setIndexType("hbase2esdemo"); //全局设值目标elasticsearch索引类型名称，如果是Elasticsearch 7以后的版本不需要配置

更多Elasticsearch配置参数文档：**[Elasticsearch参数配置](https://esdoc.bbossgroups.com/" \l "/mongodb-elasticsearch?id=_5242-elasticsearch%e5%8f%82%e6%95%b0%e9%85%8d%e7%bd%ae" \t "_blank)**

## [4.3 Elasticsearch文档\_id生成机制配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_73-elasticsearch%e6%96%87%e6%a1%a3_id%e7%94%9f%e6%88%90%e6%9c%ba%e5%88%b6%e9%85%8d%e7%bd%ae)

// 设置Elasticsearch索引文档\_id

/\*\*

\* 如果指定rowkey为文档\_id,那么需要指定前缀meta:，如果是其他数据字段就不需要

\* 例如：

\* meta:rowkey 行key byte[]

\* meta:timestamp 记录时间戳

\*/

importBuilder.setEsIdField("meta:rowkey");

// 设置自定义id生成机制

//如果指定EsIdGenerator，则根据下面的方法生成文档id，

// 否则根据setEsIdField方法设置的字段值作为文档id，

// 如果默认没有配置EsIdField和指定EsIdGenerator，则由es自动生成文档id

// importBuilder.setEsIdGenerator(new EsIdGenerator(){

//

// @Override

// public Object genId(Context context) throws Exception {

// Object id = context.getMetaValue("rowkey");

// String agentId = BytesUtils.safeTrim(BytesUtils.toString((byte[]) id, 0, PinpointConstants.AGENT\_NAME\_MAX\_LEN));

// return agentId;

// }

// });

## [4.4 hbase检索条件设置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_73-hbase%e6%a3%80%e7%b4%a2%e6%9d%a1%e4%bb%b6%e8%ae%be%e7%bd%ae)

可以通过FilterList和filter设置hbase scan检索条件，二选一，只需要设置一种（本案例中不涉及检索条件的处理）

//FilterList和filter二选一，只需要设置一种

/\*\*

\* 设置hbase检索filter

\*/

SingleColumnValueFilter scvf= new SingleColumnValueFilter(Bytes.toBytes("Info"), Bytes.toBytes("i"),

CompareOperator.EQUAL,"wap".getBytes());

scvf.setFilterIfMissing(true); //默认为false， 没有此列的数据也会返回 ，为true则只返回name=lisi的数据

importBuilder.setFilter(scvf);

/\*\*

\* 设置hbase组合条件FilterList

\* FilterList 代表一个过滤器链，它可以包含一组即将应用于目标数据集的过滤器，过滤器间具有“与” FilterList.Operator.MUST\_PASS\_ALL 和“或” FilterList.Operator.MUST\_PASS\_ONE 关系

\*/

FilterList list = new FilterList(FilterList.Operator.MUST\_PASS\_ONE); //数据只要满足一组过滤器中的一个就可以

SingleColumnValueFilter filter1 = new SingleColumnValueFilter(Bytes.toBytes("Info"), Bytes.toBytes("i"),

CompareOperator.EQUAL,"wap".getBytes());

list.addFilter(filter1);

SingleColumnValueFilter filter2 = new SingleColumnValueFilter(Bytes.toBytes("Info"), Bytes.toBytes("i"),

CompareOperator.EQUAL,Bytes.toBytes("my other value"));

list.addFilter(filter2);

importBuilder.setFilterList(list);

//设置同步起始行和终止行key条件

importBuilder.setStartRow(startRow);

importBuilder.setEndRow(endRow);

设置记录起始时间搓（>=）和截止时间搓(<),如果是基于时间范围的增量同步，则不需要指定下面两个参数

importBuilder.setStartTimestamp(startTimestam);

importBuilder.setEndTimestamp(endTimestamp);

