

**数学与信息学院学生实验报告**

**实验课程名称：** 大数据技术概论 **教师：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | 熟悉常用的HBase操作 | | | **实验成绩** |  |
| **学生姓名** |  | **学 号** | **40** | **年级专业班级** |  |
| **小组成员** | **无** | | | **实验日期** |  |

# 实验目的和要求

* 理解HBase在Hadoop体系结构中的角色；
* 熟练使用HBase操作常用的Shell命令；
* 熟悉HBase操作常用的Java API。

## 1.1 实验目的

Hadoop运行在Linux系统上，因此，需要学习实践一些常用的Linux命令。本实验旨在熟悉常用的Linux操作和Hadoop操作，为顺利开展后续其他实验奠定基础。

## 1.2 实验软硬件环境

* 操作系统：Linux（建议Ubuntu18.04）；
* Hadoop版本：3.2.1；
* HBase版本：2.1.1；
* JDK版本：1.11；
* Java IDE：Eclipse 4.3。

## 1.3 实验要求

① 在Linux系统上安装Hadoop，记录软件安装、测试的过程及结果。

② 熟悉常用的Linux操作，实现并记录30项操作的过程及结果。

③ 熟悉常用的Hadoop操作，实现并记录4项操作的过程及结果。

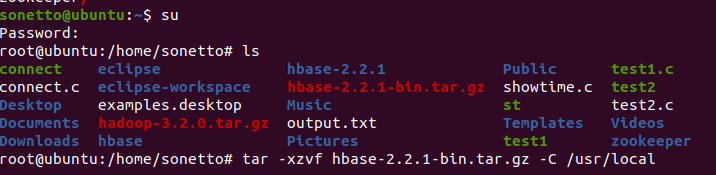
# 2. 实验记录

（〇）HBase的配置步骤：

1. 单机版hbase安装

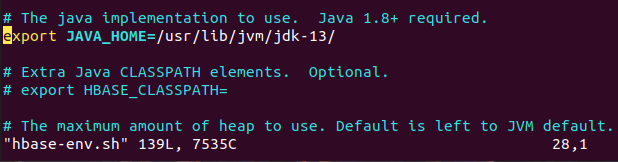
(注:如果直接安装伪分布式，会报错，还是要按照apache给的文档一步一步做下去！)

·解压到/usr/local



·在/usr/local/hbase-2.2.1/conf修改文件hbase-env.sh

（注：不推荐把hbase-2.2.1改名成hbase，可能会冲突的）



·修改/usr/local/hbase-2.2.1/conf中的hbase-site.xml（没截图）

<configuration>

<property>

<name>hbase.rootdir</name>

<value>file:///home/sonetto/hbase</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/home/sonetto/zookeeper</value>

</property>

<property>

<name>hbase.unsafe.stream.capability.enforce</name>

<value>false</value>

<description>

Controls whether HBase will check for stream capabilities (hflush/hsync).

Disable this if you intend to run on LocalFileSystem, denoted by a rootdir

with the 'file://' scheme, but be mindful of the NOTE below.

WARNING: Setting this to false blinds you to potential data loss and

inconsistent system state in the event of process and/or node failures. If

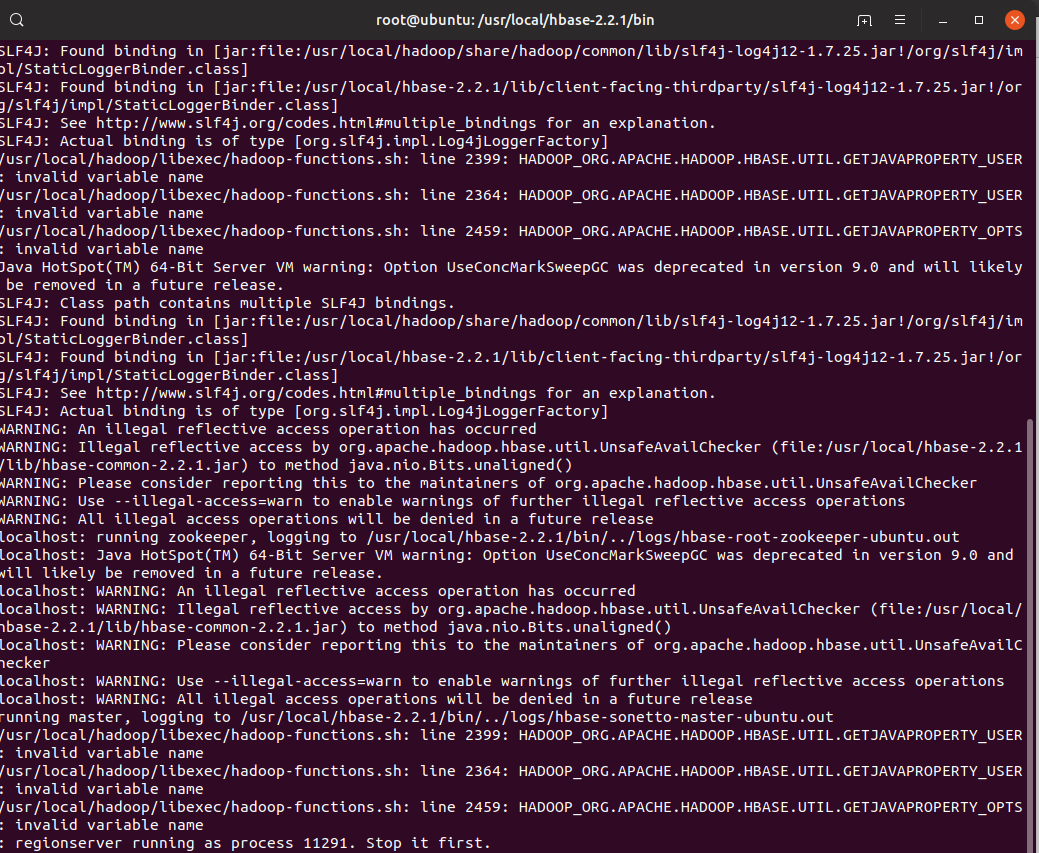
HBase is complaining of an inability to use hsync or hflush it's most

likely not a false positive.

