# DATAX文档

## DATAX简介

### DataX是什么

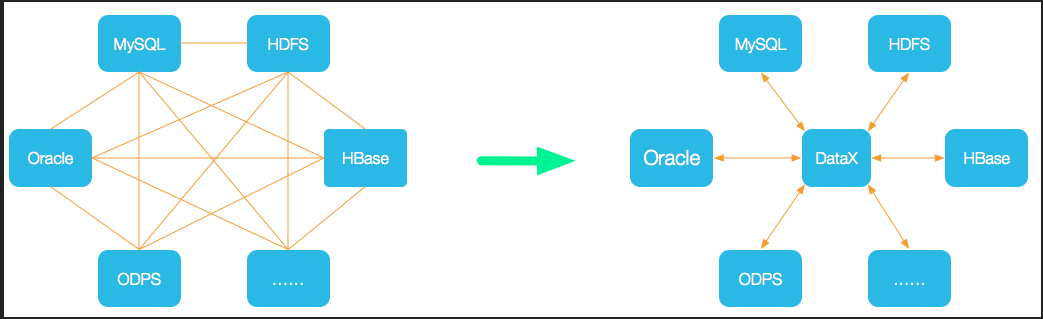
DataX 是阿里巴巴集团内被广泛使用的离线数据同步工具/平台，实现包括 MySQL、SQL Server、Oracle、PostgreSQL、HDFS、Hive、HBase、OTS、ODPS 等各种异构数据源之间高效的数据同步功能。

### 何时用DataX

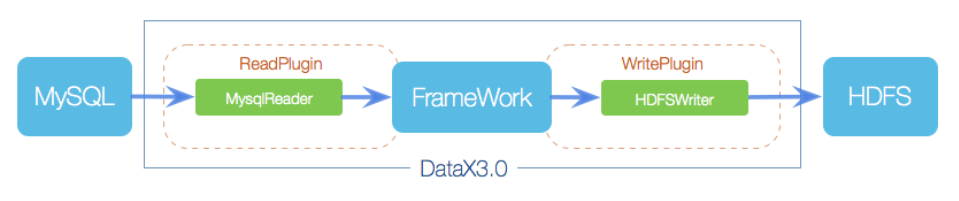
DataX是离线数据同步工具，当需要迁移增量时，建议使用DTS，而不是DataX；针对离线数据，当数据量很大或表非常多时，建议使用DataX。此时配置文件可编写脚本批量生成，详见[ODPS数据迁移指南](https://yq.aliyun.com/articles/71062)。同时可以增大DataX本身的并发，并提高运行DataX的任务机数量，来达到高并发，从而实现快速迁移；

## 框架设计

DataX本身作为数据同步框架，将不同数据源的同步抽象为从源头数据源读取数据的Reader插件，以及向目标端写入数据的Writer插件，理论上DataX框架可以支持任意数据源类型的数据同步工作。同时DataX插件体系作为一套生态系统, 每接入一套新数据源该新加入的数据源即可实现和现有的数据源互通。



为了解决异构数据源同步问题，DataX将复杂的网状的同步链路变成了星型数据链路，DataX作为中间传输载体负责连接各种数据源。当需要接入一个新的数据源的时候，只需要将此数据源对接到DataX，便能跟已有的数据源做到无缝数据同步。



DataX本身作为离线数据同步框架，采用Framework + plugin架构构建。将数据源读取和写入抽象成为Reader/Writer插件，纳入到整个同步框架中。

Reader：Reader为数据采集模块，负责采集数据源的数据，将数据发送给Framework。

Writer： Writer为数据写入模块，负责不断向Framework取数据，并将数据写入到目的端。

Framework：Framework用于连接reader和writer，作为两者的数据传输通道，并处理缓冲，流控，并发，数据转换等核心技术问题。

## 环境要求

### 1.Linux

### 2.[JDK(1.6以上，推荐1.6)](http://www.oracle.com/technetwork/cn/java/javase/downloads/index.html)

JDK [下载地址](http://download.oracle.com/otn-pub/java/jdk/8u171-b11/512cd62ec5174c3487ac17c61aaa89e8/jdk-8u171-linux-x64.tar.gz)http://download.oracle.com/otn-pub/java/jdk/8u171-b11/512cd62ec5174c3487ac17c61aaa89e8/jdk-8u171-linux-x64.tar.gz