## [4.5 HBase数据处理](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_74-hbase%e6%95%b0%e6%8d%ae%e5%a4%84%e7%90%86)

必须通过DataRefactor接口处理Hbase数据并将数据添加到Elasticsearch文档中，代码示例如下：

/\*\*

\* 设置es数据结构

\*/

importBuilder.setDataRefactor(new DataRefactor() {

public void refactor(Context context) throws Exception {

//可以根据条件定义是否丢弃当前记录

//context.setDrop(true);return;

// if(s.incrementAndGet() % 2 == 0) {

// context.setDrop(true);

// return;

// }

//获取原始的hbase记录Result对象

HBaseRecord hBaseRecord = (HBaseRecord) context.getRecord();

Result result = (Result) hBaseRecord.getData();

// 直接获取行key，对应byte[]类型，自行提取和分析保存在其中的数据

String agentId = Bytes.toString((byte[])context.getMetaValue("rowkey"));

context.addFieldValue("agentId",agentId);

Date startTime = (Date)context.getMetaValue("timestamp");

context.addFieldValue("startTime",startTime);

// 通过context.getValue方法获取hbase 列的原始值byte[],方法参数对应hbase表中列名，由"列族:列名"组成

String serializedAgentInfo = context.getStringValue("Info:i");

String serializedServerMetaData = context.getStringValue("Info:m");

String serializedJvmInfo = context.getStringValue("Info:j");

context.addFieldValue("serializedAgentInfo",serializedAgentInfo);

context.addFieldValue("serializedServerMetaData",serializedServerMetaData);

context.addFieldValue("serializedJvmInfo",serializedJvmInfo);

context.addFieldValue("subtitle","小康");

context.addFieldValue("collectTime",new Date());

// /\*\*

// \* 获取ip对应的运营商和区域信息

// \*/

// IpInfo ipInfo = context.getIpInfo("Info:agentIp");

// if(ipInfo != null)

// context.addFieldValue("ipinfo", SimpleStringUtil.object2json(ipInfo));

// else{

// context.addFieldValue("ipinfo", "");

// }

// DateFormat dateFormat = SerialUtil.getDateFormateMeta().toDateFormat();

// Date optime = context.getDateValue("logOpertime",dateFormat);

// context.addFieldValue("logOpertime",optime);

// context.addFieldValue("collecttime",new Date());

}

});

//映射和转换配置结束

关键点：

* 获取原始的hbase记录Result对象：

HBaseRecord hBaseRecord = (HBaseRecord) context.getRecord();

Result result = (Result) hBaseRecord.getData();

* 获取列族中列byte[]

byte[] serializedAgentInfo = (byte[]) context.getValue("Info:i");

* 获取具体类型列族数据，并将数据添加到Elasticsearch文档中

// 通过context提供的一系列getXXXValue方法，从hbase列族中获取相应类型的数据：int,string,long,double,float,date

String agentName = context.getStringValue("Info:agentName");

context.addFieldValue("agentName",agentName);

* 获取ip对应的运营商和区域信息

/\*\*

\* 获取ip对应的运营商和区域信息

\*/

IpInfo ipInfo = context.getIpInfo("Info:agentIp");

if(ipInfo != null)

context.addFieldValue("ipinfo", SimpleStringUtil.object2json(ipInfo));

else{

context.addFieldValue("ipinfo", "");

}

* rowkey和timstamp信息获取

String agentId = Bytes.toString((byte[])context.getMetaValue("rowkey"));

context.addFieldValue("agentId",agentId);

Date startTime = (Date)context.getMetaValue("timestamp");

context.addFieldValue("startTime",startTime);

## [4.6 定时任务配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_75-%e5%ae%9a%e6%97%b6%e4%bb%bb%e5%8a%a1%e9%85%8d%e7%bd%ae)

//定时任务配置，

importBuilder.setFixedRate(false)//参考jdk timer task文档对fixedRate的说明

// .setScheduleDate(date) //指定任务开始执行时间：日期

.setDeyLay(1000L) // 任务延迟执行deylay毫秒后执行

.setPeriod(10000L); //每隔period毫秒执行，如果不设置，只执行一次

//定时任务配置结束

## [4.7 时间范围增量配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_76-%e6%97%b6%e9%97%b4%e8%8c%83%e5%9b%b4%e5%a2%9e%e9%87%8f%e9%85%8d%e7%bd%ae)