</description>

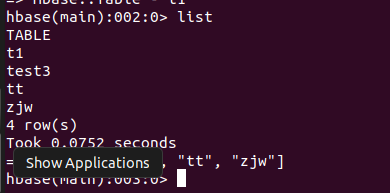
</property></configuration>

·启动./start-hbase.sh，成功之后应该有HMaster节点



有可能会出这么多warning，但是问题不大。

·进入./bin/hbase shell

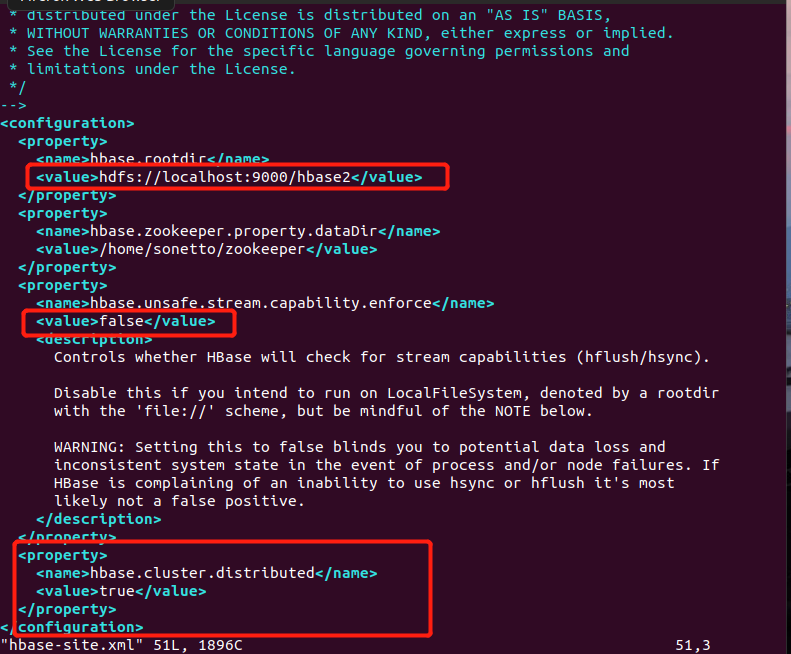


随便尝试一个命令，如果可以成功，那就完成了单机的安装。

1. 伪分布式配置

·从单机版EXIT，（可以按ctrl+z）然后stop-hbase.sh

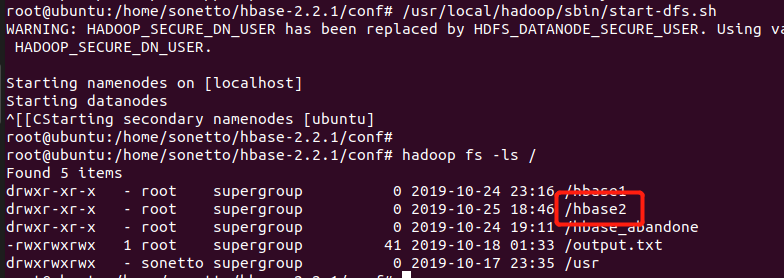
·修改/usr/local/hbase-2.2.1/conf中的hbase-site.xml



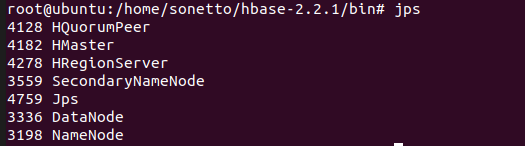
修改的是这几句，虽然hbase教程上说可以直接把

删掉，可能是我的水平太差，删掉了就没有hmaster了。

·打开hadoop，查看fs上是否有/hbase的文件夹

因为我的是装了不止一次，之前的崩掉了，所以是2

·start-hbase.sh,再执行jps，此时



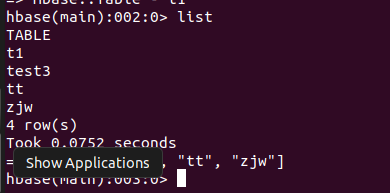
三个hadoop结点，HMASTER，HREGIONSERVER都在。

恭喜恭喜！！！伪分布式安装完成，此时就可以进入hbase shell进行快乐的编程了！！

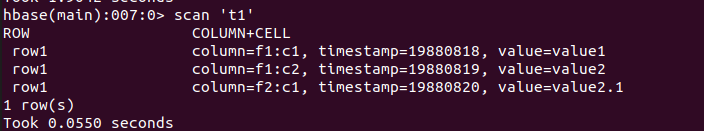
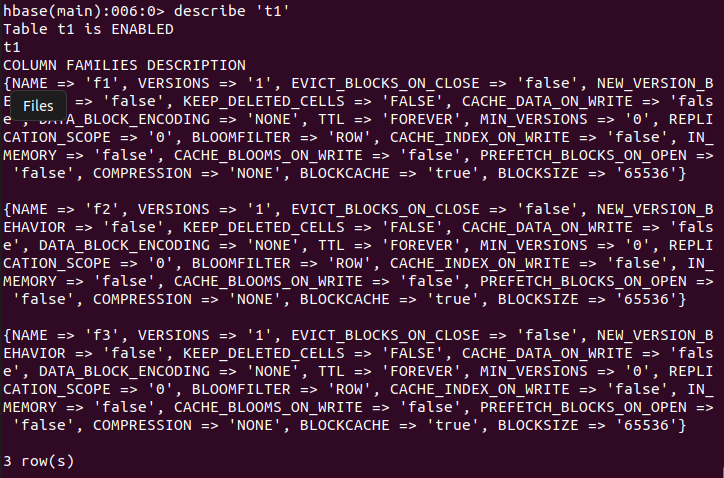
注：apache后面提供的master backup和regionserver指令在接下来的编程中并无实际应用。