解压 jdk-8u171-linux-x64.tar.gz安装包

[root@localhost software]# mkdir -p /usr/java

[root@localhost software]# tar -zxvf jdk-8u171-linux-x64.tar.gz -C /usr/lib/jvm

设置环境变量

[root@localhost software]# vim /etc/profile

在最前面添加：

export JAVA\_HOME=/usr/java/jdk1.8.0\_131

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

执行profile文件

[root@localhost software]# source /etc/profile

这样可以使配置不用重启即可立即生效。

检查新安装的jdk

[root@localhost software]# java -version

显示：

java version "1.8.0\_131"

Java(TM) SE Runtime Environment (build 1.8.0\_131-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)

建立软连接 防止后面设置定时任务报错 提示：/bin/sh: java: command not found。

[root@localhost software]# ln -s /usr/java/jdk/bin/jar /bin/jar

[root@localhost software]# ln -s /usr/java/jdk/bin/java /bin/java

[root@localhost software]# ln -s /usr/java/jdk/bin/javac /bin/javac

[root@localhost software]# ln -s /usr/java/jdk/bin/javah /bin/javah

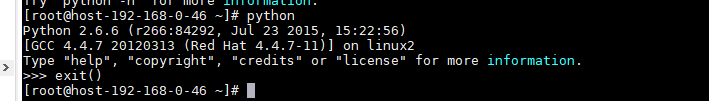
[root@localhost software]# ln -s /usr/java/jdk/bin/javadoc /bin/javadoc

到此为止，整个安装过程结束。

### 3.[Python(推荐Python2.6.X)](https://www.python.org/downloads/)

Linux自带2.6.6版本的Python 无需安装

[root@localhost software]# Python



输入Python 进入命令窗口 输入exit() 退出

## 部署运行

直接下载DataX工具包：[DataX下载地址](http://datax-opensource.oss-cn-hangzhou.aliyuncs.com/datax.tar.gz)

下载后解压至本地某个目录，进入bin目录，即可运行同步作业：

$tar -zxvf datax.tar.gz

$ cd {YOUR\_DATAX\_HOME}/bin

$ python datax.py {YOUR\_JOB.json}

自检脚本：    python {YOUR\_DATAX\_HOME}/bin/datax.py {YOUR\_DATAX\_HOME}/job/job.json

配置示例：从stream读取数据并打印到控制台

第一步、创建创业的配置文件（json格式）

可以通过命令查看配置模板： python datax.py -r {YOUR\_READER} -w {YOUR\_WRITER}

$ cd {YOUR\_DATAX\_HOME}/bin

$ python datax.py -r streamreader -w streamwriter

DataX (UNKNOWN\_DATAX\_VERSION), From Alibaba !

Copyright (C) 2010-2015, Alibaba Group. All Rights Reserved.

Please refer to the streamreader document:

https://github.com/alibaba/DataX/blob/master/streamreader/doc/streamreader.md

Please refer to the streamwriter document:

https://github.com/alibaba/DataX/blob/master/streamwriter/doc/streamwriter.md

Please save the following configuration as a json file and use

python {DATAX\_HOME}/bin/datax.py {JSON\_FILE\_NAME}.json

to run the job.

{

"job": {

"content": [

{

"reader": {

"name": "streamreader",

"parameter": {

"column": [],

"sliceRecordCount": ""

}

},

"writer": {

"name": "streamwriter",

"parameter": {

"encoding": "",

"print": true

}

}

}

],

"setting": {

"speed": {

"channel": ""

}

}

}

}

根据模板配置json如下：

#stream2stream.json

{

"job": {

"content": [

{

"reader": {

"name": "streamreader",

"parameter": {

"sliceRecordCount": 10,

"column": [

{

"type": "long",

"value": "10"

},

{

"type": "string",

"value": "hello，你好，世界-DataX"

}

]

}

},

"writer": {

"name": "streamwriter",

"parameter": {

"encoding": "UTF-8",

"print": true

}

}

}

],

"setting": {

"speed": {

"channel": 5

}

}

}

}

第二步：启动DataX

$ cd {YOUR\_DATAX\_DIR\_BIN}

$ python datax.py ./stream2stream.json

同步结束，显示日志如下：

...