//hbase表中列名，由"列族:列名"组成

// //设置任务执行拦截器结束，可以添加多个

// //增量字段配置

//// importBuilder.setNumberLastValueColumn("Info:id");//指定数字增量查询字段变量名称

// importBuilder.setDateLastValueColumn("Info:logOpertime");//手动指定日期增量查询字段变量名称

importBuilder.setFromFirst(false);//任务重启时，重新开始采集数据，true 重新开始，false不重新开始，适合于每次全量导入数据的情况，如果是全量导入，可以先删除原来的索引数据

// importBuilder.setLastValueStorePath("hbase2esdemo\_import");//记录上次采集的增量字段值的文件路径，作为下次增量（或者重启后）采集数据的起点，不同的任务这个路径要不一样

//指定增量字段类型为日期类型，如果没有指定增量字段名称,则按照hbase记录时间戳进行timerange增量检索

importBuilder.setLastValueType(ImportIncreamentConfig.TIMESTAMP\_TYPE);

// ImportIncreamentConfig.NUMBER\_TYPE 数字类型

// // ImportIncreamentConfig.TIMESTAMP\_TYPE 日期类型

//设置增量查询的起始值时间起始时间

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

try {

Date date = format.parse("2000-01-01");

importBuilder.setLastValue(date);

}

catch (Exception e){

e.printStackTrace();

}

//增量配置结束

## [4.8 并行任务设置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_77-%e5%b9%b6%e8%a1%8c%e4%bb%bb%e5%8a%a1%e8%ae%be%e7%bd%ae)

/\*\*

\* 作业创建一个内置的线程池，实现多线程并行数据导入elasticsearch功能

\*/

importBuilder.setParallel(true);//设置为多线程并行批量导入,false串行

importBuilder.setQueue(10);//设置批量导入线程池等待队列长度

importBuilder.setThreadCount(50);//设置批量导入线程池工作线程数量

importBuilder.setContinueOnError(true);//任务出现异常，是否继续执行作业：true（默认值）继续执行 false 中断作业执行

importBuilder.setAsyn(false);//true 异步方式执行，不等待所有导入作业任务结束，方法快速返回；false（默认值） 同步方式执行，等待所有导入作业任务结束，所有作业结束后方法才返回

通过Parallel参数关闭和开启并行作业机制

## [4.9 任务执行明细统计设置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_78-%e4%bb%bb%e5%8a%a1%e6%89%a7%e8%a1%8c%e6%98%8e%e7%bb%86%e7%bb%9f%e8%ae%a1%e8%ae%be%e7%bd%ae)

/\*\*

\* 设置任务执行情况回调接口

\*/

importBuilder.setExportResultHandler(new ExportResultHandler<String,String>() {

@Override

public void success(TaskCommand<String,String> taskCommand, String result) {

TaskMetrics taskMetrics = taskCommand.getTaskMetrics();

logger.info(taskMetrics.toString());

}

@Override

public void error(TaskCommand<String,String> taskCommand, String result) {

TaskMetrics taskMetrics = taskCommand.getTaskMetrics();

logger.info(taskMetrics.toString());

}

@Override

public void exception(TaskCommand<String,String> taskCommand, Exception exception) {

TaskMetrics taskMetrics = taskCommand.getTaskMetrics();

logger.info(taskMetrics.toString());

}

@Override

public int getMaxRetry() {

return 0;

}

});

## [4.10 其他配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_79-%e5%85%b6%e4%bb%96%e9%85%8d%e7%bd%ae)

importBuilder.setPrintTaskLog(true); //可选项，true 打印任务执行日志（耗时，处理记录数） false 不打印，默认值false

importBuilder.setDebugResponse(false);//设置是否将每次处理的reponse打印到日志文件中，默认false

importBuilder.setDiscardBulkResponse(true);//设置是否需要批量处理的响应报文，不需要设置为false，true为需要，默认false