（一）用Hadoop提供的HBase Shell命令完成相同任务：

1. 列出HBase所有的表的相关信息，例如表名；

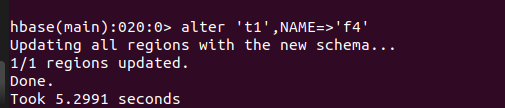


1. 在终端打印出指定的表的所有记录数据；

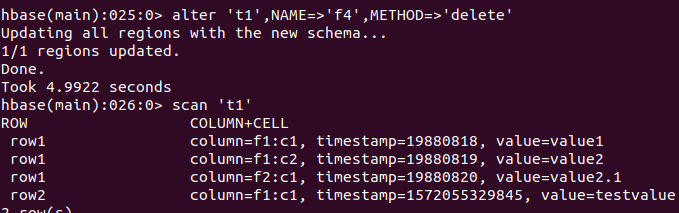


1. 向已经创建好的表添加和删除指定的列族或列；

添加列族：



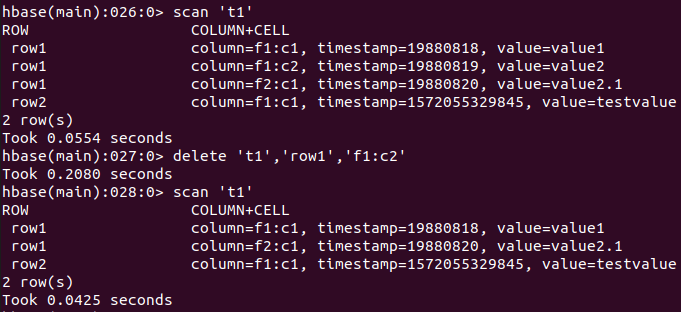
删除列族：



添加列：

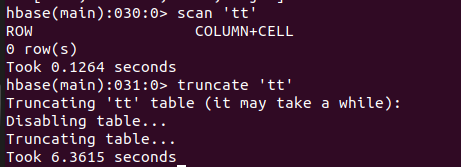


删除列：

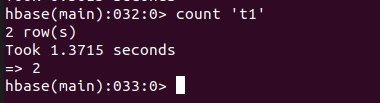


注：如果要删除表，首先disable，再drop

1. 清空指定的表的所有记录数据；



1. 统计表的行数。

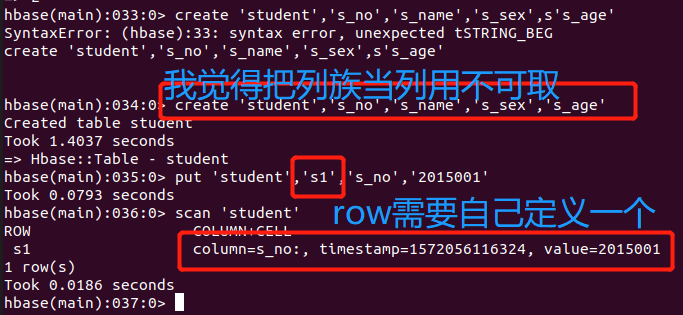


（二）HBase数据库操作

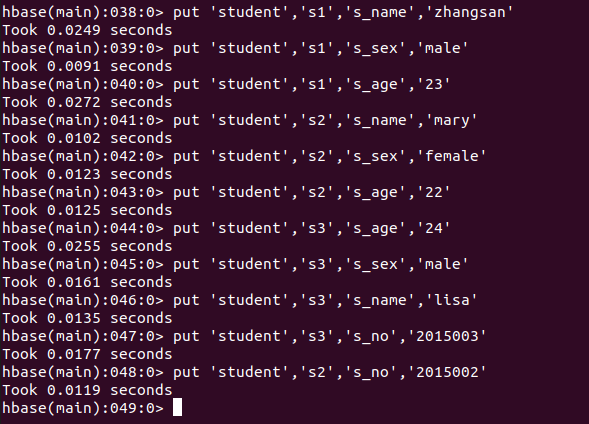
1. 现有以下关系型数据库中的表和数据，要求将其转换为适合于HBase存储的表并插入数据：

学生表（Student）

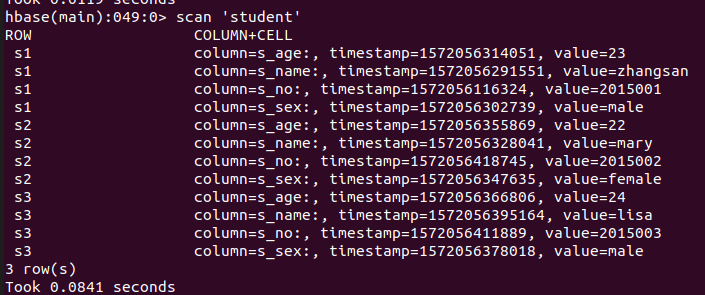
|  |  |  |  |
| --- | --- | --- | --- |
| 学号（S\_No） | 姓名（S\_Name） | 性别（S\_Sex） | 年龄（S\_Age） |
| 2015001 | Zhangsan | male | 23 |
| 2015003 | Mary | female | 22 |
| 2015003 | Lisi | male | 24 |