2015-12-17 11:20:25.263 [job-0] INFO JobContainer -

任务启动时刻 : 2015-12-17 11:20:15

任务结束时刻 : 2015-12-17 11:20:25

任务总计耗时 : 10s

任务平均流量 : 205B/s

记录写入速度 : 5rec/s

读出记录总数 : 50

读写失败总数 : 0

## 配置文件详解

本例中源和目的都是ODPS 如下是DataX的配置文件示例：

{

"job": {

"content":[

{

"reader":{

"name":"odpsreader",

"parameter":{

"accessId":"<accessID>",

"accessKey":"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",

"column":[

"col\_1",

"col\_2"

],

"odpsServer":"http://service.odps.aliyun-inc.com/api",

"partition":[

"dt=20160524"

],

"project":"src\_project\_name",

"splitMode":"record",

"table":"table\_name\_1"

}

},

"writer":{

"name":"odpswriter",

"parameter":{

"accessId":"<accessId>",

"accessKey":"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",

"accountType":"aliyun",

"column":[

"ci\_name",

"geohash"

],

"odpsServer":"http://service.odps.xxx.com/api",

"partition":"dt=20160524",

"project":"dst\_project\_name",

"table":"nb\_tab\_http"

}

}

}

],

"setting":{

"speed":{

"channel":20

}

}

}

}

1. 整个配置文件是一个job的描述；
2. job下面有两个配置项，content和setting，其中content用来描述该任务的源和目的端的信息，setting用来描述任务本身的信息；
3. content又分为两部分，reader和writer，分别用来描述源端和目的端的信息；
4. 本例中由于源和目的都是ODPS，所以类型为odpsreader和odpswriter。均包含accessId，accessKey与odpsServer的api地址。
5. 同时预迁移表的project，table以及列名和分区信息都要一一填写清楚。
6. setting中的speed项表示同时起20个并发去跑该任务。

## 定时任务

### 1.安装crond服务

yum install crontabs

命令说明：

/sbin/service crond start //启动服务  
/sbin/service crond stop //关闭服务  
/sbin/service crond restart //重启服务  
/sbin/service crond reload //重新载入配置

查看crond服务状态   输入

service crond status

显示如下表示服务已启动

[root@localhost /]# service crond status  
Redirecting to /bin/systemctl status  crond.service  
● crond.service - Command Scheduler  
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2017-05-15 15:51:05 CST; 9min ago  
 Main PID: 6550 (crond)  
   CGroup: /system.slice/crond.service  
           └─6550 /usr/sbin/crond -n  
May 15 15:51:05 localhost.localdomain systemd[1]: Started Command Scheduler.  
May 15 15:51:05 localhost.localdomain systemd[1]: Starting Command Scheduler...  
May 15 15:51:05 localhost.localdomain crond[6550]: (CRON) INFO (RANDOM\_DELAY ...  
May 15 15:51:06 localhost.localdomain crond[6550]: (CRON) INFO (running with ...  
May 15 15:51:06 localhost.localdomain crond[6550]: (CRON) INFO (@reboot jobs ...  
Hint: Some lines were ellipsized, use -l to show in full.

### 2.创建作业配置文件（json格式）

这个是从一个mysql数据库读取数据并写入mysql数据库的一个作业配置，

mysql2mysql.json

{

"job": {

"content": [

{

"reader": {

"name": "mysqlreader",

"parameter": {

"column": ["id","tags\_id","financing\_count","ipo\_count","merge\_count","exit\_count","update\_time","modify\_time"],

"connection": [

{

"jdbcUrl": ["jdbc:mysql://119.254.13.244:3306/datacenter\_cmp"],

"table": ["cmp\_tags\_event"]

}

],

"password": "VYxIgZY4FhHFmPvG",

"username": "cvs"

