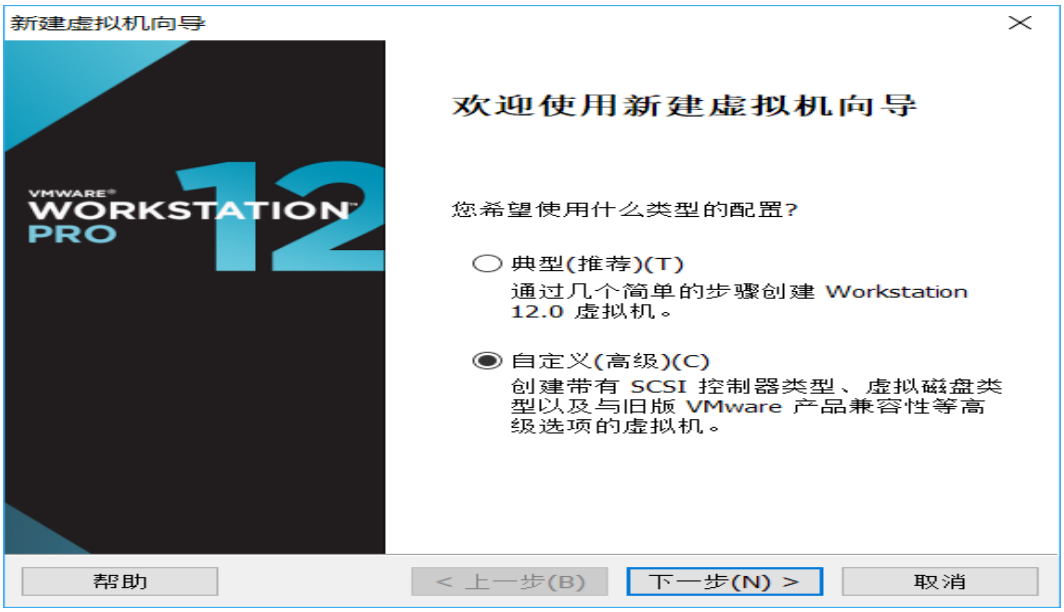
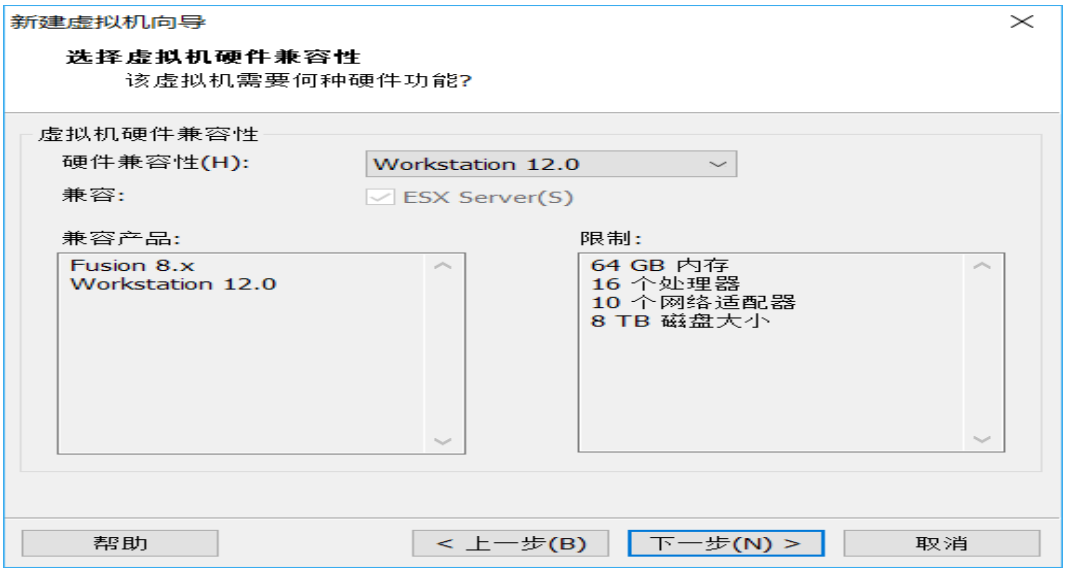