···四维确定一个value！···

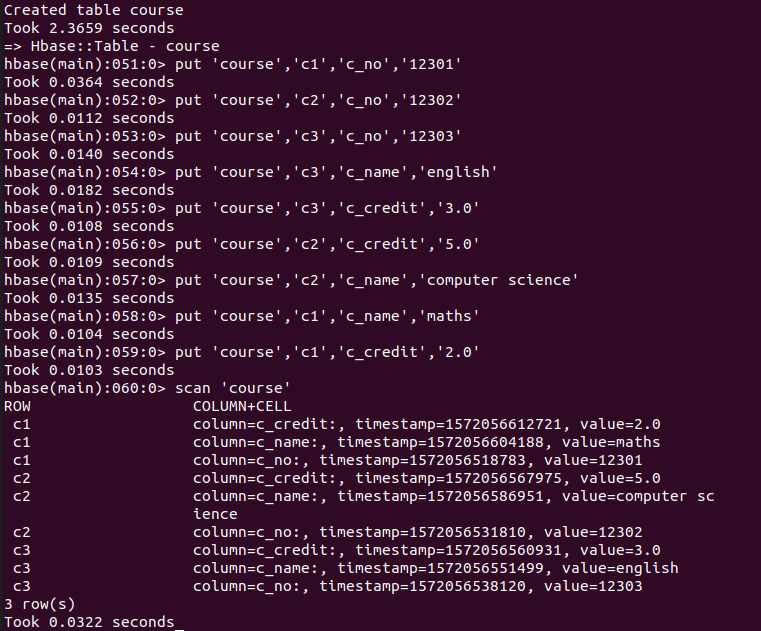


检查一下：



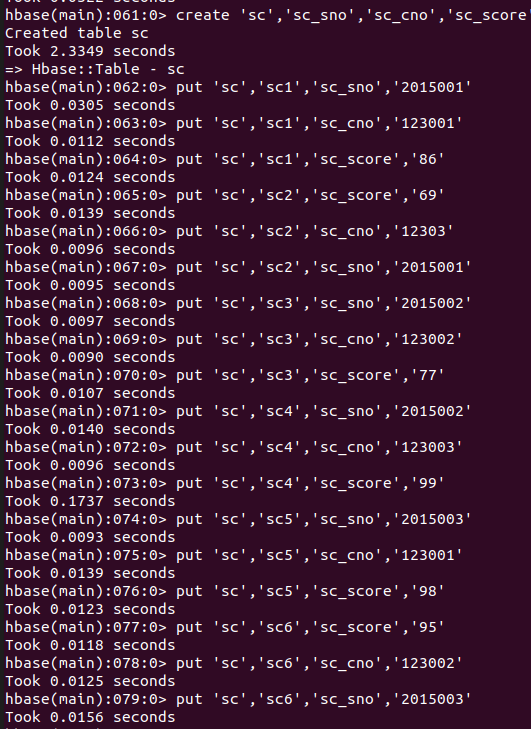
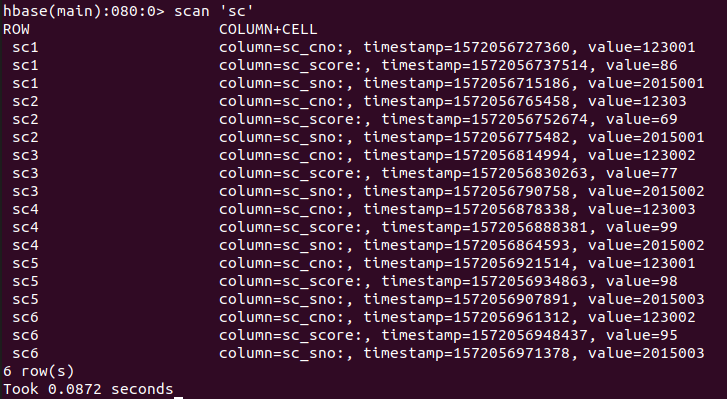
课程表（Course）

|  |  |  |
| --- | --- | --- |
| 课程号（C\_No） | 课程名（C\_Name） | 学分（C\_Credit） |
| 123001 | Math | 2.0 |
| 123002 | Computer Science | 5.0 |
| 123003 | English | 3.0 |



选课表（SC）

|  |  |  |
| --- | --- | --- |
| 学号（SC\_Sno） | 课程号（SC\_Cno） | 成绩（SC\_Score） |
| 2015001 | 123001 | 86 |
| 2015001 | 123003 | 69 |
| 2015002 | 123002 | 77 |
| 2015002 | 123003 | 99 |
| 2015003 | 123001 | 98 |
| 2015003 | 123002 | 95 |

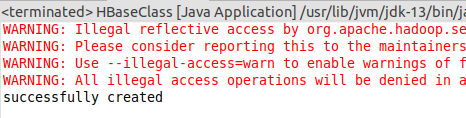
2. 请编程实现以下功能：

（1）createTable(String tableName, String[] fields)

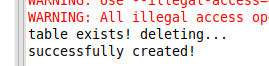
创建表，参数tableName为表的名称，字符串数组fields为存储记录各个字段名称的数组。要求当HBase已经存在名为tableName的表的时候，先删除原有的表，然后再创建新的表。

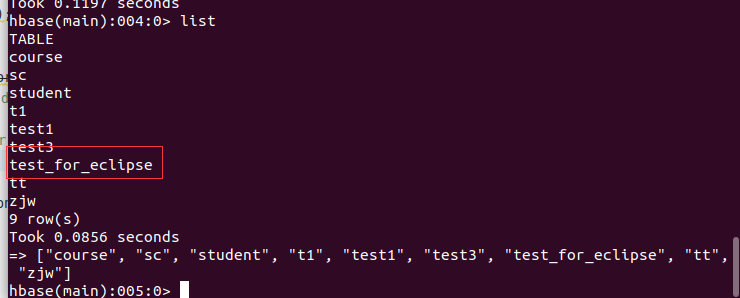
·结果：

·第一次执行



·第二次执行





·注：这次直接创建了一个HBaseClass类，把所有函数都写在里面了，所以代码都是一段一段的。

·主函数：

**public** **static** **void** main(String[] args) **throws** Exception {

*init*();

String[] columns= {"aa","bb"};

String tableName="test\_for\_eclipse";

*createTable*(tableName,columns);

*close*();

}

·其他函数、变量：

**public** **static** Configuration *conf*;

**public** **static** Connection *conn*;

**public** **static** Admin *admin*;

//

**public** **static** **void** init()

{

*conf* = HBaseConfiguration.*create*(); //配置管理信息

*conf*.set("hbase.rootdir", "hdfs://localhost:9000/hbase2");

**try** {

*conn* = ConnectionFactory.*createConnection*(*conf*); //通过管理信息，建立连接

*admin* = *conn*.getAdmin(); //admin是对数据库进行管理的类，用于管理表的创建删除

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//close a connection

**public** **static** **void** close() **throws** IOException

{

**if**(*admin*!=**null**)

{

*admin*.close();

}

**if**(*conn*!=**null**)

{

*conn*.close();

}

}

//create a new table

**static** **void** createTable(String tableName, String[] fields)**throws** IOException

{

TableName name = TableName.*valueOf*(tableName); //table's name

**if**(*admin*.tableExists(name)) //如果存在，删除

{

*admin*.disableTable(name);

*admin*.deleteTable(name);

System.***out***.println("table exists! deleting...");

}

//创建

~~HTableDescriptor~~ hTableDescriptor = **new** ~~HTableDescriptor~~(name);//将 表名 存入table descriptor

**for**(String str:fields)//put col\_family

{

~~HColumnDescriptor~~ hColumnDescriptor = **new** ~~HColumnDescriptor~~(str); //将 列族 依次存入 column descriptor

hTableDescriptor.~~addFamily~~(hColumnDescriptor); //将column descriptor存入 table descriptor

}

*admin*.createTable(hTableDescriptor);//admin添加table descriptor，创建表

System.***out***.println("successfully created!");

}

**public** **static** **void** deleteTable(String tableName) **throws** IOException

{

TableName name = TableName.*valueOf*(tableName);

**if**(*admin*.tableExists(name))

{

*admin*.deleteTable(name);

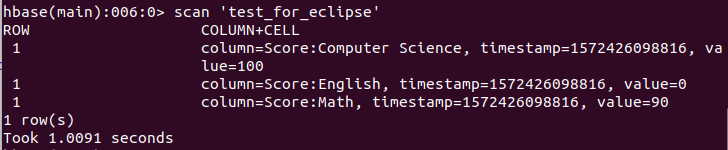
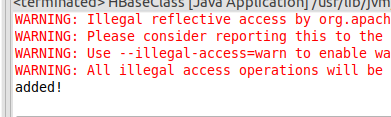
}

}

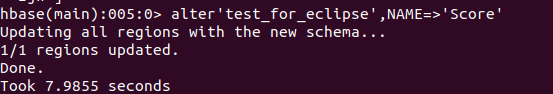
（2）addRecord(String tableName, String row, String[] fields, String[] values)

向表tableName、行row（用S\_Name表示）和字符串数组fields指定的单元格中添加对应的数据values。其中，fields中每个元素如果对应的列族下还有相应的列限定符的话，用“columnFamily:column”表示。例如，同时向“Math”、“Computer Science”、“English”三列添加成绩时，字符串数组fields为{“Score:Math”, ”Score:Computer Science”, ”Score:English”}，数组values存储这三门课的成绩。