}

},

"writer": {

"name": "mysqlwriter",

"parameter": {

"column": ["id","tags\_id","financing\_count","ipo\_count","merge\_count","exit\_count","update\_time","modify\_time"],

"connection": [

{

"jdbcUrl": "jdbc:mysql://119.254.13.244:3306/datacenter\_cmp",

"table": ["cmp\_tags\_event2"]

}

],

"password": "VYxIgZY4FhHFmPvG",

"username": "cvs"

}

}

}

],

"setting": {

"speed": {

"channel": "1"

}

}

}

}

{

"job": {

"setting": {

"speed": {

"byte":1048

},

},

"content": [

{

"reader": {

"name": "oraclereader",

"parameter": {

"column" : [

"name","age"

],

"connection":[{

"jdbcUrl":["jdbc:oracle:thin:@192.168.10.15:1521:orcl"],

"table":["TEST\_A"],

}],

"password":"123456",

"username":"root",

}

},

"writer": {

"name": "oraclewriter",

"parameter": {

"username": "root",

"password": "111222",

"column": ["filed1","filed2"],

"preSql": [

"delete from TEST\_B"

],

"connection":[{

"jdbcUrl":"jdbc:oracle:thin:@192.168.11.18:1521:orcl",

"table":["TEST\_B"],

}],

}

}

}

]

}

}

### 3.列出crontab文件，输入crontab -l

若显示以下内容 ，则表示当前用户没有定时任务（xxx表示当前用户），需要创建crontab文件，并提交，查看步骤（4）

no crontal for xxx

若显示类似以下内容 ，则表示当前用户已经有正在运行的定时任务，只需输入crontab -e 来编辑crontab文件，查看步骤（5）

0,10,20,35,44,50 \* \* \* \* python /home/admin/datax3/bin/datax.py /home/admin/mysql2odps.json >>/home/hanfa.shf/log.`date +\%Y\%m\%d\%H\%M\%S`  2>&1，

### 4.创建crontab文件(以前从未创建过crontab任务)

创建一个名为crondatax的crontab文件，命令行进入到相应的目录   vi crondatax，创建并打开文件，输入以下内容：

5  13 \* \* \*  python /home/admin/datax3/bin/datax.py /home/admin/mysql2mysql.json  >>/home/hanfa.shf/log.`date +\%Y\%m\%d\%H\%M\%S`  2>&1

5 13 \* \* \*表示每天的13点5分执行这个任务。

python/home/admin/datax3/bin/datax.py表示安装的DataX datax.py所在的目录的绝对路径，一般在datax/bin/目录下。

/home/admin/oracle2oracle.json.json表示作业配置文件的绝对路径

/home/hanfa.shf/log.`date +\%Y\%m\%d\%H\%M\%S`表示任务运行时产生日志的输出路径，并以log.当前时间命名，要替换成真实存在的绝对路径。

文件编辑完成之后，按esc，再按shift+;，再输入wq，则保存并退出文件编辑。

### 5.编辑已有crontab文件，修改或增加定时任务

输入crontab -e 打开任务配置文件。增加任务即可，格式和步骤（4）一致。

### 6.提交 crontab文件

输入crontab /home/datax/job/crondatax crondatax是步骤(4)中创建的文件的名称，如果终端所在目录就是crondatax文件所在目录，

可以直接输入crontab crondatax。

提交之后输入crontab -l 就能看到刚配置的任务。

### 7、重启crontab服务

输入

/sbin/service crond restart

在定时的时间到后，任务便开始执行，同时会在配置的log目录下生成对应的日志文件，在日志文件中可查看任务的运行情况。

### 8、任务配置文件crontab命令格式说明

参数可以配置多个，中间用逗号分隔比如想每10分钟执行一次定时任务，可以按以下方式配置。

0,10,20,30,40,50  \* \* \* \*  python /home/admin/datax3/bin/datax.py /home/admin/mysql2odps.json  >>/home/hanfa.shf/log.`date +\%Y\%m\%d\%H\%M\%S`  2>&1



