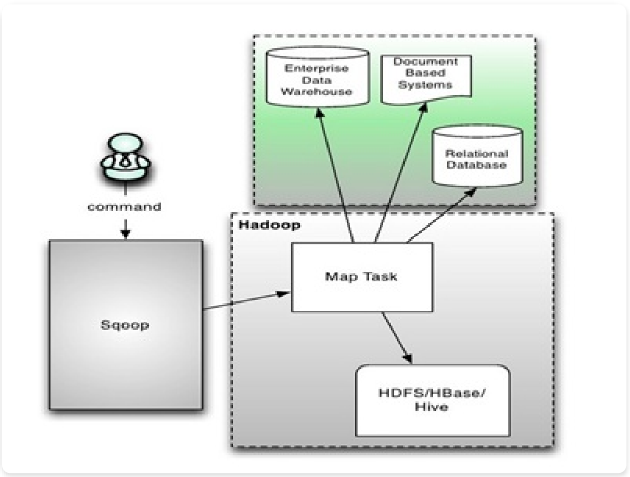
Day22 sqoop

hadoop sqoop

sqoop

> sqoop是Apache顶级项目，主要用来在Hadoop和关系数据库中传递数据。通过sqoop，我们可以方便的将数据从关系数据库导入到HDFS，或者将数据从HDFS导出到关系数据库。

sqoop架构



sqoop架构示意图

Sqoop工具接收到客户端的shell命令或者Java api命令后，通过Sqoop中的任务翻译器(Task Translator)将命令转换为对应的MapReduce任务，而后将关系型数据库和Hadoop中的数据进行相互转移，进而完成数据的拷贝。

sqoop安装

1. 解压 `tar -zxvf sqoop-1.99.7-bin-hadoop200.tar.gz`
2. 修改hadoop中的core-site.xml

```
<property>
  <name>hadoop.proxyuser.sqoop2.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.sqoop2.groups</name>
  <value>*</value>
</property>
```

3. 在【/opt/software/sqoop/sqoop-1.99.7-bin-hadoop200/conf】下修改sqoop.properties

```
# Hadoop configuration directory
org.apache.sqoop.submission.engine.mapreduce.configuration.directory=/opt/software/hadoop/hadoop
-2.7.4/etc/hadoop
```

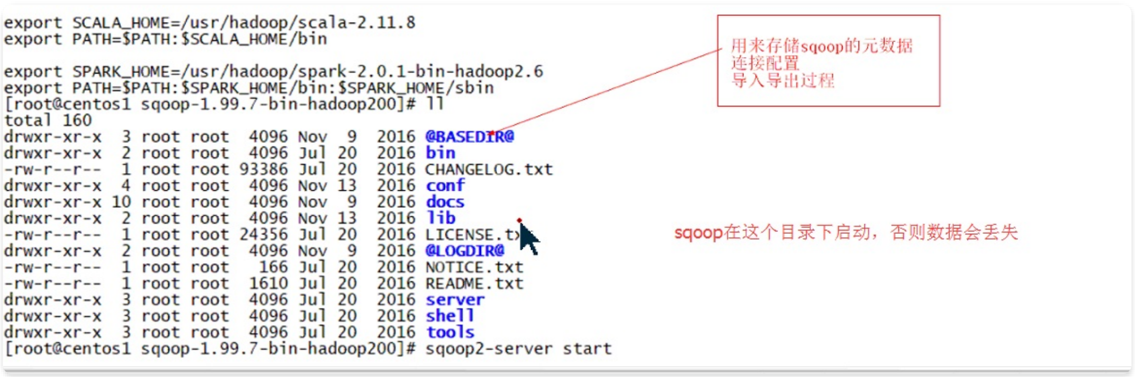
4. 配置环境变量

```
# set sqoop enviroment

export SQOOP_HOME=/opt/software/sqoop/sqoop-1.99.7-bin-hadoop200
# set JDBC drivers to this directory
export SQOOP_SERVER_EXTRA_LIB=$SQOOP_HOME/extra
export PATH=$PATH:$SQOOP_HOME/bin
export LOGDIR=$SQOOP_HOME/logs
export BASEDIR=$SQOOP_HOME/base
```

5. 配置sqoop加载的驱动的文件夹，也可以不配置，因为它默认加载lib下的jar，我们只需要将jar放在lib下面就可以了

```
export SQOOP_SERVER_EXTRA_LIB=/var/lib/sqoop2/
```

6. 初始化sqoop `sqoop2-tool upgrade`
7. 检查是否初始化成功 `sqoop2-tool verify`
8. 启动sqoop服务 `sqoop2-server start` 关闭sqoop服务 `sqoop2-server stop`
9. 连接client `sqoop2-shell`