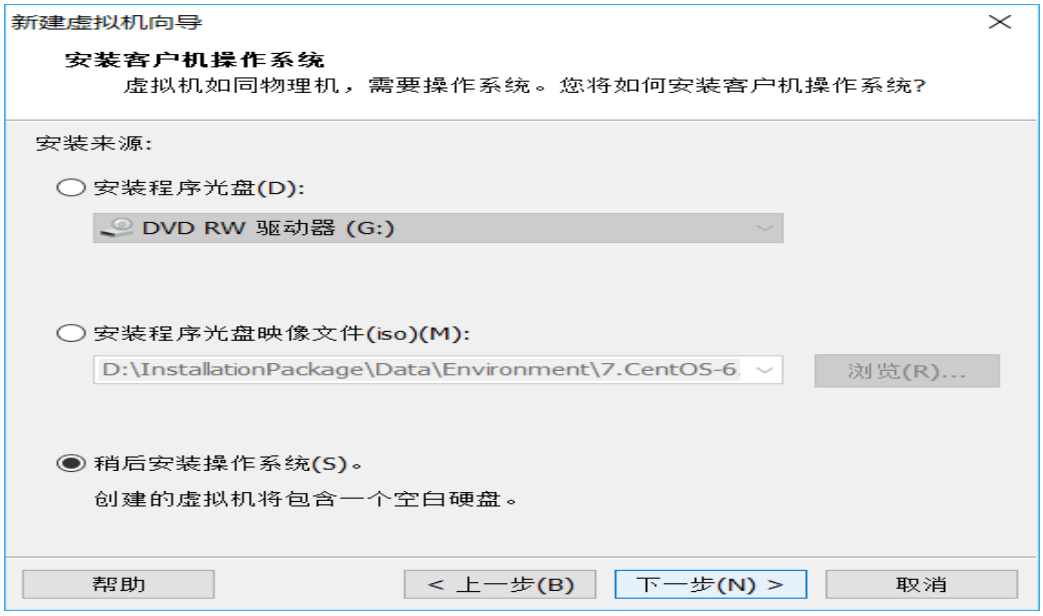
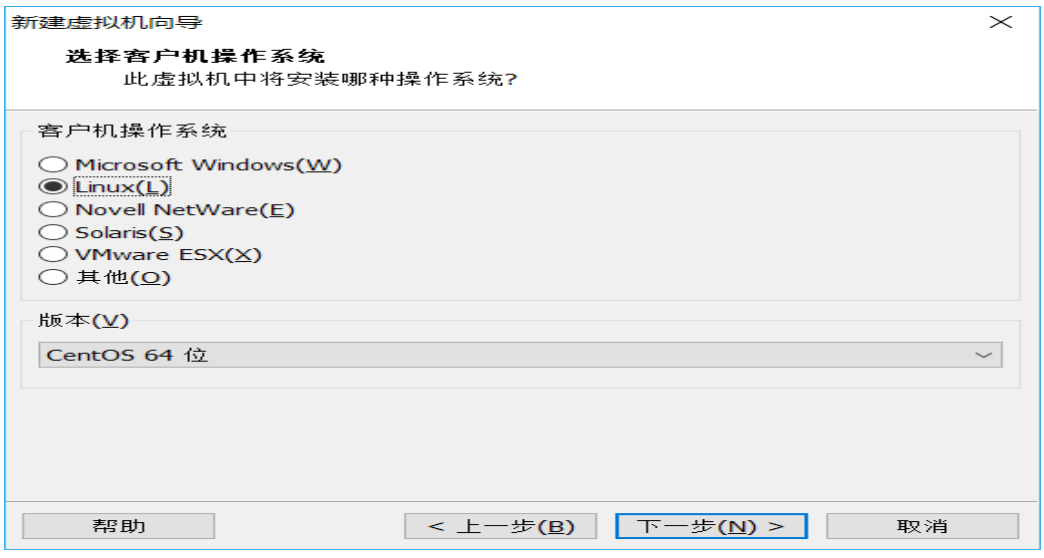
HBase