## [4.11 启动和执行作业](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_710-%e5%90%af%e5%8a%a8%e5%92%8c%e6%89%a7%e8%a1%8c%e4%bd%9c%e4%b8%9a)

/\*\*

\* 执行es数据导入数据库表操作

\*/

DataStream dataStream = importBuilder.builder();

dataStream.execute();//执行导入操作

# [5.构建部署](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_8%e6%9e%84%e5%bb%ba%e9%83%a8%e7%bd%b2)

## [5.1 准备工作](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_81-%e5%87%86%e5%a4%87%e5%b7%a5%e4%bd%9c)

需要通过gradle构建发布版本,gradle安装配置参考文档：

[**https://esdoc.bbossgroups.com/#/bboss-build**](https://esdoc.bbossgroups.com/#/bboss-build)

下载最新版本的hbase和elasticsearch部署启动完毕。

在hbase中创建以下表：

create 'AgentInfo', { NAME => 'Info', TTL => 31536000, DATA\_BLOCK\_ENCODING => 'PREFIX' }

## [5.2 下载源码工程-基于gradle](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_82-%e4%b8%8b%e8%bd%bd%e6%ba%90%e7%a0%81%e5%b7%a5%e7%a8%8b-%e5%9f%ba%e4%ba%8egradle)

[**https://github.com/bbossgroups/hbase-elasticsearch**](https://github.com/bbossgroups/hbase-elasticsearch)

从上面的地址下载源码工程，然后导入idea或者eclipse，根据自己的需求，修改导入程序

org.frameworkset.elasticsearch.imp.HBase2ESScrollTimestampDemo223

## [5.3 配置修改和测试及发布版本](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_83-%e9%85%8d%e7%bd%ae%e4%bf%ae%e6%94%b9%e5%92%8c%e6%b5%8b%e8%af%95%e5%8f%8a%e5%8f%91%e5%b8%83%e7%89%88%e6%9c%ac)

### [5.3.1 修改hbase地址](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_831-%e4%bf%ae%e6%94%b9hbase%e5%9c%b0%e5%9d%80)

修改hbase地址-hbase-elasticsearch\src\main\resources\application.properties

hbase.zookeeper.quorum = 192.168.137.133

hbase.zookeeper.property.clientPort = 2183

zookeeper.znode.parent = /hbase

### [5.3.2 修改es配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_832-%e4%bf%ae%e6%94%b9es%e9%85%8d%e7%bd%ae)

修改es配置-hbase-elasticsearch\src\main\resources\application.properties

##targetElasticsearch数据源配置，Hbase-Elasticsearch数据同步测试

# x-pack或者searchguard安全认证和口令配置

targetElasticsearch.elasticUser=elastic

targetElasticsearch.elasticPassword=changeme

targetElasticsearch.elasticsearch.rest.hostNames=192.168.137.1:9200

### [5.3.3 导入测试数据](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_833-%e5%af%bc%e5%85%a5%e6%b5%8b%e8%af%95%e6%95%b0%e6%8d%ae)

运行junit类test/java/org/frameworkset/elasticsearch/imp/HBaseHelperTest.java中的testPutDatas方法

@Test

public void testPutDatas(){

Map<String,String> properties = new HashMap<String, String>();

properties.put("hbase.zookeeper.quorum","192.168.137.133"); //根据实际情况修改

properties.put("hbase.zookeeper.property.clientPort","2183");//根据实际情况修改

properties.put("zookeeper.znode.parent","/hbase"); //根据实际情况修改

properties.put("hbase.ipc.client.tcpnodelay","true");

properties.put("hbase.rpc.timeout","10000");

properties.put("hbase.client.operation.timeout","10000");

properties.put("hbase.ipc.client.socket.timeout.read","20000");

properties.put("hbase.ipc.client.socket.timeout.write","30000");

//异步写入hbase

/\*\*

\* public static final String TABLE\_MULTIPLEXER\_FLUSH\_PERIOD\_MS = "hbase.tablemultiplexer.flush.period.ms";