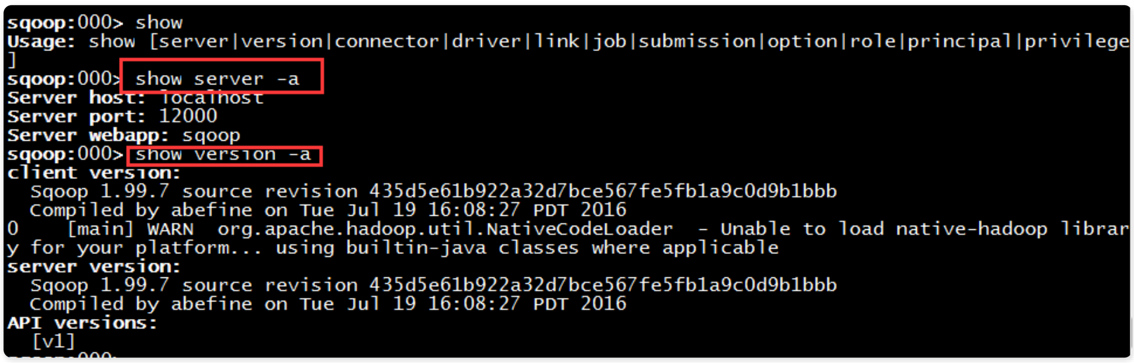


```
export SCALA_HOME=/usr/hadoop/scala-2.11.8
export PATH=$PATH:$SCALA_HOME/bin

export SPARK_HOME=/usr/hadoop/spark-2.0.1-bin-hadoop2.6
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
[root@centos1 sqoop-1.99.7-bin-hadoop200]# ll
total 160
drwxr-xr-x  3 root root  4096 Nov  9  2016 @BASEDIR@
drwxr-xr-x  2 root root  4096 Jul 20  2016 bin
-rw-r--r--  1 root root 93386 Jul 20  2016 CHANGELOG.txt
drwxr-xr-x  4 root root  4096 Nov 13  2016 conf
drwxr-xr-x 10 root root  4096 Nov  9  2016 docs
drwxr-xr-x  2 root root  4096 Nov 13  2016 lib
-rw-r--r--  1 root root 24356 Jul 20  2016 LICENSE.tx
drwxr-xr-x  2 root root  4096 Nov  9  2016 @LOGDIR@
-rw-r--r--  1 root root   166 Jul 20  2016 NOTICE.txt
-rw-r--r--  1 root root  1610 Jul 20  2016 README.txt
drwxr-xr-x  3 root root  4096 Jul 20  2016 server
drwxr-xr-x  3 root root  4096 Jul 20  2016 shell
drwxr-xr-x  3 root root  4096 Jul 20  2016 tools
[root@centos1 sqoop-1.99.7-bin-hadoop200]# sqoop2-server start
```

sqoop下的object

基本信息

server
version

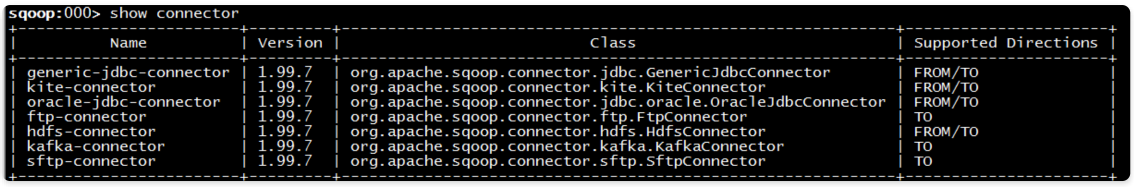


```
sqoop:000> show
Usage: show [server|version|connector|driver|link|job|submission|option|role|principal|privilege
]
sqoop:000> show server -a
Server host: localhost
Server port: 12000
Server webapp: sqoop
sqoop:000> show version -a
client version:
  Sqoop 1.99.7 source revision 435d5e61b922a32d7bce567fe5fb1a9c0d9b1bbb
  Compiled by abefine on Tue Jul 19 16:08:27 PDT 2016
0    [main] WARN  org.apache.hadoop.util.NativeCodeLoader  - Unable to load native-hadoop librar
y for your platform... using builtin-java classes where applicable
server version:
  Sqoop 1.99.7 source revision 435d5e61b922a32d7bce567fe5fb1a9c0d9b1bbb
  Compiled by abefine on Tue Jul 19 16:08:27 PDT 2016
API versions:
  [v1]
```