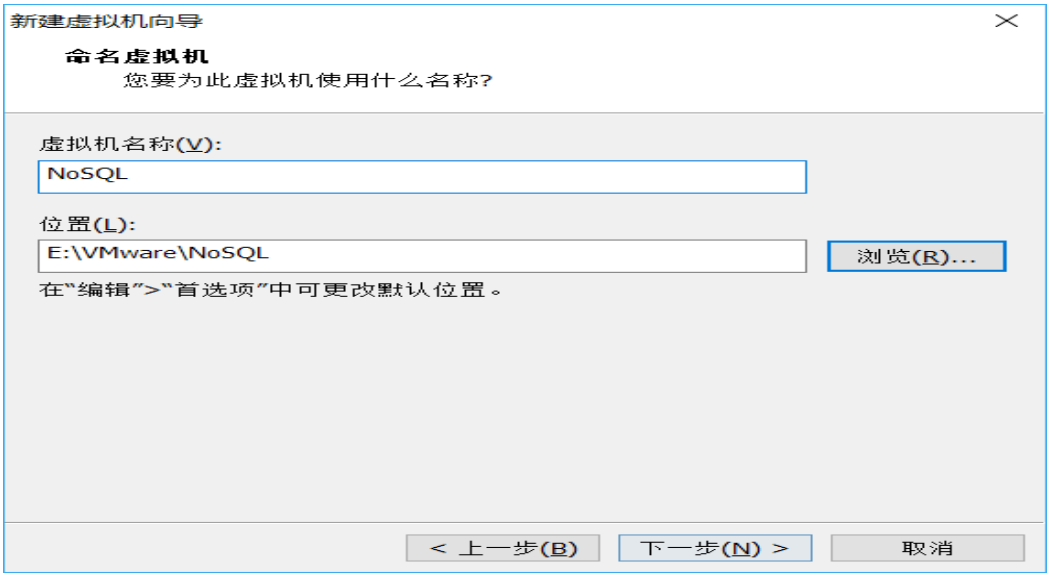
虚拟机安装 CentOS7



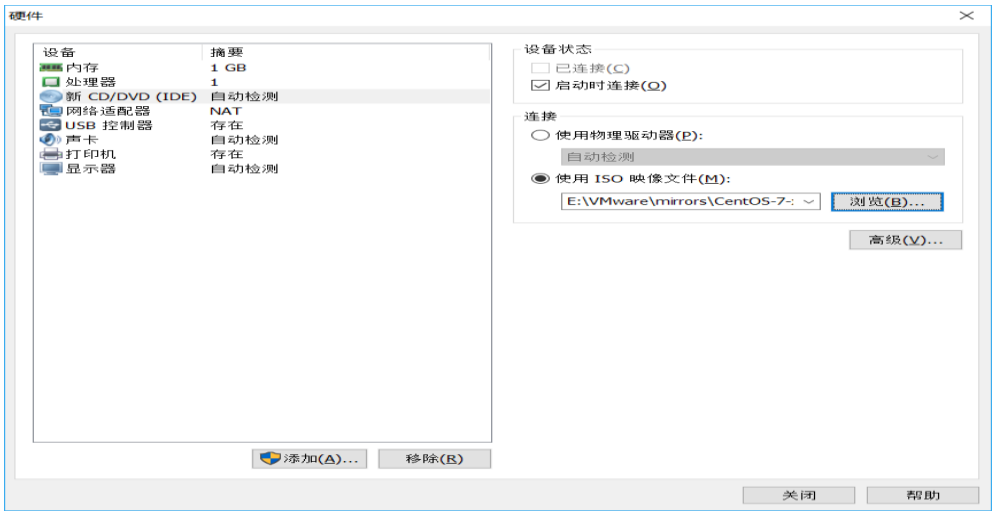

新建虚拟机向导
选择虚拟机硬件兼容性
该虚拟机需要何种硬件功能?
虚拟机硬件兼容性
硬件兼容性(H):
Workstation 12.0
兼容:
ESX Server(S)
兼容产品:
Fusion 8.x
Workstation 12.0
限制:
64 GB 内存
16 个处理器
10 个网络适配器
8 TB 磁盘大小
帮助
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
安装客户机操作系统
虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统?
安装来源:
安装程序光盘(D):
DVD RW 驱动器 (G:)
安装程序光盘映像文件(iso)(M):
D:\InstallationPackage\Data\Environment\7.CentOS-6
浏览(R)...
稍后安装操作系统(S)。
创建的虚拟机将包含一个空白硬盘。
帮助
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
选择客户机操作系统
此虚拟机中将安装哪种操作系统?
客户机操作系统
Microsoft Windows(W)
Linux(L)
Novell NetWare(E)
Solaris(S)
VMware ESX(X)
其他(O)
版本(V)
CentOS 64 位
帮助
< 上一步(B)
下一步(N) >
取消