\* public static final String TABLE\_MULTIPLEXER\_INIT\_THREADS = "hbase.tablemultiplexer.init.threads";

\* public static final String TABLE\_MULTIPLEXER\_MAX\_RETRIES\_IN\_QUEUE = "hbase.client.max.retries.in.queue";

\*/

properties.put("hbase.client.async.enable","true");

properties.put("hbase.client.async.in.queuesize","10000");

HBaseHelper.buildHBaseClient(properties,100,100,0L,1000l,1000,true,true,false);

HbaseTemplate2 hbaseTemplate2 = HBaseHelper.getHbaseTemplate2();

byte[] CF = Bytes.toBytes("Info");

byte[] C\_I = Bytes.toBytes("i");

byte[] C\_j = Bytes.toBytes("j");

byte[] C\_m = Bytes.toBytes("m");

final List<Put> datas = new ArrayList<>();

for(int i= 0; i < 100; i ++){

long timestamp = System.currentTimeMillis() ;

final byte[] rowKey = Bytes.toBytes(i+"-"+timestamp);

final Put put = new Put(rowKey, timestamp);

put.addColumn(CF, C\_I,timestamp, Bytes.toBytes( "wap\_"+i));

put.addColumn(CF, C\_j,timestamp, Bytes.toBytes( "jdk 1.8\_"+i));

put.addColumn(CF, C\_m,timestamp, Bytes.toBytes( "asdfasfd\_"+i));

datas.add(put);

}

TableName traceTableName = TableName.valueOf("AgentInfo");

hbaseTemplate2.asyncPut(traceTableName,datas);

}

### [5.3.4 作业调试](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_834-%e4%bd%9c%e4%b8%9a%e8%b0%83%e8%af%95)

修改完毕配置并导入测试数据后，就可以进行功能调试了。

如果需要测试和调试导入功能，运行HBase2ESScrollTimestampDemo223的main方法即可即可：

public class HBase2ESScrollTimestampDemo223 {

public static void main(String args[]){

HBase2ESScrollTimestampDemo223 esDemo = new HBase2ESScrollTimestampDemo223();

esDemo.scheduleScrollRefactorImportData();

}

.....