核心对象

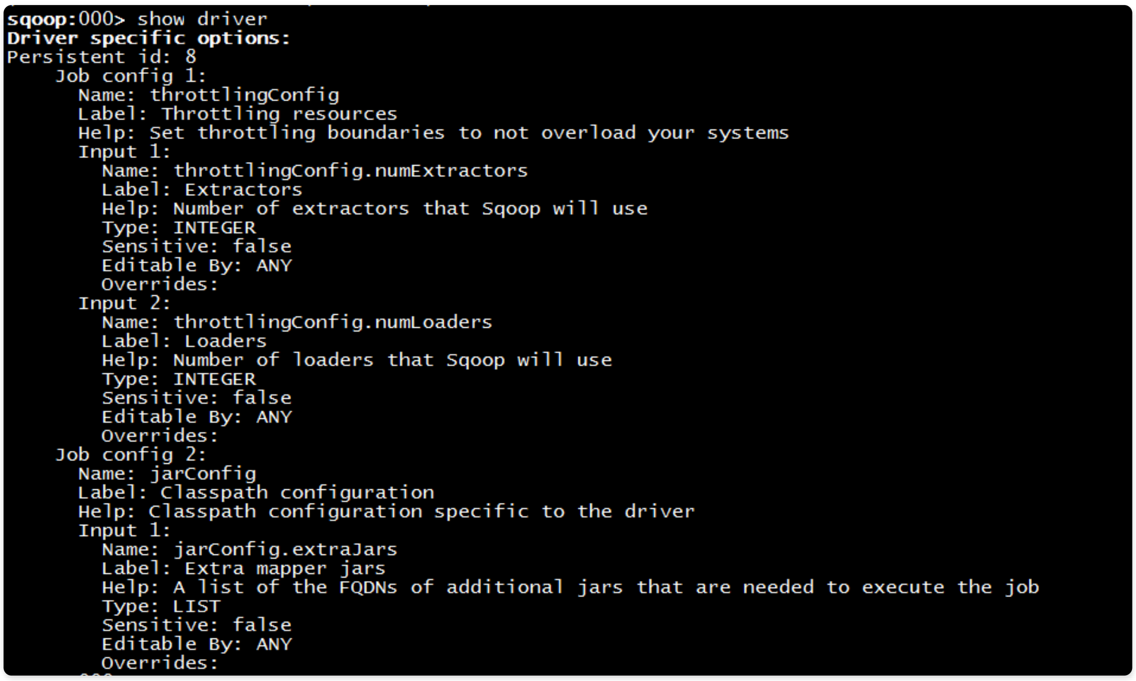
connector

> connector是sqoop当前支持的存储系统连接配置类型
> connector Name是sqoop默认支持的数据连接类型
> Supported Direction中`From/to`表示连接方式，From表示数据来源(导入)，To表示数据去向(导出)



```
sqoop:000> show connector
+------------------------+---------+------------------------------------------------------------+----------------------+
|          Name          | Version |                           Class                            | Supported Directions |
+------------------------+---------+------------------------------------------------------------+----------------------+
| generic-jdbc-connector | 1.99.7  | org.apache.sqoop.connector.jdbc.GenericJdbcConnector       | FROM/TO              |
| kite-connector         | 1.99.7  | org.apache.sqoop.connector.kite.KiteConnector              | FROM/TO              |
| oracle-jdbc-connector  | 1.99.7  | org.apache.sqoop.connector.jdbc.oracle.OracleJdbcConnector | FROM/TO              |
| ftp-connector          | 1.99.7  | org.apache.sqoop.connector.ftp.FtpConnector                | TO                   |
| hdfs-connector         | 1.99.7  | org.apache.sqoop.connector.hdfs.HdfsConnector              | FROM/TO              |
| kafka-connector        | 1.99.7  | org.apache.sqoop.connector.kafka.KafkaConnector            | TO                   |
| sftp-connector         | 1.99.7  | org.apache.sqoop.connector.sftp.SftpConnector              | TO                   |
+------------------------+---------+------------------------------------------------------------+----------------------+
```

driver： 驱动配置信息，在此查看



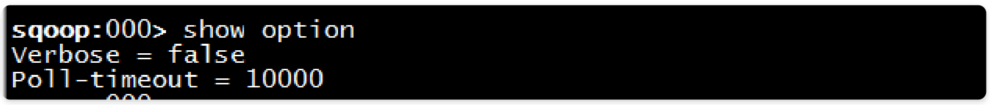
```
sqoop:000> show driver
Driver specific options:
Persistent id: 8
    Job config 1:
      Name: throttlingConfig
      Label: Throttling resources
      Help: Set throttling boundaries to not overload your systems
      Input 1:
        Name: throttlingConfig.numExtractors
        Label: Extractors
        Help: Number of extractors that Sqoop will use
        Type: INTEGER
        Sensitive: false
        Editable By: ANY
        Overrides:
      Input 2:
        Name: throttlingConfig.numLoaders
        Label: Loaders
        Help: Number of loaders that Sqoop will use
        Type: INTEGER
        Sensitive: false
        Editable By: ANY
        Overrides:
    Job config 2:
      Name: jarConfig
      Label: Classpath configuration
      Help: Classpath configuration specific to the driver
      Input 1:
        Name: jarConfig.extraJars
        Label: Extra mapper jars
        Help: A list of the FQDNs of additional jars that are needed to execute the job
        Type: LIST
        Sensitive: false
        Editable By: ANY
        Overrides:
```

link、job数据导入导出配置对象
link：配置具体的存储连接，他是以connecter作为类型的
比如某个jdbc数据库的连接，某个hdfs集群的连接等等
job 配置一次导入导出过程的全部细节信息，它是以link作为输入输出的，通常用`from link1 to link2` 表示把link1中的数据导入到link2中
导出数据的具体制定在job里面配置

submission：查看当前已提交的sqoop导入导出任务

参数信息

option



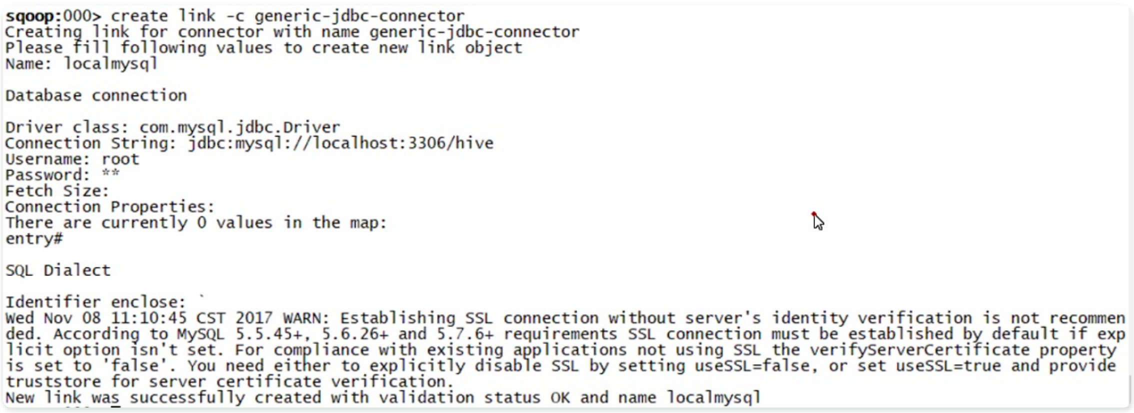
```
sqoop:000> show option
Verbose = false
Poll-timeout = 10000
```

权限信息

role
principal|
privilege

将mysql中的数据导入到hadoop平台

创建mysql link

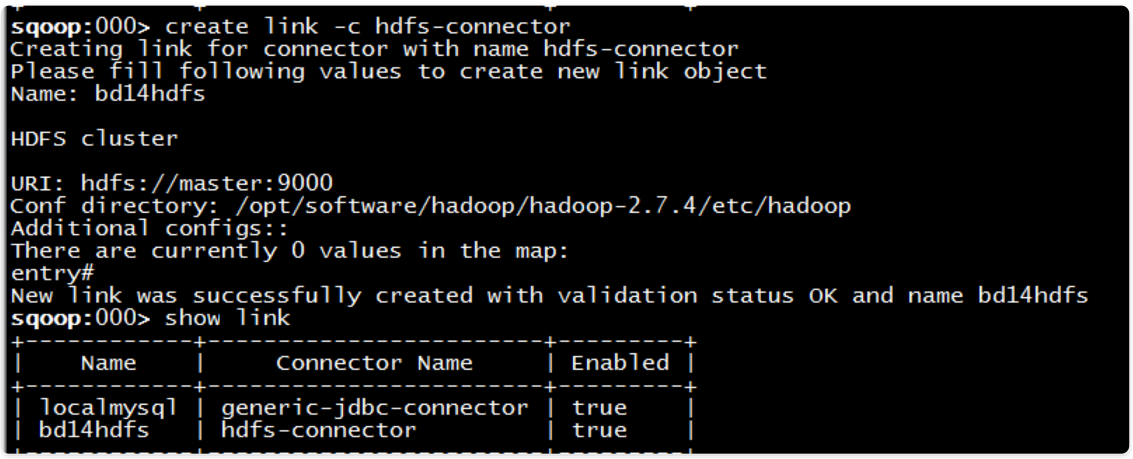


```
sqoop:000> create link -c generic-jdbc-connector
Creating link for connector with name generic-jdbc-connector
Please fill following values to create new link object
Name: localmysql

Database connection

Driver class: com.mysql.jdbc.Driver
Connection String: jdbc:mysql://localhost:3306/hive
Username: root
Password: **
Fetch Size:
Connection Properties:
There are currently 0 values in the map:
entry#

SQL Dialect

Identifier enclose: `
Wed Nov 08 11:10:45 CST 2017 WARN: Establishing SSL connection without server's identity verification is not recommen
ded. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if exp
licit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property
is set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide
truststore for server certificate verification.
New link was successfully created with validation status OK and name localmysql
```