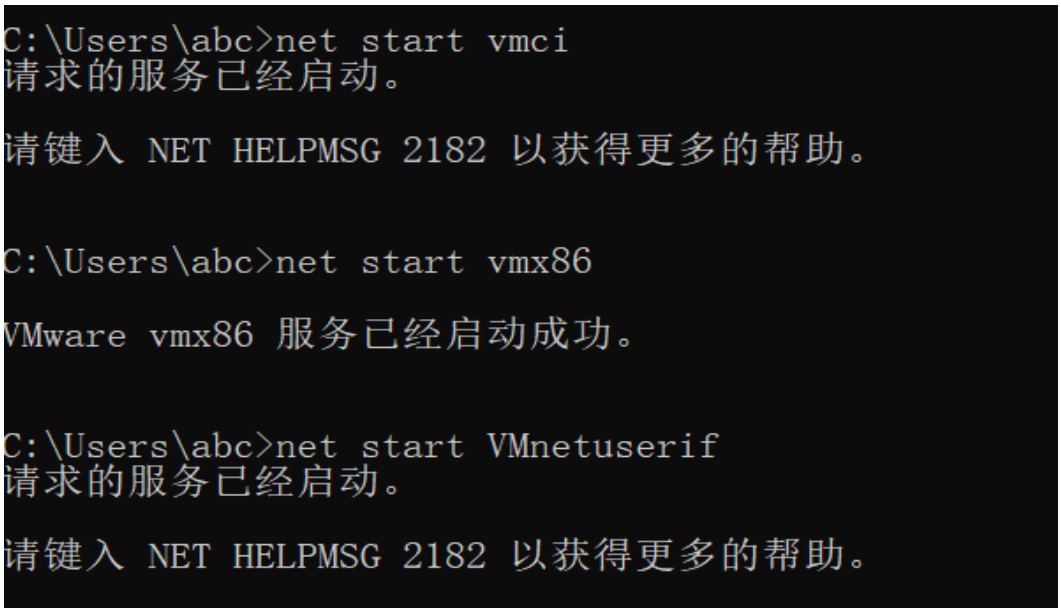

新建虚拟机向导
命名虚拟机
您要为此虚拟机使用什么名称?
虚拟机名称(V):
NoSQL
位置(L):
E:\VMware\NoSQL
浏览(R)...
在"编辑">"首选项"中可更改默认位置。
< 上一步(B)
下一步(N) >
取消


新建虚拟机向导
指定磁盘容量
磁盘大小为多少?
最大磁盘大小(GB)(S):
20.0
针对 CentOS 64 位 的建议大小: 20 GB
立即分配所有磁盘空间(A)。
分配所有容量可以提高性能，但要求所有物理磁盘空间立即可用。如果不立即分配所有空间，虚拟磁盘的空间最初很小，会随着您向其中添加数据而不断变大。
将虚拟磁盘存储为单个文件(O)
将虚拟磁盘拆分成多个文件(M)
拆分磁盘后，可以更轻松地在计算机之间移动虚拟机，但可能会降低大容量磁盘的性能。
帮助
< 上一步(B)
下一步(N) >
取消


硬件
设备
摘要
内存
1 GB
处理器
1
新 CD/DVD (IDE)
自动检测
网络适配器
NAT
USB 控制器
存在
声卡
自动检测
打印机
存在
显示器
自动检测
添加(A)...
移除(R)
设备状态
已连接(C)
启动时连接(O)
连接
使用物理驱动器(P):
自动检测
使用 ISO 映像文件(M):
E:\VMware\mirrors\CentOS-7-
浏览(B)...
高级(V)...
关闭
帮助

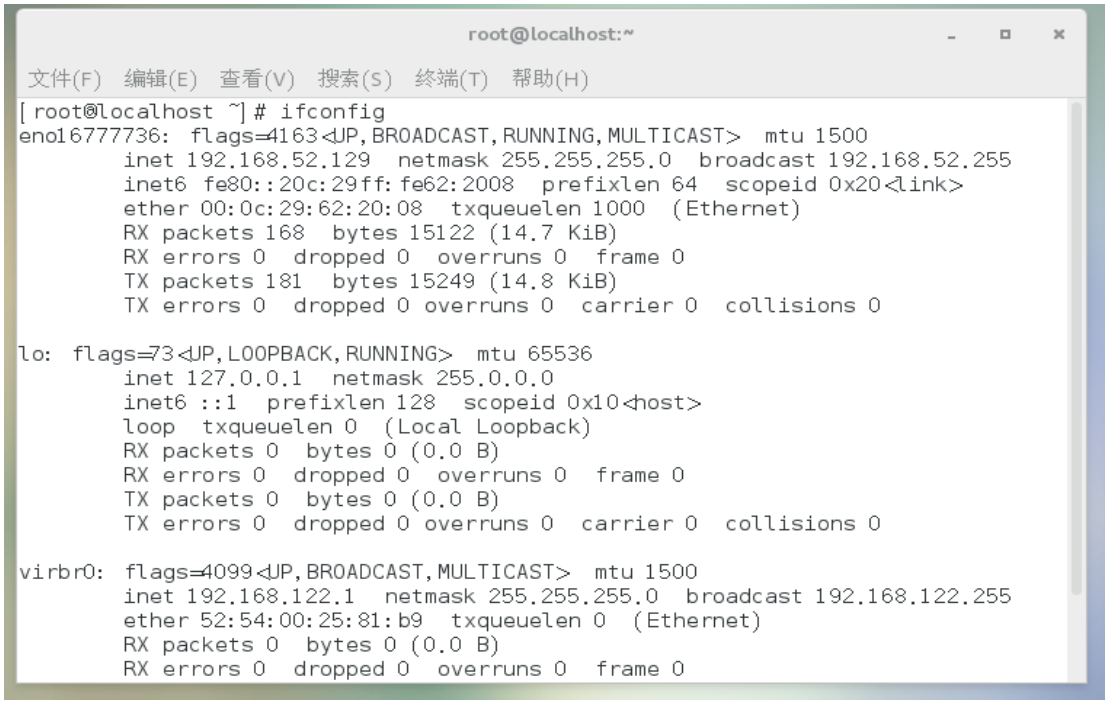
原创 无法打开内核设备“\\.\Global\vmx86”：系统找不到指定的文件。是否在安装VMwar