}

### [5.3.5 发布版本](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_835-%e5%8f%91%e5%b8%83%e7%89%88%e6%9c%ac)

测试调试通过后，就可以构建发布可运行的版本了：进入命令行模式，在源码工程根目录hbase-elasticsearch下运行以下gradle指令打包发布版本

release.bat

## [5.4 运行作业](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_84-%e8%bf%90%e8%a1%8c%e4%bd%9c%e4%b8%9a)

gradle构建成功后，在build/distributions目录下会生成可以运行的zip包，解压运行导入程序

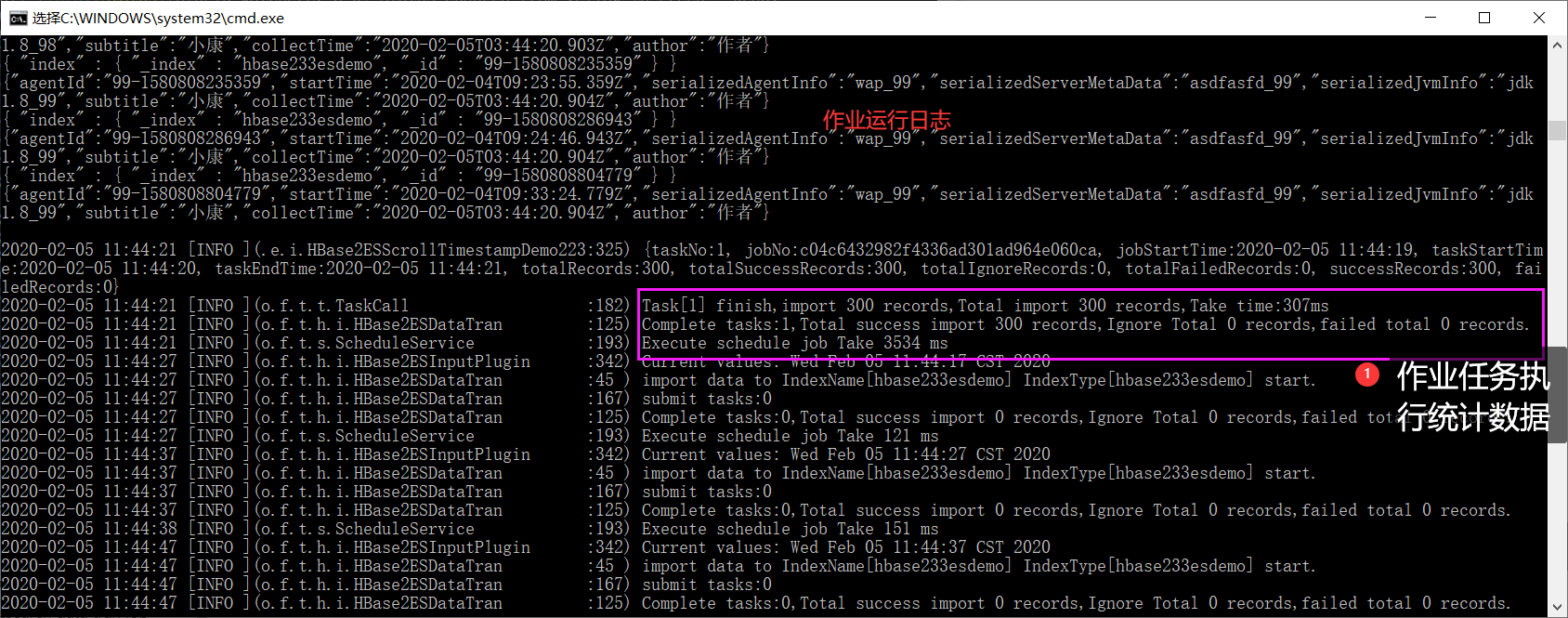
linux：

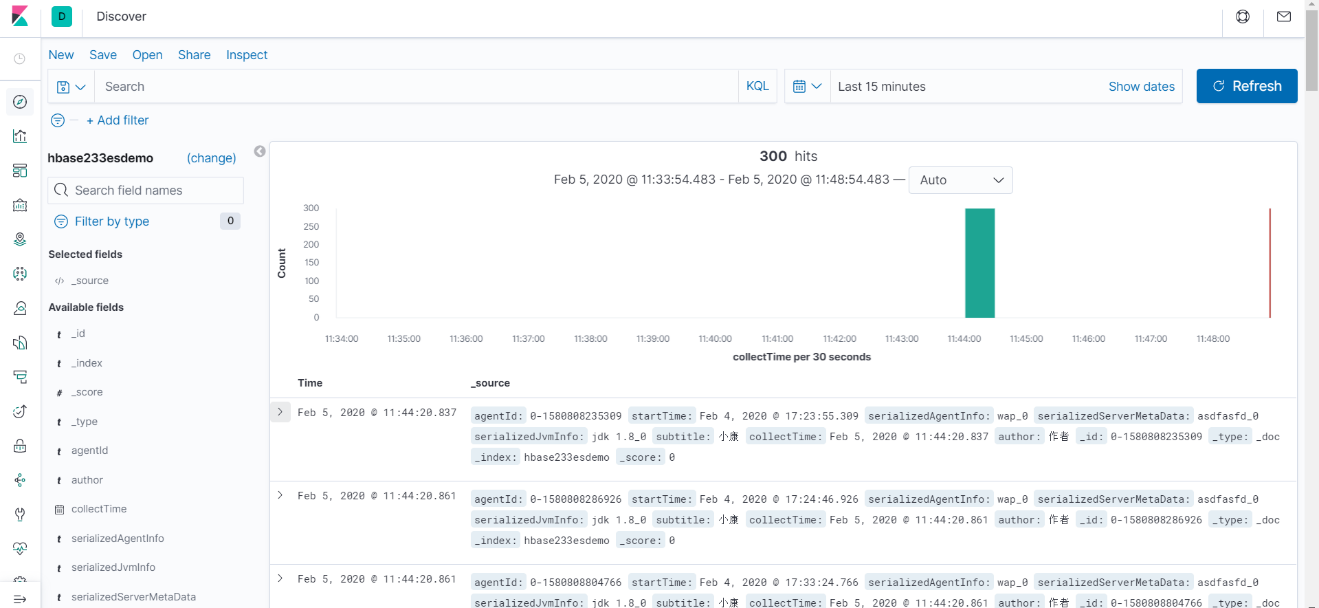
chmod +x restart.sh

./restart.sh

windows: restart.bat

控制台任务执行日志：



kibana检索从hbase同步到Elasticsearch的数据（kibana 7.5.0） 

## [5.5 作业jvm配置](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_85-%e4%bd%9c%e4%b8%9ajvm%e9%85%8d%e7%bd%ae)

修改jvm.options，设置内存大小和其他jvm参数

-Xms1g

-Xmx1g

# [6.作业代码中参数提取](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_9%e4%bd%9c%e4%b8%9a%e4%bb%a3%e7%a0%81%e4%b8%ad%e5%8f%82%e6%95%b0%e6%8f%90%e5%8f%96)

在使用[**hbase-elasticsearch**](https://github.com/bbossgroups/hbase-elasticsearch)时，会根据实际情况调整作业运行参数，为了避免调试和运行过程中修改作业代码中的参数不断打包构建发布数据同步工具，可以将写死在代码中的控制参数提取到启动配置文件resources/application.properties中,然后在代码中通过以下方法获取配置的参数：

#工具主程序

mainclass=org.frameworkset.elasticsearch.imp.HBase2ESScrollTimestampDemo223

# 参数配置

dropIndice=false

# 在代码中获取方法：CommonLauncher.getBooleanAttribute("dropIndice",false);//同时指定了默认值false

在代码中获取参数dropIndice方法：

boolean dropIndice = CommonLauncher.getBooleanAttribute("dropIndice",false);//同时指定了默认值false