## Elasticsearch 5.5.1部署和启动

Elastic 是 Lucene 的封装，提供了 REST API 的操作接口，开箱即用。

### 环境要求 [JDK(1.8以上）](http://www.oracle.com/technetwork/cn/java/javase/downloads/index.html)

安装步骤同（三、2）

### 下载Elasticsearch 5.5.1并解压

$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.5.1.zip $ unzip elasticsearch-5.5.1.zip

1. 创建elasticsearch 用户并授权

因为版本的问题，最新的版本安全级别提高了，不允许采用root帐号启动，所以我们要添加一个用户。

#添加一个用户：elasticsearch

$useradd elasticsearch

#给用户elasticsearch设置密码，连续输入2次

$passwd elasticsearch

将当前elasticsearch文件夹权限给elasticsearch用户

$chown elasticsearch -R /data/elasticsearch-5.5.1

切换到 elasticsearch 用户

$su elasticsearch

### 启动 Elastic

进入解压后的目录，运行下面的命令

$ cd elasticsearch-5.5.1/

$ ./bin/elasticsearch

（建议通过后台启动 命令如下）

$ nohup ./bin/elasticsearch &

（查看日志输出 命令如下）

$ tail -f nohup.out

### 遇到的错误及解决方式

问题二：ERROR: bootstrap checks failed max file descriptors [4096] for elasticsearch process likely too low, increase to at least [65536] max number of threads [1024] for user [lishang] likely too low, increase to at least [2048]

解决：切换到root用户，编辑limits.conf 添加类似如下内容

vi /etc/security/limits.conf

添加如下内容:

\* soft nofile 65536

\* hard nofile 131072

\* soft nproc 2048

\* hard nproc 4096

问题三：max number of threads [1024] for user [lish] likely too low, increase to at least [2048]

解决：切换到root用户，进入limits.d目录下修改配置文件。

vi /etc/security/limits.d/90-nproc.conf

修改如下内容：

\* soft nproc 1024

#修改为 \* soft nproc 2048

问题四：max virtual memory areas vm.max\_map\_count [65530] likely too low, increase to at least [262144]

解决：切换到root用户修改配置sysctl.conf

vi /etc/sysctl.conf

添加下面配置： vm.max\_map\_count=655360

并执行命令： sysctl -p

然后，重新启动elasticsearch，即可启动成功。

如果一切正常，Elastic 就会在默认的9200端口运行。这时，打开另一个命令行窗口，请求该端口，会得到说明信息。

$ curl localhost:9200

{

"name" : "atntrTf",

"cluster\_name" : "elasticsearch",

"cluster\_uuid" : "tf9250XhQ6ee4h7YI11anA",

"version" : {

"number" : "5.5.1",

"build\_hash" : "19c13d0",

"build\_date" : "2017-07-18T20:44:24.823Z",

"build\_snapshot" : false,

"lucene\_version" : "6.6.0"

},

"tagline" : "You Know, for Search"}

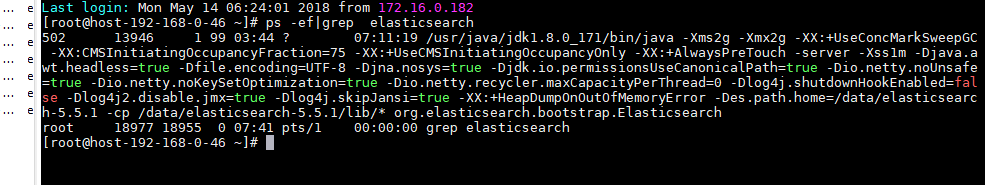
上面代码中，请求9200端口，Elastic 返回一个 JSON 对象，包含当前节点、集群、版本等信息。

按下 Ctrl + C，Elastic 就会停止运行。

### 查看及杀死elastic进程

$ ps -ef|grep elasticsearch

如下图说明当前有elasticsearch的进程存在 进程号为 13946



杀死当前进程

$ kill -9 进程号

## Mysql到Elasticsearch 数据同步

Datax 目前支持Elasticsearch 数据同步 有elasticsearchwriter 插件

### 首先通过命令得到Mysql到Elasticsearch的配置模板

进入datax bin 目录

$ cd {YOUR\_DATAX\_HOME}/bin

$ python datax.py -r mysqlreader -w elasticsearchwriter

{

"job": {

"content": [

{

"reader": {

"name": "mysqlreader",

"parameter": {

"column": [],

"connection": [

{

"jdbcUrl": [],

"table": []