·结果：



·事先创建好列族score



·主函数：

**public** **static** **void** main(String[] args) **throws** Exception {

*init*();

String row="1";

String[] columns={"Score:Math", "Score:Computer Science", "Score:English"};

String[] values= {"90","100","0"};

String tableName="test\_for\_eclipse";

*addRecord*(tableName, row, columns, values);

*close*();

}

·其他函数、变量：

**public** **static** Configuration *conf*;

**public** **static** Connection *conn*;

**public** **static** Admin *admin*;

//

**public** **static** **void** init()

{

*conf* = HBaseConfiguration.*create*(); //配置管理信息

*conf*.set("hbase.rootdir", "hdfs://localhost:9000/hbase2");

**try** {

*conn* = ConnectionFactory.*createConnection*(*conf*); //通过管理信息，建立连接

*admin* = *conn*.getAdmin(); //admin是对数据库进行管理的类，用于管理表的创建删除

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//close a connection

**public** **static** **void** close() **throws** IOException

{

**if**(*admin*!=**null**)

{

*admin*.close();

}

**if**(*conn*!=**null**)

{

*conn*.close();

}

}

//添加数据

**public** **static** **void** addRecord(String tableName,String row,String[] fields,String[] values) **throws** IOException

{

Table table = *conn*.getTable(TableName.*valueOf*(tableName));

Put put = **new** Put(row.getBytes()); //根据行键，找到put行

**for**(**int** i=0;i<fields.length;i++)

{

String[] col=fields[i].split(":");

put.addColumn(col[0].getBytes(), col[1].getBytes(), values[i].getBytes());//col[0]:col\_family col[1]:qulifier

table.put(put);

}

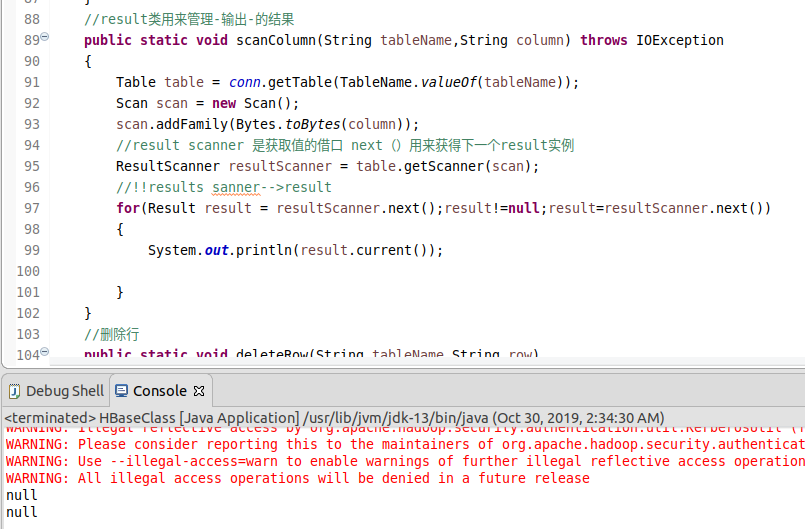
System.***out***.println("added!");

table.close();

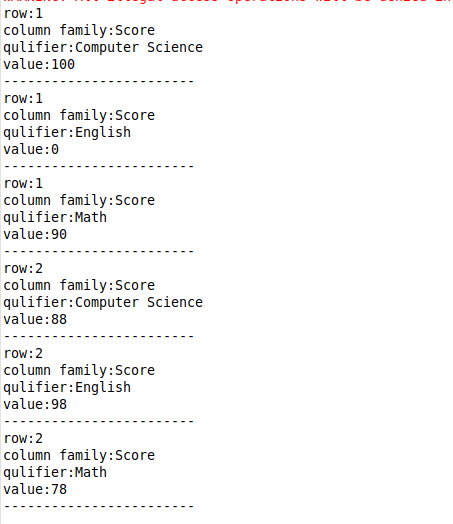
}

（3）scanColumn(String tableName, String column)

浏览表tableName某一列的数据，如果某一行记录中该列数据不存在，则返回null。要求当参数column为某一列族名称时，如果底下有若干个列限定符，则要列出每个列限定符代表的列的数据；当参数column为某一列具体名称（例如“Score:Math”）时，只需要列出该列的数据。

我太天真了

·结果：



主函数：

**public** **static** **void** main(String[] args) **throws** Exception {

*init*();

String columns="Score";

String tableName="test\_for\_eclipse";

*scanColumn*(tableName, columns);

*close*();

}

其他变量、函数：

**public** **static** Configuration *conf*;

**public** **static** Connection *conn*;

**public** **static** Admin *admin*;

//

**public** **static** **void** init()

{

*conf* = HBaseConfiguration.*create*(); //配置管理信息

*conf*.set("hbase.rootdir", "hdfs://localhost:9000/hbase2");

**try** {

*conn* = ConnectionFactory.*createConnection*(*conf*); //通过管理信息，建立连接

*admin* = *conn*.getAdmin(); //admin是对数据库进行管理的类，用于管理表的创建删除

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//close a connection

**public** **static** **void** close() **throws** IOException

{

**if**(*admin*!=**null**)

{

*admin*.close();

}

**if**(*conn*!=**null**)

{

*conn*.close();

}

}

//result类用来管理-输出-的结果

**public** **static** **void** scanColumn(String tableName,String column) **throws** IOException

{

Table table = *conn*.getTable(TableName.*valueOf*(tableName));

Scan scan = **new** Scan();

scan.addFamily(Bytes.*toBytes*(column));

//result scanner 是获取值的借口 next（）用来获得下一个result实例

ResultScanner resultScanner = table.getScanner(scan);

//!!results sanner-->result

**for**(Result result = resultScanner.next();result!=**null**;result=resultScanner.next())

{

Cell[] cell=result.rawCells();

**for**(Cell c:cell)

{

System.***out***.println("row:"+**new** String(CellUtil.*cloneRow*(c)));

System.***out***.println("column family:"+**new** String(CellUtil.*cloneFamily*(c)));

System.***out***.println("qulifier:"+**new** String(CellUtil.*cloneQualifier*(c)));

System.***out***.println("value:"+**new** String(CellUtil.*cloneValue*(c)));

System.***out***.println("------------------------");