另外可以在resources/application.properties配置控制作业执行的一些参数，例如工作线程数，等待队列数，批处理size等等：

queueSize=50

workThreads=10

batchSize=20

在作业执行方法中获取并使用上述参数：

int batchSize = CommonLauncher.getIntProperty("batchSize",10);//同时指定了默认值

int queueSize = CommonLauncher.getIntProperty("queueSize",50);//同时指定了默认值

int workThreads = CommonLauncher.getIntProperty("workThreads",10);//同时指定了默认值

importBuilder.setBatchSize(batchSize);

importBuilder.setQueue(queueSize);//设置批量导入线程池等待队列长度

importBuilder.setThreadCount(workThreads);//设置批量导入线程池工作线程数量

# [7.spring boot web应用中启动和停止调度hbase同步作业](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_10spring-boot-web%e5%ba%94%e7%94%a8%e4%b8%ad%e5%90%af%e5%8a%a8%e5%92%8c%e5%81%9c%e6%ad%a2%e8%b0%83%e5%ba%a6hbase%e5%90%8c%e6%ad%a5%e4%bd%9c%e4%b8%9a)

## [7.1 准备工作](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_101-%e5%87%86%e5%a4%87%e5%b7%a5%e4%bd%9c)

下载包含hbase-elasticsearch同步作业的spring boot elasticsearch web应用

[**https://github.com/bbossgroups/springboot-elasticsearch-webservice**](https://github.com/bbossgroups/springboot-elasticsearch-webservice)

下载后参数上面的文档，修改同步作业程序中hbase地址：

[**https://github.com/bbossgroups/springboot-elasticsearch-webservice/blob/master/src/main/java/com/example/esbboss/service/DataTran.java**](https://github.com/bbossgroups/springboot-elasticsearch-webservice/blob/master/src/main/java/com/example/esbboss/service/DataTran.java)