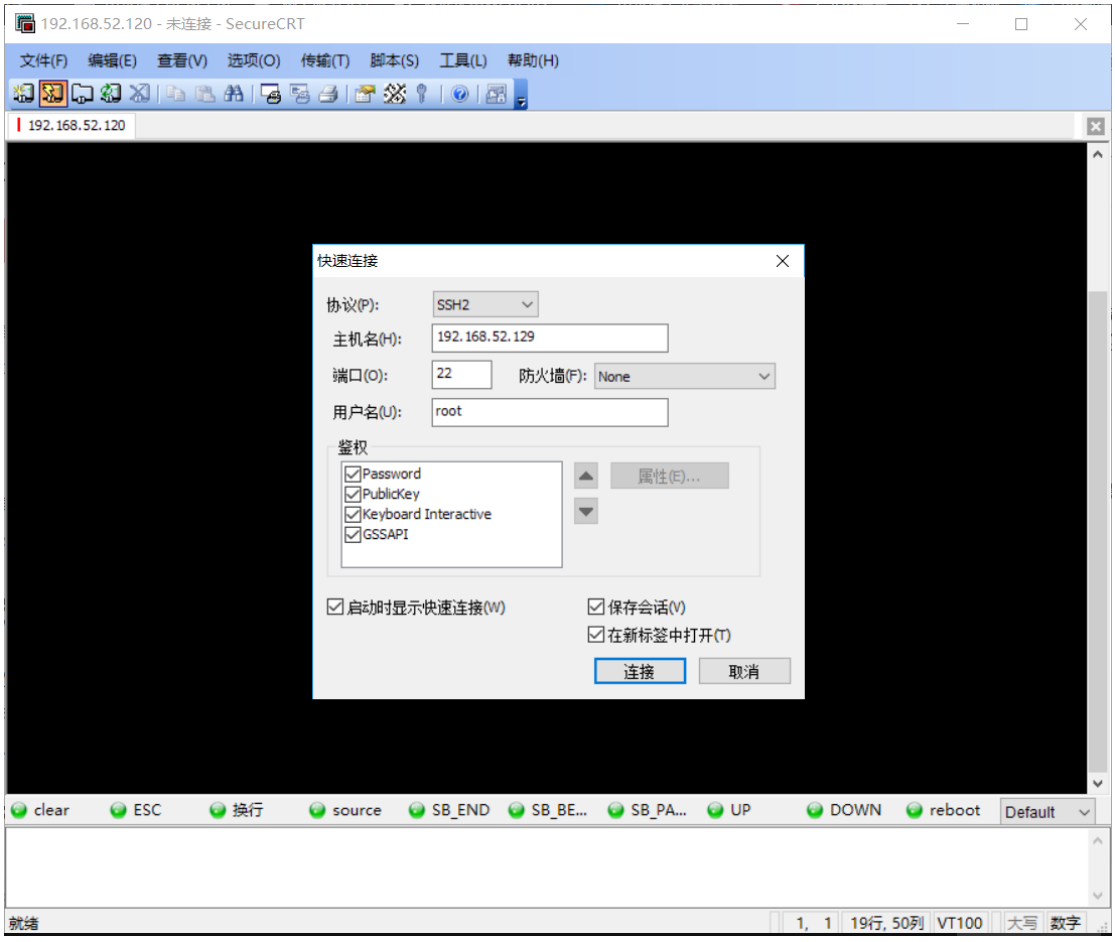
1. 点击“开始→运行”，在运行框中输入 cmd 回车打开命令提示符，然后依次执行以下命令
2. 输入以下的命令并回车
   net start vmci
   net start vmx86
   net start VMnetuserif
3. 改变vmware几种服务的启动方式为：
   sc config vmci start= auto
   sc config vmx86 start= auto
   sc config VMnetuserif start= auto
   这一点儿与win7下面的有所不同，特此提醒，win7下面的是：
   sc config vmci=auto
   sc config vmx86=auto
   sc config VMnetuserif=auto