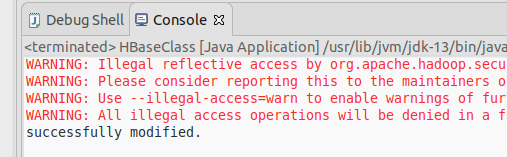
}

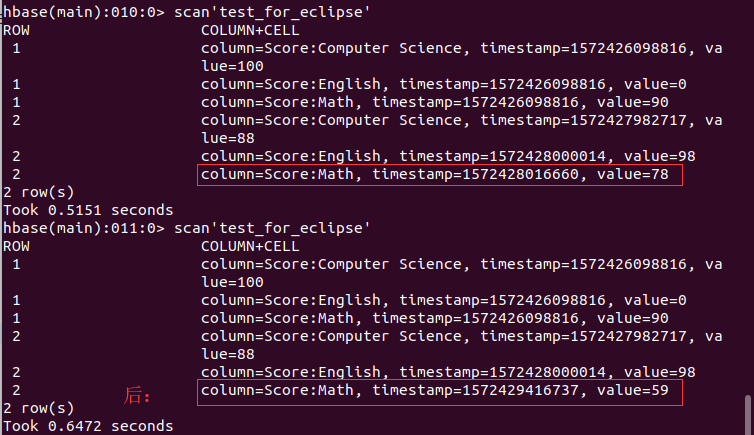
}

（4）modifyData(String tableName, String row, String column)

修改表tableName，行row（可以用学生姓名S\_Name表示），列column指定的单元格的数据。

·结果：





·主函数：

**public** **static** **void** main(String[] args) **throws** Exception {

*init*();

String row="2";

String tableName="test\_for\_eclipse";

String field="Score:Math";

String value="59";

*modifyData*(tableName, row, field, value);

*close*();

}

·其他函数、变量：

**public** **static** Configuration *conf*;

**public** **static** Connection *conn*;

**public** **static** Admin *admin*;

//

**public** **static** **void** init()

{

*conf* = HBaseConfiguration.*create*(); //配置管理信息

*conf*.set("hbase.rootdir", "hdfs://localhost:9000/hbase2");

**try** {

*conn* = ConnectionFactory.*createConnection*(*conf*); //通过管理信息，建立连接

*admin* = *conn*.getAdmin(); //admin是对数据库进行管理的类，用于管理表的创建删除

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//close a connection

**public** **static** **void** close() **throws** IOException

{

**if**(*admin*!=**null**)

{

*admin*.close();

}

**if**(*conn*!=**null**)

{

*conn*.close();

}

}

//修改数据

**public** **static** **void** modifyData(String tableName,String row,String field,String value) **throws** IOException

{

Table table = *conn*.getTable(TableName.*valueOf*(tableName));

Put put = **new** Put(row.getBytes()); //根据行键，找到put行

String[] col=field.split(":");

put.addColumn(col[0].getBytes(), col[1].getBytes(), value.getBytes());

table.put(put);

System.***out***.println("successfully modified.");

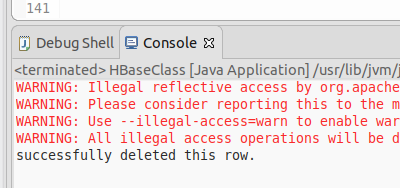
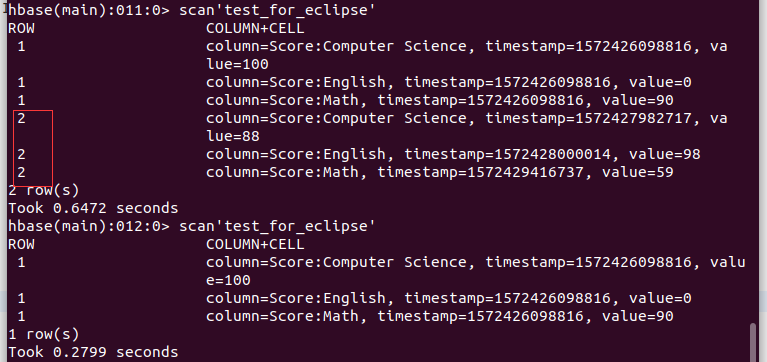
table.close();

}

（5）deleteRow(String tableName, String row)

删除表tableName中row指定的行的记录。

·结果



·主函数：

**public** **static** **void** main(String[] args) **throws** Exception {

*init*();

String row="2";

String tableName="test\_for\_eclipse";

*deleteRow*(tableName, row);

*close*();

}

·其他函数、变量：

**public** **static** Configuration *conf*;

**public** **static** Connection *conn*;

**public** **static** Admin *admin*;

//

**public** **static** **void** init()

{

*conf* = HBaseConfiguration.*create*(); //配置管理信息

*conf*.set("hbase.rootdir", "hdfs://localhost:9000/hbase2");

**try** {

*conn* = ConnectionFactory.*createConnection*(*conf*); //通过管理信息，建立连接

*admin* = *conn*.getAdmin(); //admin是对数据库进行管理的类，用于管理表的创建删除

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

//close a connection

**public** **static** **void** close() **throws** IOException

{

**if**(*admin*!=**null**)

{

*admin*.close();

}

**if**(*conn*!=**null**)

{

*conn*.close();

}

}

//删除行

**public** **static** **void** deleteRow(String tableName,String row) **throws** IOException

{

Table table = *conn*.getTable(TableName.*valueOf*(tableName));

Delete del = **new** Delete(row.getBytes());

table.delete(del);

System.***out***.println("successfully deleted this row.");