将connection String的值改为jdbc:mysql://master:3306/hive,否则会报错

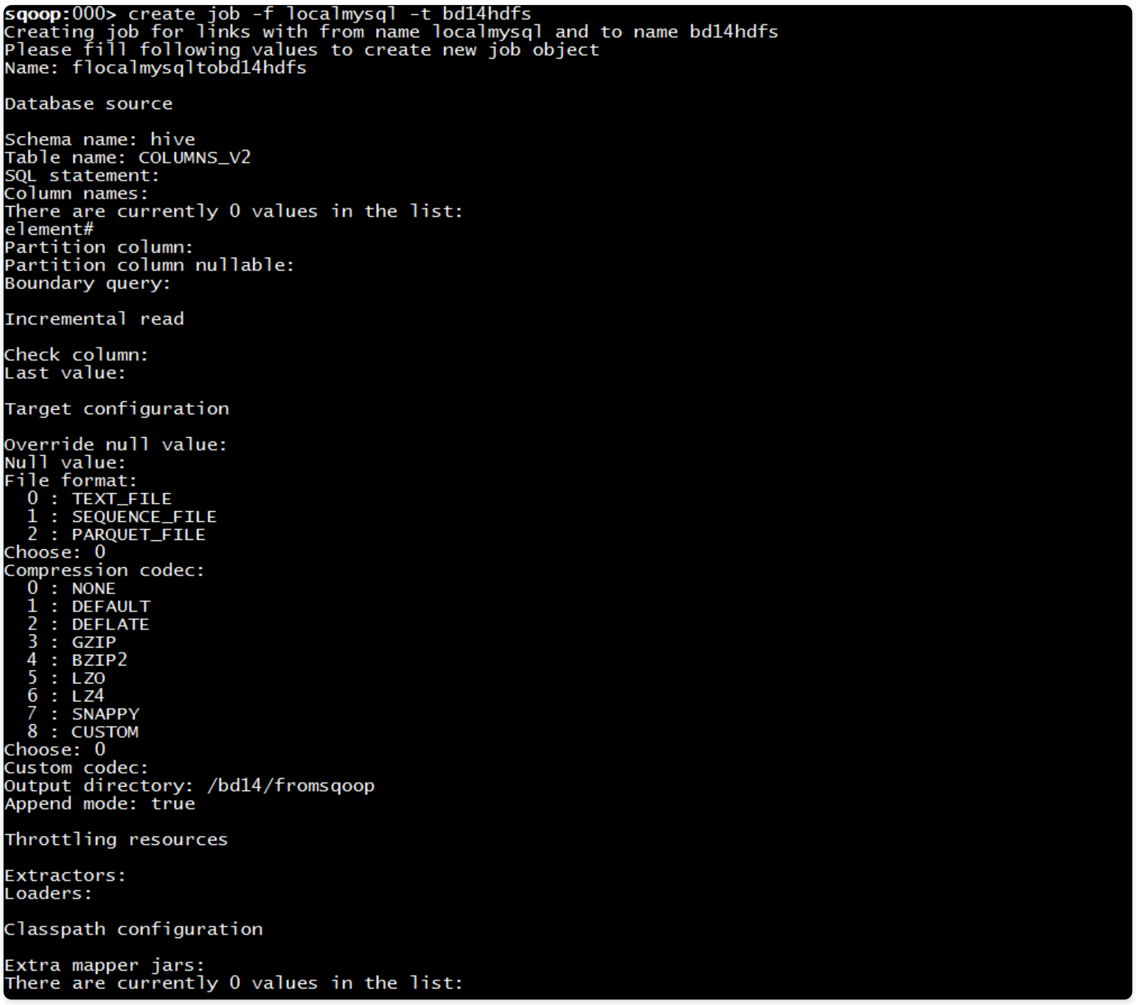
查看link `show link`
connection Properties：数据库的配置参数，可以不写
Identifier enclose：标识符的封装符号，mysql中使用反引号作为标识符 `` ` ``，sqoop中默认的是逗号。

创建hdfs link



```
sqoop:000> create link -c hdfs-connector
Creating link for connector with name hdfs-connector
Please fill following values to create new link object
Name: bd14hdfs

HDFS cluster

URI: hdfs://master:9000
Conf directory: /opt/software/hadoop/hadoop-2.7.4/etc/hadoop
Additional configs::
There are currently 0 values in the map:
entry#
New link was successfully created with validation status OK and name bd14hdfs
sqoop:000> show link
+------------+------------------------+---------+
|    Name    |     Connector Name     | Enabled |
+------------+------------------------+---------+
| localmysql | generic-jdbc-connector | true    |
| bd14hdfs   | hdfs-connector         | true    |
```