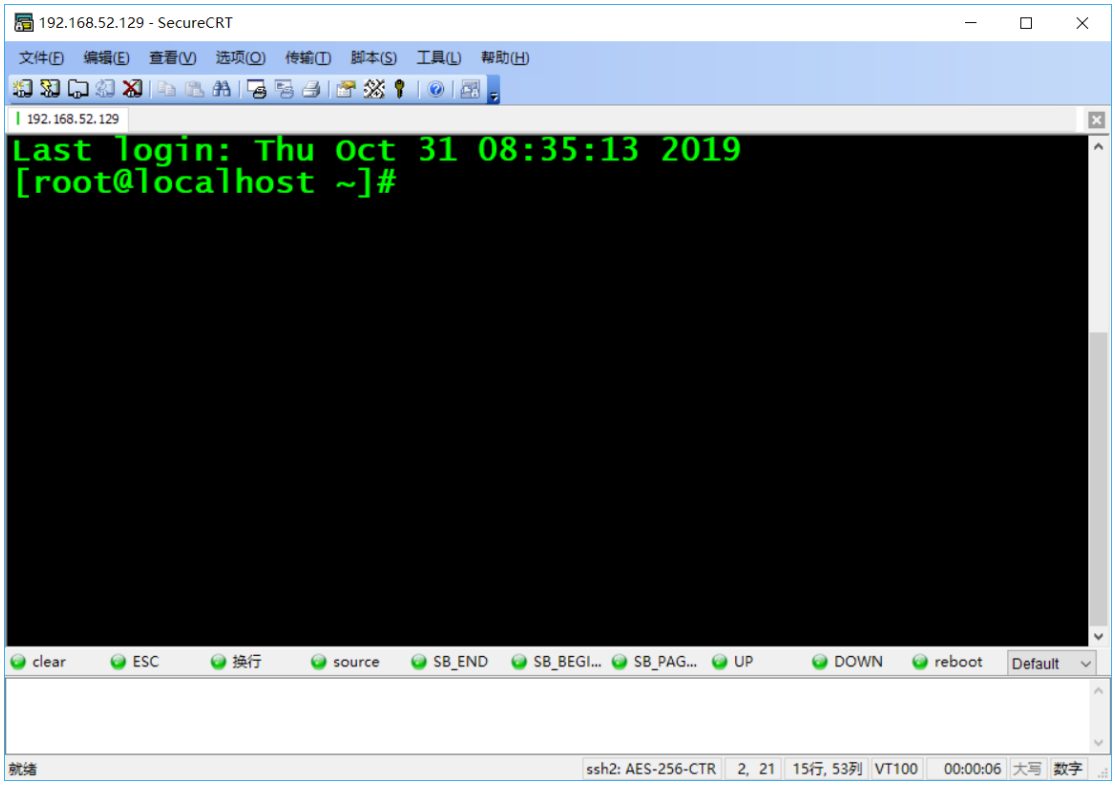


远程连接虚拟机

ifconfig







HDFS 伪分布式搭建

查看自带的 openjdk

```
rpm -qa | grep java
```

```
[root@localhost ~]# rpm -qa | grep java
java-1.8.0-openjdk-headless-1.8.0.65-3.b17.el7.x86_64
libvirt-java-0.4.9-4.el7.noarch
java-1.7.0-openjdk-devel-1.7.0.91-2.6.2.3.el7.x86_64
java-1.6.0-openjdk-1.6.0.36-1.13.8.1.el7_1.x86_64
javamail-1.4.6-8.el7.noarch
nuxwdog-client-java-1.0.3-2.el7.x86_64
java-1.7.0-openjdk-1.7.0.91-2.6.2.3.el7.x86_64
javassist-3.16.1-10.el7.noarch
java-1.6.0-openjdk-devel-1.6.0.36-1.13.8.1.el7_1.x86_
64
java-1.7.0-openjdk-headless-1.7.0.91-2.6.2.3.el7.x86_
64
libvirt-java-devel-0.4.9-4.el7.noarch
tzdata-java-2015g-1.el7.noarch
javapackages-tools-3.4.1-11.el7.noarch
java-1.8.0-openjdk-1.8.0.65-3.b17.el7.x86_64
python-javapackages-3.4.1-11.el7.noarch
java-1.8.0-openjdk-devel-1.8.0.65-3.b17.el7.x86_64
[root@localhost ~]#
```