}

],

"password": "",

"username": "",

"where": ""

}

},

"writer": {

"name": "elasticsearchwriter",

"parameter": {

"accessId": "OyR5xxxxxxxxXaXi",

"accessKey": "Z3wVxxxxxxxxxxxxxxxxxxxxxxxxRZ",

"batchSize": 100,

"cleanup": false,

"column": [

{

"name": "phone",

"type": "text"

}

],

"discovery": true,

"endpoint": "http://127.0.0.1:9200",

"index": "school\_index",

"type": "user\_info"

}

}

}

],

"setting": {

"speed": {

"channel": ""

}

}

}

}

### Mysql和elasticsearchwriter之间的数据Mapping

1. {
2. "job": {
3. "setting": {
4. ...
5. },
6. "content": [
7. {
8. "reader": {
9. ...
10. },
11. "writer": {
12. "name": "elasticsearchwriter",
13. "parameter": {
14. "endpoint": ""http://xxxx.com:9999"",
15. "accessId": "xxxx",
16. "accessKey": "yyyy",
17. "index": "test-1",
18. "type": "default",
19. "cleanup": true,
20. "settings": {"index" :{"number\_of\_shards": 1, "number\_of\_replicas": 0}},
21. "discovery": false,
22. "batchSize": 1000,
23. "splitter": ",",
24. "column": [
25. {"name": "pk", "type": "id"},
26. { "name": "col\_ip","type": "ip" },
27. { "name": "col\_double","type": "double" },
28. { "name": "col\_long","type": "long" },
29. { "name": "col\_integer","type": "integer" },
30. { "name": "col\_keyword", "type": "keyword" },
31. { "name": "col\_text", "type": "text", "analyzer": "ik\_max\_word"},
32. { "name": "col\_geo\_point", "type": "geo\_point" },
33. { "name": "col\_date", "type": "date", "format": "yyyy-MM-dd HH:mm:ss"},
34. { "name": "col\_nested1", "type": "nested" },
35. { "name": "col\_nested2", "type": "nested" },
36. { "name": "col\_object1", "type": "object" },
37. { "name": "col\_object2", "type": "object" },
38. { "name": "col\_integer\_array", "type":"integer", "array":true},
39. { "name": "col\_geo\_shape", "type":"geo\_shape", "tree": "quadtree", "precision": "10m"}
40. ]
41. }
42. }
43. }
44. ]
45. }
46. }

### 根据模板新建json文件作为job

例如：testESjob2.json

{

"job": {

"content": [

{

"reader": {

"name": "mysqlreader",

"parameter": {

"column": ["id","ipo\_count"],

"connection": [

{

"jdbcUrl": ["jdbc:mysql://192.168.0.68:3306/datacenter\_online"],

"table": ["cmp\_tags\_event"]

}

],

"password": "VYxIgZY4FhHFmPvG",

"username": "cvs",

"where": "update\_time is not null"

}

},

"writer": {

"name": "elasticsearchwriter",

"parameter": {

"accessId": "1",

"accessKey": "1",

"batchSize": 100,

"cleanup": false,

"column": [

{

"name": "id",

"type": "id"

},

{

"name": "ipo\_count",

"type": "long"

}

],

"discovery": true,

"endpoint": "http://127.0.0.1:9200",

"index": "company\_index",

"type": "tyc\_info"

}

}

}

],

"setting": {

"speed": {

"channel": "10"

}

}

}

}

### 运行命令导入数据测试

$ cd {YOUR\_DATAX\_HOME}/bin

$ python datax.py {YOUR\_JOB.json}

同步结束，显示日志如下说明导入成功：

...