创建job



```
sqoop:000> create job -f localmysql -t bd14hdfs
Creating job for links with from name localmysql and to name bd14hdfs
Please fill following values to create new job object
Name: flocalmysqltobd14hdfs

Database source

Schema name: hive
Table name: COLUMNS_V2
SQL statement:
Column names:
There are currently 0 values in the list:
element#
Partition column:
Partition column nullable:
Boundary query:

Incremental read

Check column:
Last value:

Target configuration

Override null value:
Null value:
File format:
  0 : TEXT_FILE
  1 : SEQUENCE_FILE
  2 : PARQUET_FILE
Choose: 0
Compression codec:
  0 : NONE
  1 : DEFAULT
  2 : DEFLATE
  3 : GZIP
  4 : BZIP2
  5 : LZO
  6 : LZ4
  7 : SNAPPY
  8 : CUSTOM
Choose: 0
Custom codec:
Output directory: /bd14/fromsqoop
Append mode: true

Throttling resources

Extractors:
Loaders:

Classpath configuration

Extra mapper jars:
There are currently 0 values in the list:
```

查看job `show job`

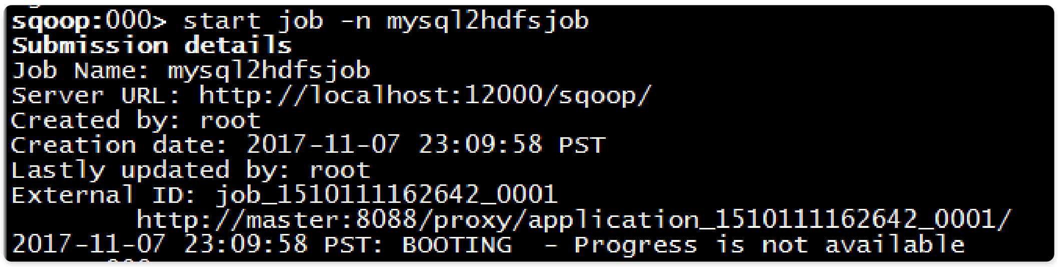
查看提交的状态信息，需要用到jobhistory服务，下面是启动过程
启动jobhistory

```
mr-jobhistory-daemon.sh start historyserver
```

update link -n localmysql
show link -n localmysql

启动sqoop job

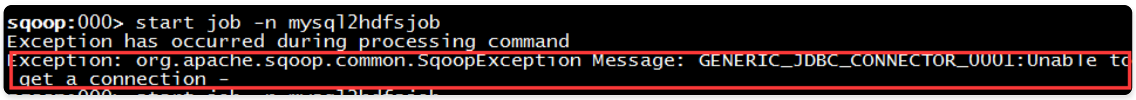
启动命令 `start job -n mysql2hdfsjob`
启动成功界面



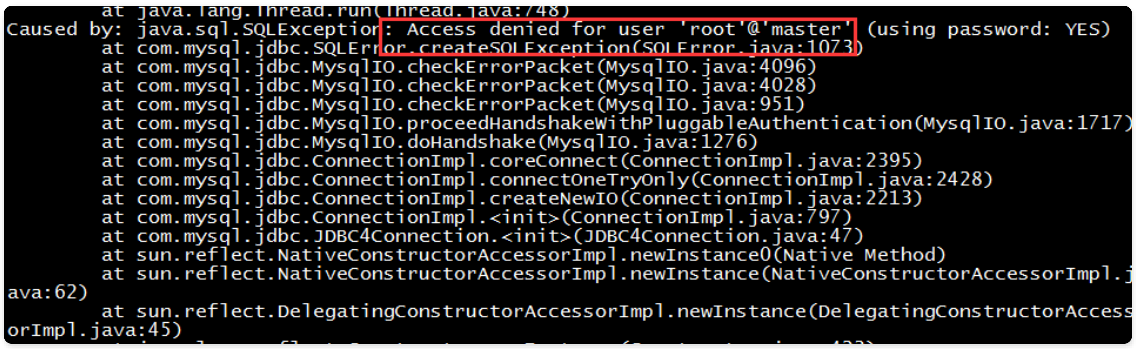
```
sqoop:000> start job -n mysql2hdfsjob
Submission details
Job Name: mysql2hdfsjob
Server URL: http://localhost:12000/sqoop/
Created by: root
Creation date: 2017-11-07 23:09:58 PST
Lastly updated by: root
External ID: job_1510111162642_0001
        http://master:8088/proxy/application_1510111162642_0001/
2017-11-07 23:09:58 PST: BOOTING  - Progress is not available
```