/\*\*

\* hbase参数配置

\*/

importBuilder.addHbaseClientProperty("hbase.zookeeper.quorum","192.168.137.133") //hbase客户端连接参数设置，参数含义参考hbase官方客户端文档

.addHbaseClientProperty("hbase.zookeeper.property.clientPort","2183")

.addHbaseClientProperty("zookeeper.znode.parent","/hbase")

修改src/main/resources/application.properties中的elasticsearch地址：

spring.elasticsearch.bboss.elasticsearch.rest.hostNames=192.168.137.1:9200

然后参考以下步骤构建和运行、停止作业。

## [7.2 run spring boot web项目](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_102-run-spring-boot-web%e9%a1%b9%e7%9b%ae)

First run elasticsearch 5 or elasticsearch 6 or Elasticsearch 7.Then build and run the demo:

mvn install

cd target

java -jar springboot-elasticsearch-webservice-0.0.1-SNAPSHOT.jar

## [7.3 run the hbase-elasticsearch data tran job](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_103-run-the-hbase-elasticsearch-data-tran-job)

Enter the following address in the browser to run the hbase-elasticsearch data tran job:

[**http://localhost:808/scheduleHBase2ESJob**](http://localhost:808/scheduleHBase2ESJob)

Return the following results in the browser to show successful execution:

作业启动成功

HBase2ES job started.

作业已经启动

HBase2ES job has started.

## [7.4 stop the db-elasticsearch data tran job](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_104-stop-the-db-elasticsearch-data-tran-job)

Enter the following address in the browser to stop the hbase-elasticsearch data tran job:

[**http://localhost:808/stopHBase2ESJob**](http://localhost:808/stopHBase2ESJob)

Return the following search results in the browser to show successful execution: 作业停止成功

HBase2ES job started.

作业已经停止

HBase2ES job has started.

# [8.开发交流](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_11%e5%bc%80%e5%8f%91%e4%ba%a4%e6%b5%81)

## [8.1 相关文档](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_111-%e7%9b%b8%e5%85%b3%e6%96%87%e6%a1%a3)

* [**数据库和Elasticsearch同步工具**](https://esdoc.bbossgroups.com/#/db-es-tool)
* [**Spring boot与数据同步工具应用**](https://esdoc.bbossgroups.com/#/usedatatran-in-spring-boot)
* [**Mongodb-Elasticsearch同步工具**](https://esdoc.bbossgroups.com/#/mongodb-elasticsearch)
* [**Database-Database数据同步使用方法**](https://esdoc.bbossgroups.com/#/db-es-tool?id=_5-database-database%e6%95%b0%e6%8d%ae%e5%90%8c%e6%ad%a5%e4%bd%bf%e7%94%a8%e6%96%b9%e6%b3%95)
* [**Kafka1x-Elasticsearch数据同步使用方法**](https://esdoc.bbossgroups.com/#/db-es-tool?id=_6-kafka1x-elasticsearch%e6%95%b0%e6%8d%ae%e5%90%8c%e6%ad%a5%e4%bd%bf%e7%94%a8%e6%96%b9%e6%b3%95)
* [**Kafka2x-Elasticsearch数据同步使用方法**](https://esdoc.bbossgroups.com/#/db-es-tool?id=_7-kafka2x-elasticsearch%e6%95%b0%e6%8d%ae%e5%90%8c%e6%ad%a5%e4%bd%bf%e7%94%a8%e6%96%b9%e6%b3%95)
* [**Elasticsearch-Elasticsearch数据同步使用方法**](https://esdoc.bbossgroups.com/#/db-es-tool?id=_8-elasticsearch-elasticsearch%e6%95%b0%e6%8d%ae%e5%90%8c%e6%ad%a5%e4%bd%bf%e7%94%a8%e6%96%b9%e6%b3%95)

## [8.2 讨论交流](https://esdoc.bbossgroups.com/#/hbase-elasticsearch?id=_112-%e8%ae%a8%e8%ae%ba%e4%ba%a4%e6%b5%81)

elasticsearch技术交流群:166471282

elasticsearch微信公众号:bbossgroup