2015-12-17 11:20:25.263 [job-0] INFO JobContainer -

任务启动时刻 : 2015-12-17 11:20:15

任务结束时刻 : 2015-12-17 11:20:25

任务总计耗时 : 10s

任务平均流量 : 205B/s

记录写入速度 : 5rec/s

读出记录总数 : 50

读写失败总数 : 0

### 查询ES服务数据测试是否导入成功

查询指定索引和类型下的所有结果

$ curl 'localhost:9200/{YOUR\_INDEX}/{YOUR\_INDEX\_TYPE}/\_search'

返回json数据如下说明插入成功

{ "took":2, "timed\_out":false, "\_shards":{"total":5,"successful":5,"failed":0}, "hits":{ "total":2, "max\_score":1.0, "hits":[ { "\_index":"accounts", "\_type":"person", "\_id":"AV3qGfrC6jMbsbXb6k1p", "\_score":1.0, "\_source": { "user": "李四", "title": "工程师", "desc": "系统管理" } }, { "\_index":"accounts", "\_type":"person", "\_id":"1", "\_score":1.0, "\_source": { "user" : "张三", "title" : "工程师", "desc" : "数据库管理，软件开发" } } ] } }

## Mysql到Elasticsearch 数据定时增量同步改造

### 1.定义job.json

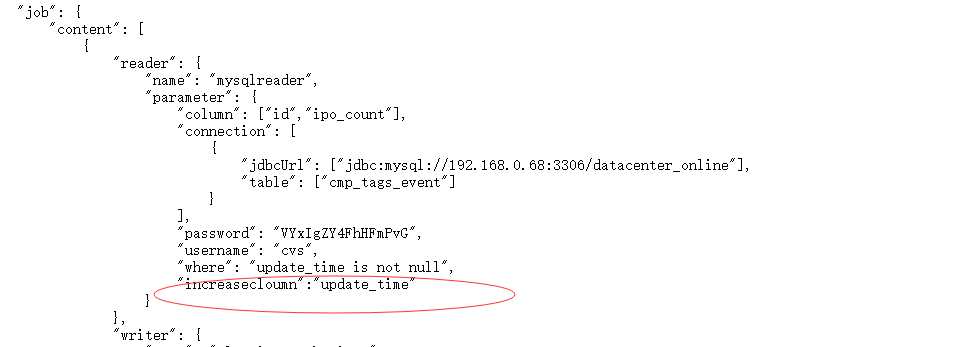
一般的一次性增量任务可以通过模板的where条件来实现 例如



对于定时自动增量同步datax没有给出解决方案

定时自动增量同步需要一个变量作为增量的依据

通过对mysqlreader的改造 增加了参照列 increasecloumn 来指定当前表的那一列作为参照列 如下图所示



完整例子：

{

"job": {

"content": [

{

"reader": {

"name": "mysqlreader",

"parameter": {

"column": ["id","ipo\_count"],

"connection": [

{

"jdbcUrl": ["jdbc:mysql://192.168.0.68:3306/datacenter\_online"],

"table": ["cmp\_tags\_event"]

}

],

"password": "VYxIgZY4FhHFmPvG",

"username": "cvs",

"where": "update\_time is not null",

"increasecloumn":"update\_time"

}

},

"writer": {

"name": "elasticsearchwriter",

"parameter": {

"accessId": "1",

"accessKey": "1",

"batchSize": 100,

"cleanup": false,

"column": [

{

"name": "id",

"type": "id"

},

{

"name": "ipo\_count",

"type": "long"

}

],

"discovery": true,

"endpoint": "http://127.0.0.1:9200",

"index": "company\_index",

"type": "tyc\_info"

}

}

}

],

"setting": {

"speed": {

"channel": "10"