启动异常

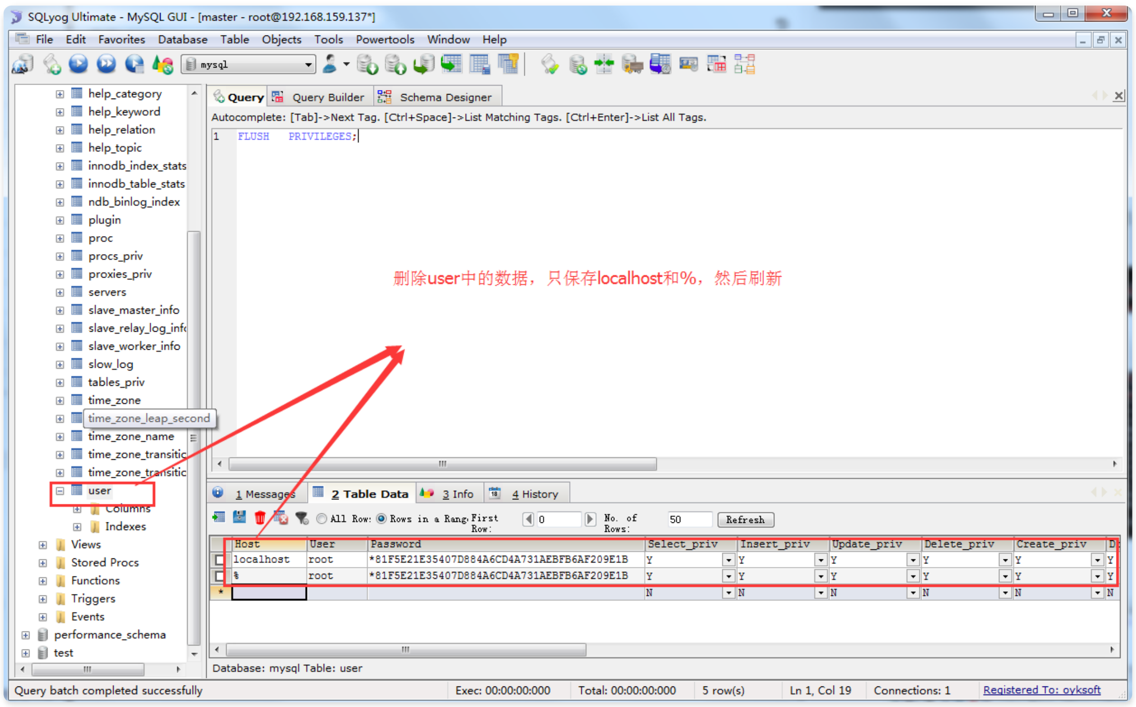
启动job时，出现链接不上的现象，查看日志，看是否是权限问题



```
sqoop:000> start job -n mysql2hdfsjob
Exception has occurred during processing command
Exception: org.apache.sqoop.common.SqoopException Message: GENERIC_JDBC_CONNECTOR_0001:Unable to
 get a connection -
```



```
        at java.lang.Thread.run(Thread.java:748)
Caused by: java.sql.SQLException: Access denied for user 'root'@'master' (using password: YES)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1073)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4096)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:4028)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:951)
        at com.mysql.jdbc.MysqlIO.proceedHandshakeWithPluggableAuthentication(MysqlIO.java:1717)
        at com.mysql.jdbc.MysqlIO.doHandshake(MysqlIO.java:1276)
        at com.mysql.jdbc.ConnectionImpl.coreConnect(ConnectionImpl.java:2395)
        at com.mysql.jdbc.ConnectionImpl.connectOneTryOnly(ConnectionImpl.java:2428)
        at com.mysql.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:2213)
        at com.mysql.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:797)
        at com.mysql.jdbc.JDBC4Connection.<init>(JDBC4Connection.java:47)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.j
ava:62)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccess
orImpl.java:45)
```



```
1  FLUSH  PRIVILEGES;
```

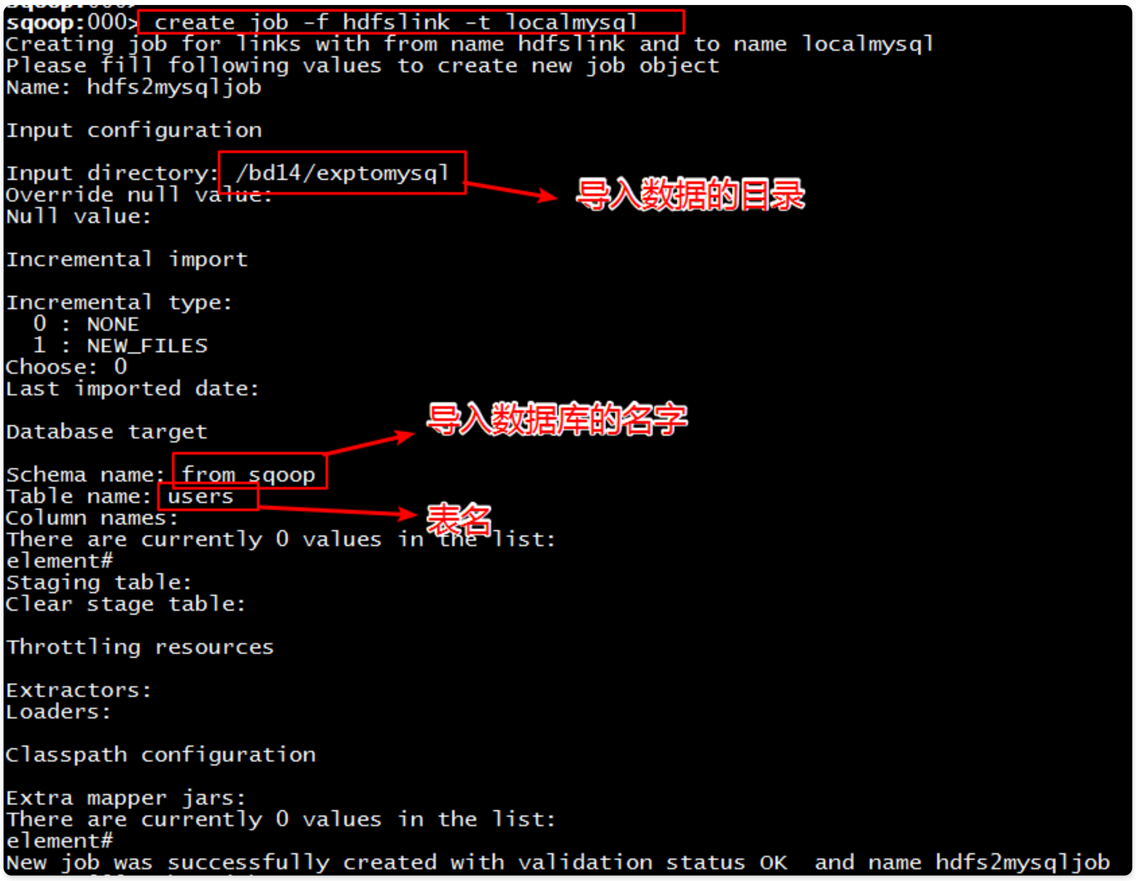
删除user中的数据，只保存localhost和%，然后刷新

Host	User	Password	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv
localhost	root	*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B	Y	Y	Y	Y	Y
%	root	*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B	Y	Y	Y	Y	Y

Database: mysql Table: user

Query batch completed successfully　Exec: 00:00:00:000　Total: 00:00:00:000　5 row(s)　Ln 1, Col 19　Connections: 1