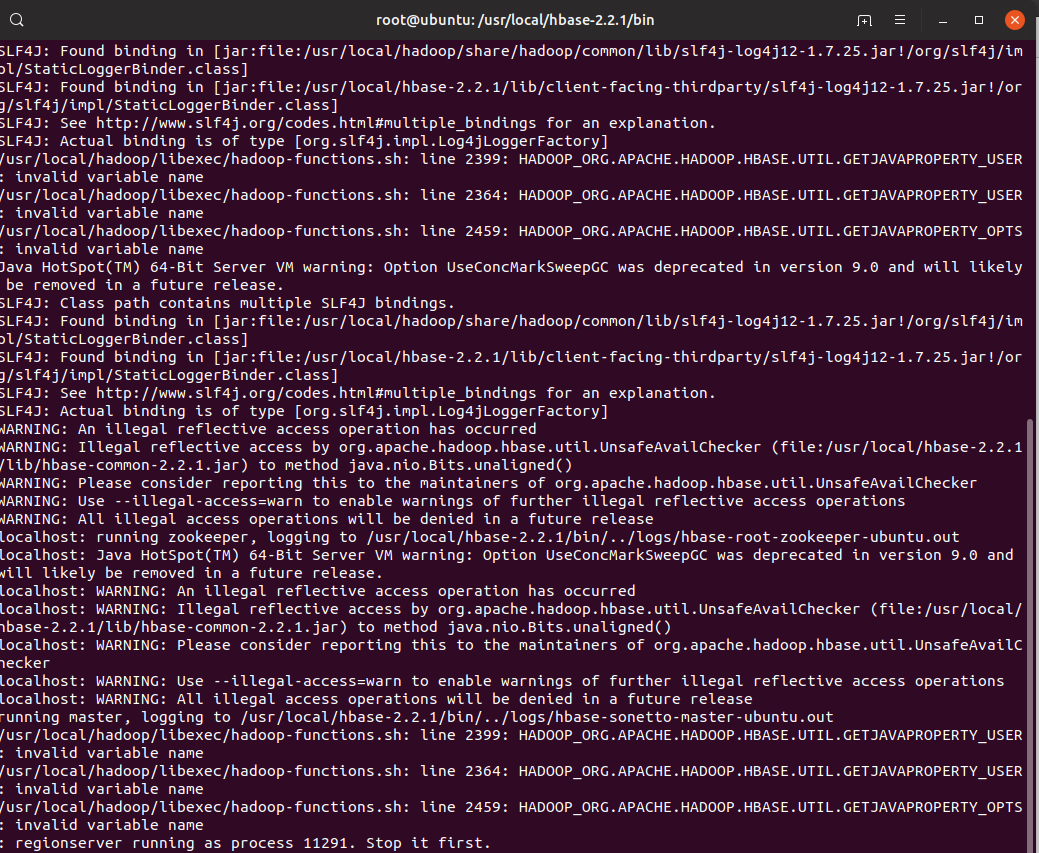
table.close();

}

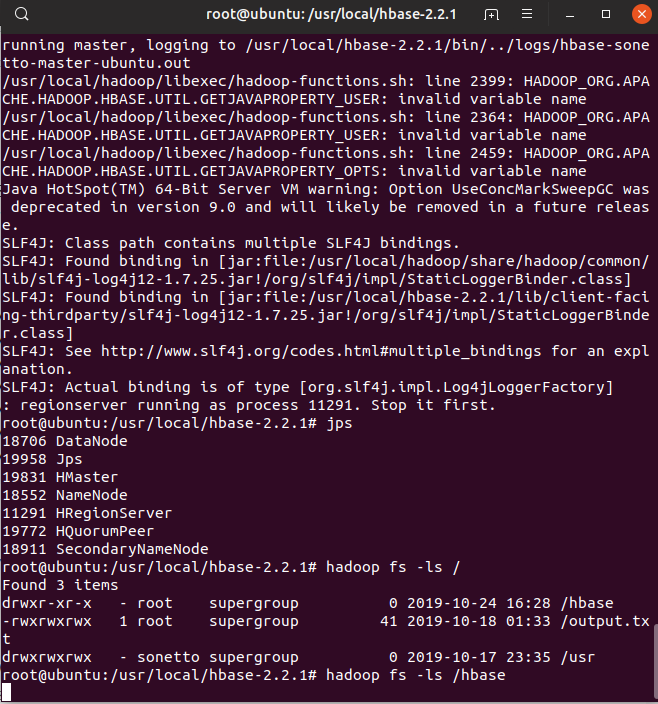
# 实验总结

·虽然有row\_key,但是row\_key只能帮助定位到数据..跟关系型数据库很不一样

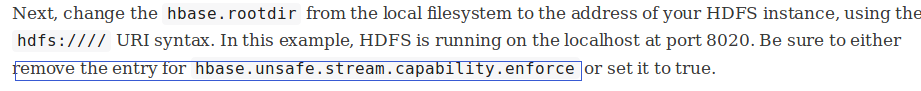
·列族、列限定符需要注意函数的使用



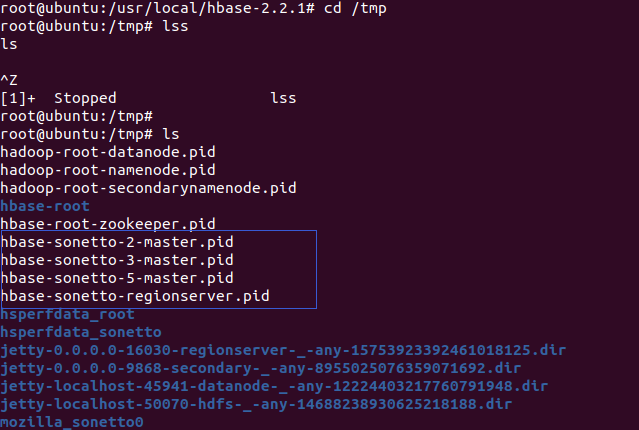
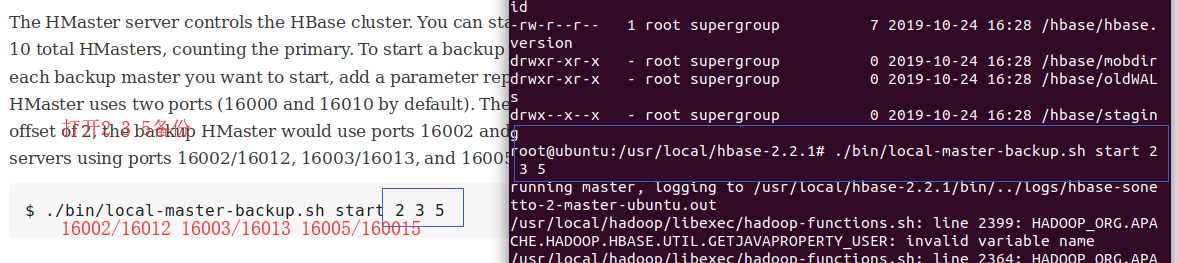
虽然hbase有很多的warning和日志冲突什么的问题，但是运行起来暂时还能用，百度了也没有解答的，先这样吧，船到桥头自然沉。