}

}

}

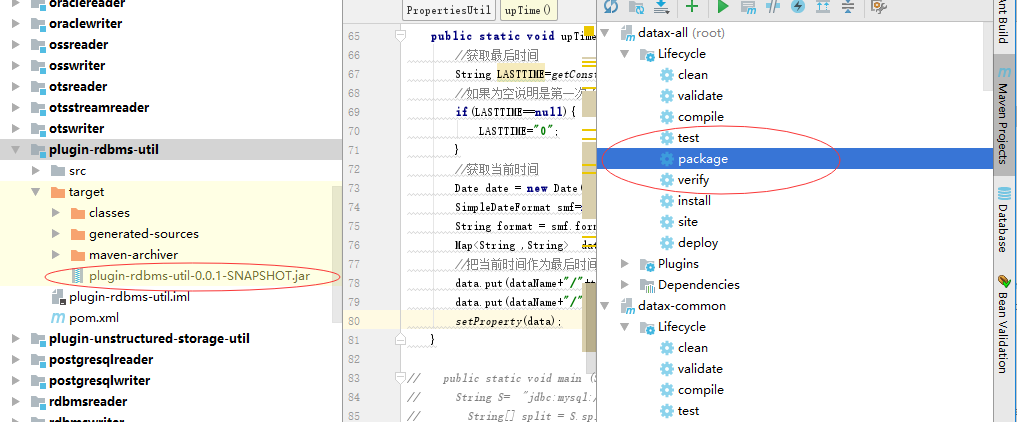
}

### 2.down代码重新打包 替换指定jar包

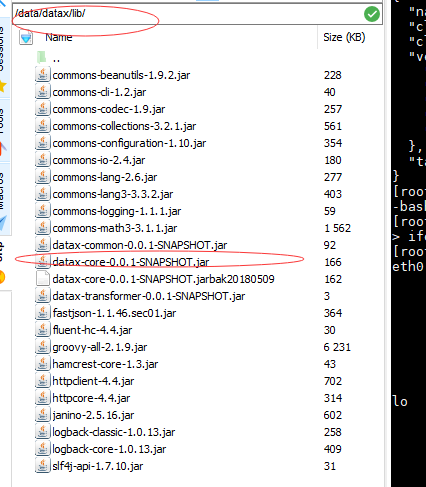
修改后代码的git地址 [ssh://git](ssh://git" \t "_blank)@192.168.2.220:10022/tyc/datax.git

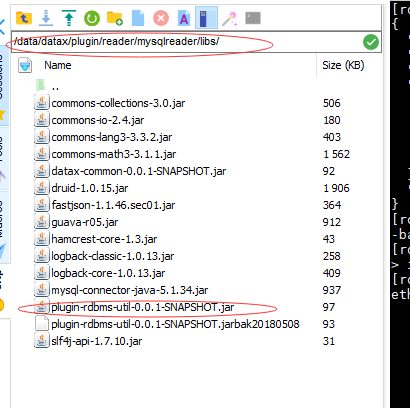
代码更改了 datax-core plugin-rdbms-util 两个模块 需要把代码 down下来 把两个模块重新打包 使用maven package 打包 maven版本不低于3.0

如下图所示



并把 datax 文件夹下对应的jar包替换掉 如下图所示

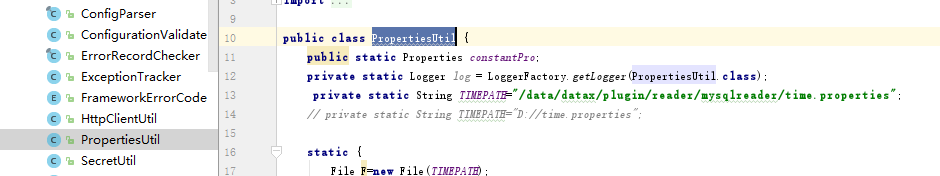




### 3.维护time.properties

代码中维护时间的配置文件路径 为**/data/datax/plugin/reader/mysqlreader/time.properties**

**所在类为 模块 datax-core 下的** com.alibaba.datax.core.util.PropertiesUtil



设置增量任务是需要 在对应的目录下新建配置文件time.properties

路径必须完全一致 不需要维护key值 代码会自动生成键值和时间 库名/表名/lasttime和库名/表名/secondtime，

库名/表名/lasttime为最后一次时间 库名/表名/secondtime为上传时间

系统以库名/表名/lasttime为参照时间

如果没有对应的键值 默认第一次库名/表名/lasttime是0 系统会全量加载

每次执行完job会替换lasttime为当前时间

注意：key值按照 库名/表名 的规则生成

如果有不同服务器上相同的数据库名和表名设置定时增量任务的极端情况要避免 会造成参照时间错乱

替换完jar包 定义好job.json 维护完time.properties就可以设置定时增量任务了

定时任务设置参考 六、定时任务