将hdfs上的数据导入到mysql数据库

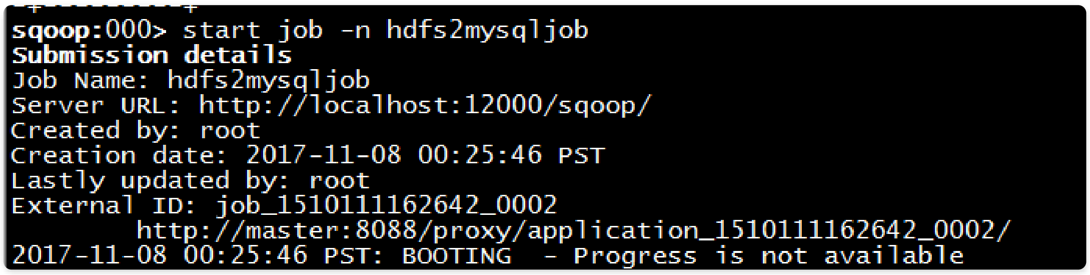


```
sqoop:000> create job -f hdfslink -t localmysql
Creating job for links with from name hdfslink and to name localmysql
Please fill following values to create new job object
Name: hdfs2mysqljob

Input configuration

Input directory: /bd14/exptomysql
Override null value:
Null value:

Incremental import

Incremental type:
  0 : NONE
  1 : NEW_FILES
Choose: 0
Last imported date:

Database target

Schema name: from_sqoop
Table name: users
Column names:
There are currently 0 values in the list:
element#
Staging table:
Clear stage table:

Throttling resources

Extractors:
Loaders:

Classpath configuration

Extra mapper jars:
There are currently 0 values in the list:
element#
New job was successfully created with validation status OK  and name hdfs2mysqljob
```

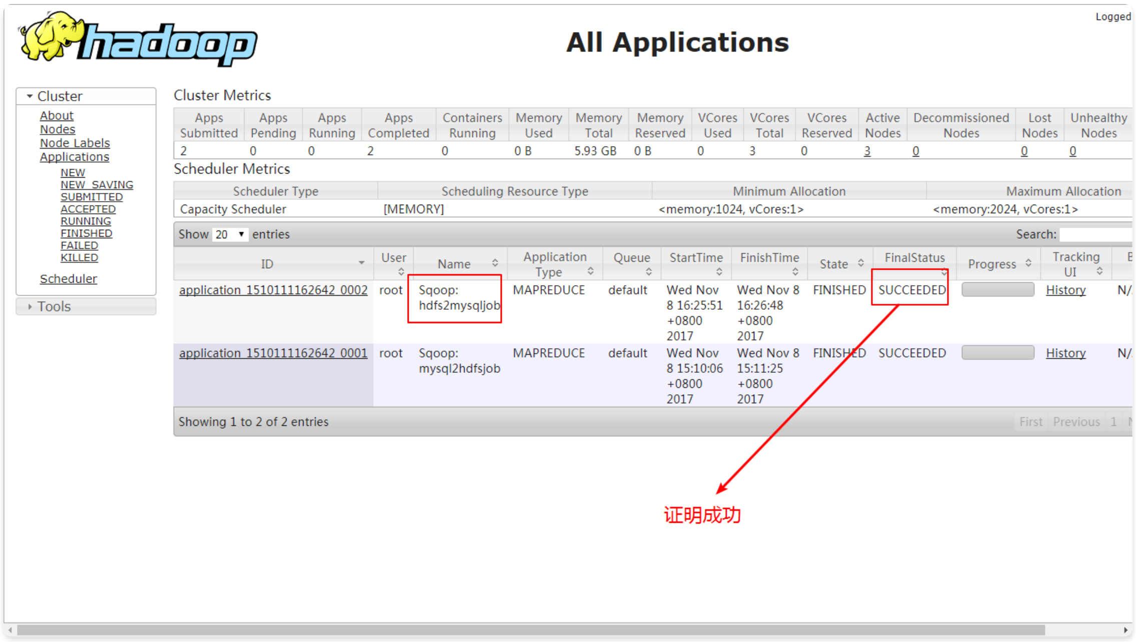
我们在hdfs上创建文件导入目录文件夹 `hdfs dfs -mkdir /bd14/exptomysql`
将数据放入到此文件夹内，就可以完成导入操作了，但是到目前为止sqoop只支持csv格式的文件导入导出，因此，我们需要将数据转换成csv格式，再放入到目录下
在mysql中创建数据库 `create database from_sqoop` 和表

```
create table users(
    id integer,
    name varchar(30),
    age integer
    );
```

启动任务 `start job -n hdfs2mysqljob`
出现如下图示，说明成功



```
sqoop:000> start job -n hdfs2mysqljob
Submission details
Job Name: hdfs2mysqljob
Server URL: http://localhost:12000/sqoop/
Created by: root
Creation date: 2017-11-08 00:25:46 PST
Lastly updated by: root
External ID: job_1510111162642_0002
        http://master:8088/proxy/application_1510111162642_0002/
2017-11-08 00:25:46 PST: BOOTING  - Progress is not available
```


Logged

All Applications

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
2	0	0	2	0	0 B	5.93 GB	0 B	0	3	0	3	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:2024, vCores:1>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	B
application_1510111162642_0002	root	Sqoop: hdfs2mysqljob	MAPREDUCE	default	Wed Nov 8 16:25:51 +0800 2017	Wed Nov 8 16:26:48 +0800 2017	FINISHED	SUCCEEDED		History	N/
application_1510111162642_0001	root	Sqoop: mysql2hdfsjob	MAPREDUCE	default	Wed Nov 8 15:10:06 +0800 2017	Wed Nov 8 15:11:25 +0800 2017	FINISHED	SUCCEEDED		History	N/