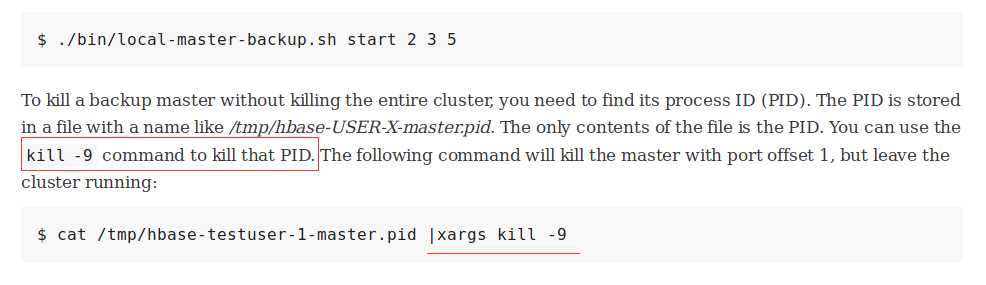
本来没有这个hmaster，因为官方教程上说：



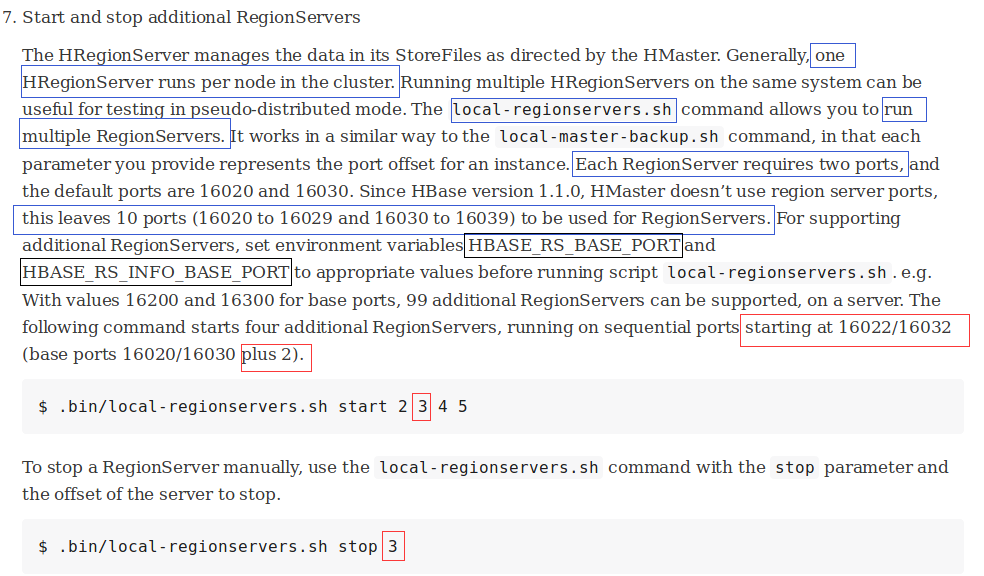
我说怕错，干脆把这个hbase.unsfe.stream.capability.enforcee删了，然后就无了，不过问题不大，加回来，set it false@！！！！。



为我good的english鼓掌：做笔记中



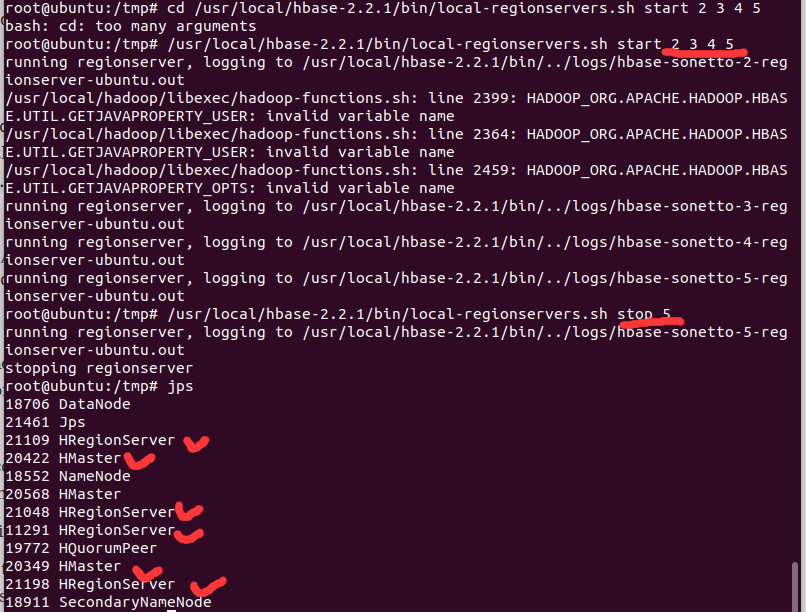
Kill local-master-backup



Kill local-regionservers的节点

我觉得事情逐渐变得不简单。

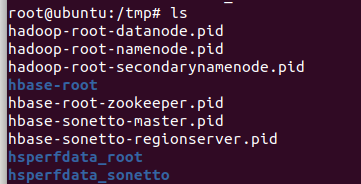
但是我不确定做完实验要不要全部stop，应该不要把，我直接stop-hbase应该可以把



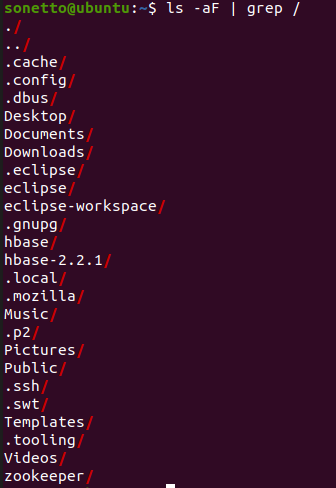
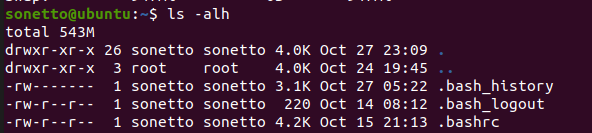
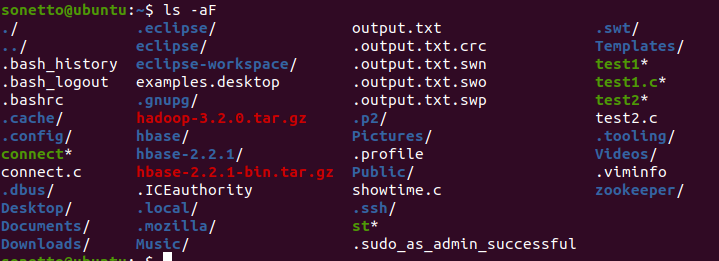
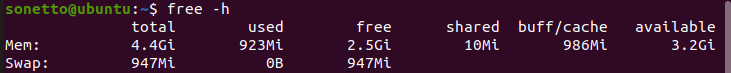
我就礼貌性的留下好了!因为我想了一下，如果不留的话就不伪分布式系统了。



重启了一下好像就没了，

抱着严谨的心态，我得记一下，我的hadoop和 hbase的用户名不是同一个。

讲评时学到的一些指令用法（记录下来方便我以后看）：



删除表之前如果不disable会报错的噢：



列族要先定义，列可以不用事先定义！！