卸载系统自带的 openjdk

```
rpm -e java-1.8.0-openjdk-headless-1.8.0.65-3.b17.el7.x86_64 libvirt-java-0.4.9-4.el7.noarch java-1.7.0-openjdk-devel-1.7.0.91-2.6.2.3.el7.x86_64 java-1.6.0-openjdk-1.6.0.36-1.13.8.1.el7_1.x86_64 javamail-1.4.6-8.el7.noarch nuxwdog-client-java-1.0.3-2.el7.x86_64 java-1.7.0-openjdk-1.7.0.91-2.6.2.3.el7.x86_64 javassist-3.16.1-10.el7.noarch java-1.6.0-openjdk-devel-1.6.0.36-1.13.8.1.el7_1.x86_64 java-1.7.0-openjdk-headless-1.7.0.91-2.6.2.3.el7.x86_64 libvirt-java-devel-0.4.9-4.el7.noarch tzdata-java-2015g-1.el7.noarch javapackages-tools-3.4.1-11.el7.noarch java-1.8.0-openjdk-1.8.0.65-3.b17.el7.x86_64 python-javapackages-3.4.1-11.el7.noarch java-1.8.0-openjdk-devel-1.8.0.65-3.b17.el7.x86_64 --nodeps
```

```
[root@localhost ~]# rpm -e java-1.8.0-openjdk-headless-1.8.0.65-3.b17.el7.x86_64 libvirt-java-0.4.9-4.el7.noarch java-1.7.0-openjdk-devel-1.7.0.91-2.6.2.3.el7.x86_64 java-1.6.0-openjdk-1.6.0.36-1.13.8.1.el7_1.x86_64 javamail-1.4.6-8.el7.noarch nuxwdog-client-java-1.0.3-2.el7.x86_64 java-1.7.0-openjdk-1.7.0.91-2.6.2.3.el7.x86_64 javassist-3.16.1-10.el7.noarch java-1.6.0-openjdk-devel-1.6.0.36-1.13.8.1.el7_1.x86_64 java-1.7.0-openjdk-headless-1.7.0.91-2.6.2.3.el7.x86_64 libvirt-java-devel-0.4.9-4.el7.noarch tzdata-java-2015g-1.el7.noarch javapackages-tools-3.4.1-11.el7.noarch java-1.8.0-openjdk-1.8.0.65-3.b17.el7.x86_64 python-javapackages-3.4.1-11.el7.noarch java-1.8.0-openjdk-devel-1.8.0.65-3.b17.el7.x86_64 --nodeps
```

```
[root@localhost ~]# java
bash: java: 未找到命令...
[root@localhost ~]#
```

各台虚拟机关闭防火墙

各台机器执行以下命令（root 用户来执行）

```
service iptables stop

chkconfig iptables off
```

```
[root@node01 hadoop-2.6.0-cdh5.14.0]# service iptables stop
Redirecting to /bin/systemctl stop  iptables.service
Failed to stop iptables.service: Unit iptables.service not loaded.
[root@node01 hadoop-2.6.0-cdh5.14.0]# chkconfig iptables off
```

1:查看防火状态

```
systemctl status firewalld

service  iptables status
```

2:暂时关闭防火墙

```
systemctl stop firewalld

service  iptables stop
```

3:永久关闭防火墙

```
systemctl disable firewalld

chkconfig iptables off
```

4:重启防火墙

```
systemctl enable firewalld

service iptables restart
```

5:永久关闭后重启

```
chkconfig iptables on
```

各台机器关闭 selinux（linux 里面的安全策略，类似防火墙）

```
vim /etc/selinux/config
```

各台机器更改主机名

```
vim /etc/sysconfig/network
```

```
[root@localhost ~]# vim /etc/sysconfig/network
```

```
NETWORKING=yes
HOSTNAME=node01.hadoop.com
```

```
NETWORKING=yes
HOSTNAME=node01.hadoop.com
```

各台机器做主机名与 IP 地址的映射

```
vim /etc/hosts
```

```
192.168.52.129 node01.hadoop.com   node01
```

```
1 192.168.52.129
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.52.129 node01.hadoop.com  node01
~
```