Showing 1 to 2 of 2 entries First Previous 1

java API 操作sqoop

创建link

```
// 创建一个link
 public void createLink() {
  MLink link = client.createLink("generic-jdbc-connector");
  link.setName("window_mysql");
  // link.getConnectorLinkConfig()获取connector的link配置信息
  MLinkConfig linkConfig = link.getConnectorLinkConfig();

  // 取出所有的配置项
  /*List<MConfig> list = linkConfig.getConfigs();
  for (MConfig mConfig : list) {
   List<MInput<?>> inputs = mConfig.getInputs();
   for (MInput<?> input : inputs) {
    System.out.println(input);
   }
  }*/

  // MLinkConfig相关配置项名称设置配置项
  linkConfig.getStringInput("linkConfig.jdbcDriver").setValue("com.mysql.jdbc.Driver")
;
  linkConfig.getStringInput("linkConfig.connectionString").setValue("jdbc:mysql://192.
168.6.81:3306/test");
  linkConfig.getStringInput("linkConfig.username").setValue("root");
  linkConfig.getStringInput("linkConfig.password").setValue("root");
  linkConfig.getStringInput("dialect.identifierEnclose").setValue(" ");

  Status status = client.saveLink(link);
  if (status.canProceed()) {
   System.out.println("创建link " + link.getName() + "成功");
  } else {
   System.out.println("创建link " + link.getName() + "失败,请检查配置项");
  }
 }
```

创建job

```
public void createJob(){
  MJob job = client.createJob("hdfslink", "window_mysql");
  job.setName("hdfs2_Window");

  MFromConfig fromjobConfig = job.getFromJobConfig();
  MToConfig toJobConfig = job.getToJobConfig();

  // 列举出配置项信息
  /*List<MConfig> configs = fromjobConfig.getConfigs();
  for (MConfig mConfig : configs) {
   List<MInput<?>> inputs = mConfig.getInputs();
   for (MInput<?> mInput : inputs) {
    System.out.println(mInput);
   }
  }*/

  fromjobConfig.getStringInput("fromJobConfig.inputDirectory").setValue("/bd14/exptomy
sql");

  // 列举出配置项信息
  /*System.out.println("----------------");
  List<MConfig> configs2 = toJobConfig.getConfigs();
  for (MConfig mConfig : configs2) {
   List<MInput<?>> inputs = mConfig.getInputs();
   for (MInput<?> mInput : inputs) {
    System.out.println(mInput);
   }
  }*/

  toJobConfig.getStringInput("toJobConfig.schemaName").setValue("xs");
  toJobConfig.getStringInput("toJobConfig.tableName").setValue("users");

  Status status = client.saveJob(job);
  if(status.canProceed()){
   System.out.println("创建job " + job.getName() + "成功");
  }else{
   System.out.println("创建job " + job.getName() + "失败, 请检查配置");
  }
 }
```

启动job

```
package top.xiesen.sqoopcleint;

import java.util.List;

import org.apache.sqoop.client.SqoopClient;
import org.apache.sqoop.model.MConfig;
import org.apache.sqoop.model.MFromConfig;
import org.apache.sqoop.model.MInput;
import org.apache.sqoop.model.MJob;
import org.apache.sqoop.model.MLink;
import org.apache.sqoop.model.MLinkConfig;
import org.apache.sqoop.model.MToConfig;
import org.apache.sqoop.validation.Status;

public class SqoopTest {
 // sqoop的服务端url地址
 private final String URL = "http://master:12000/sqoop/";
 // 创建客户端对象
 private SqoopClient client = new SqoopClient(URL);
 // 创建一个link
 public void createLink() {
  MLink link = client.createLink("generic-jdbc-connector");
  link.setName("window_mysql");
  // link.getConnectorLinkConfig()获取connector的link配置信息
  MLinkConfig linkConfig = link.getConnectorLinkConfig();

  // 取出所有的配置项
  /*List<MConfig> list = linkConfig.getConfigs();
  for (MConfig mConfig : list) {
   List<MInput<?>> inputs = mConfig.getInputs();
   for (MInput<?> input : inputs) {
    System.out.println(input);
   }
  }*/

  // MLinkConfig相关配置项名称设置配置项
  linkConfig.getStringInput("linkConfig.jdbcDriver").setValue("com.mysql.jdbc.Driver")
;
  linkConfig.getStringInput("linkConfig.connectionString").setValue("jdbc:mysql://192.
168.6.81:3306/test");
  linkConfig.getStringInput("linkConfig.username").setValue("root");
  linkConfig.getStringInput("linkConfig.password").setValue("root");
  linkConfig.getStringInput("dialect.identifierEnclose").setValue(" ");

  Status status = client.saveLink(link);
  if (status.canProceed()) {
   System.out.println("创建link " + link.getName() + "成功");
  } else {
   System.out.println("创建link " + link.getName() + "失败,请检查配置项");
  }
 }


 public void createJob(){
  MJob job = client.createJob("hdfslink", "window_mysql");
  job.setName("hdfs2_Window");

  MFromConfig fromjobConfig = job.getFromJobConfig();
  MToConfig toJobConfig = job.getToJobConfig();

  // 列举出配置项信息
  /*List<MConfig> configs = fromjobConfig.getConfigs();
  for (MConfig mConfig : configs) {
   List<MInput<?>> inputs = mConfig.getInputs();
   for (MInput<?> mInput : inputs) {
    System.out.println(mInput);
   }
  }*/

  fromjobConfig.getStringInput("fromJobConfig.inputDirectory").setValue("/bd14/exptomy
sql");

  // 列举出配置项信息
  /*System.out.println("----------------");
  List<MConfig> configs2 = toJobConfig.getConfigs();
  for (MConfig mConfig : configs2) {
   List<MInput<?>> inputs = mConfig.getInputs();
   for (MInput<?> mInput : inputs) {
    System.out.println(mInput);
   }
  }*/

  toJobConfig.getStringInput("toJobConfig.schemaName").setValue("xs");
  toJobConfig.getStringInput("toJobConfig.tableName").setValue("users");

  Status status = client.saveJob(job);
  if(status.canProceed()){
   System.out.println("创建job " + job.getName() + "成功");
   }else{
    System.out.println("创建job " + job.getName() + "失败, 请检查配置");
   }
  }

 public static void main(String[] args) {
  SqoopTest st = new SqoopTest();
//  st.createLink();
//  st.createJob();
  // 启动job
  st.client.startJob("hdfs2_Window");
 }
}
```