各台机器重启

```
reboot  -h  now
```

创建文件夹

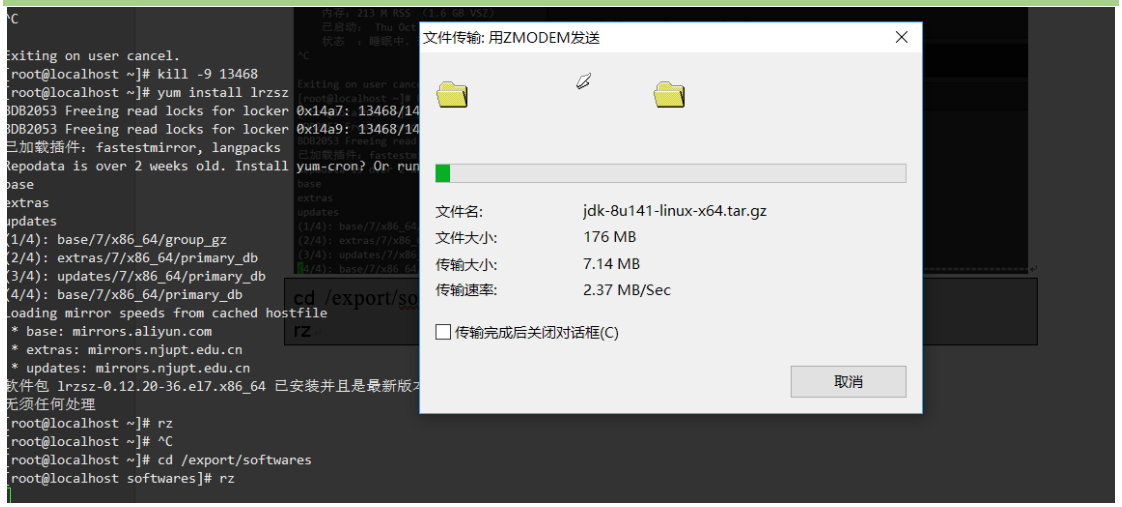
```
mkdir -p /export/servers

mkdir -p /export/softwares
```

```
[root@localhost ~]# mkdir -p /export/servers
[root@localhost ~]# mkdir -p /export/softwares
```

```
yum install lrzsz
```

```
Another app is currently holding the yum lock; waiting for it to exit...
  另一个应用程序是：PackageKit
    内存：213 M RSS （1.6 GB VSZ）
    已启动： Thu Oct 31 08:36:25 2019 - 23:45之前
    状态 ：睡眠中，进程ID：13468
^C

Exiting on user cancel.
[root@localhost ~]# kill -9 13468
[root@localhost ~]# yum install lrzsz
BDB2053 Freeing read locks for locker 0x14a7: 13468/140064473245504
BDB2053 Freeing read locks for locker 0x14a9: 13468/140064473245504
已加载插件：fastestmirror, langpacks
Repodata is over 2 weeks old. Install yum-cron? Or run: yum makecache fast
base
extras
updates
(1/4): base/7/x86_64/group_gz
(2/4): extras/7/x86_64/primary_db
(3/4): updates/7/x86_64/primary_db
(4/4): base/7/x86_64/primary_db                                89% [=========================================
```

```
cd /export/softwares

rz
```




```
[root@localhost softwares]# ll
总用量 181172
-rw-r--r--. 1 root root 185516505 9月  27 2018 jdk-8u141-linux-x64.tar.gz
[root@localhost softwares]#
```

```
tar -zxvf jdk-8u141-linux-x64.tar.gz -C ../servers/
```

```
[root@localhost softwares]# cd ..
[root@localhost export]# ll
总用量 0
drwxr-xr-x. 3 root root 25 10月 31 09:05 servers
drwxr-xr-x. 2 root root 39 10月 31 09:03 softwares
[root@localhost export]# cd servers/
[root@localhost servers]# ll
总用量 4
drwxr-xr-x. 8 10 143 4096 7月  12 2017 jdk1.8.0_141
[root@localhost servers]#
```

配置环境变量

```
vim /etc/profile
```

```
export JAVA_HOME=/export/servers/jdk1.8.0_141
export PATH=:$JAVA_HOME/bin:$PATH
```

```
# By default, we want umask to get set. This sets it for login shell
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge

export JAVA_HOME=/export/servers/jdk1.8.0_141
export PATH=:$JAVA_HOME/bin:$PATH
-- 插入 --
```

```
source /etc/profile
```

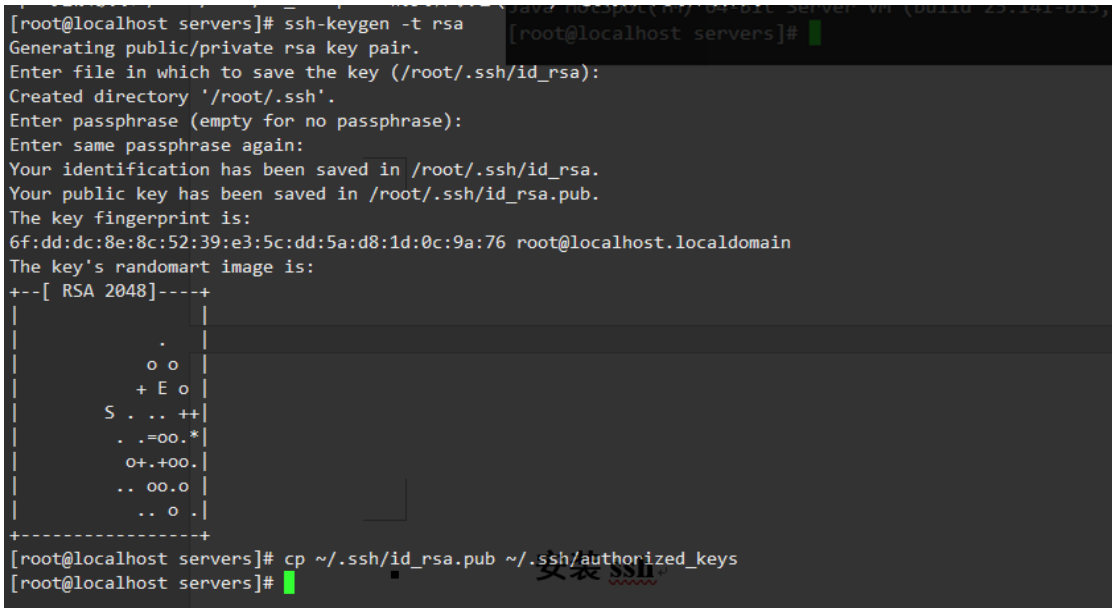
```
[root@localhost servers]# source /etc/profile
[root@localhost servers]#
```

安装包的分发

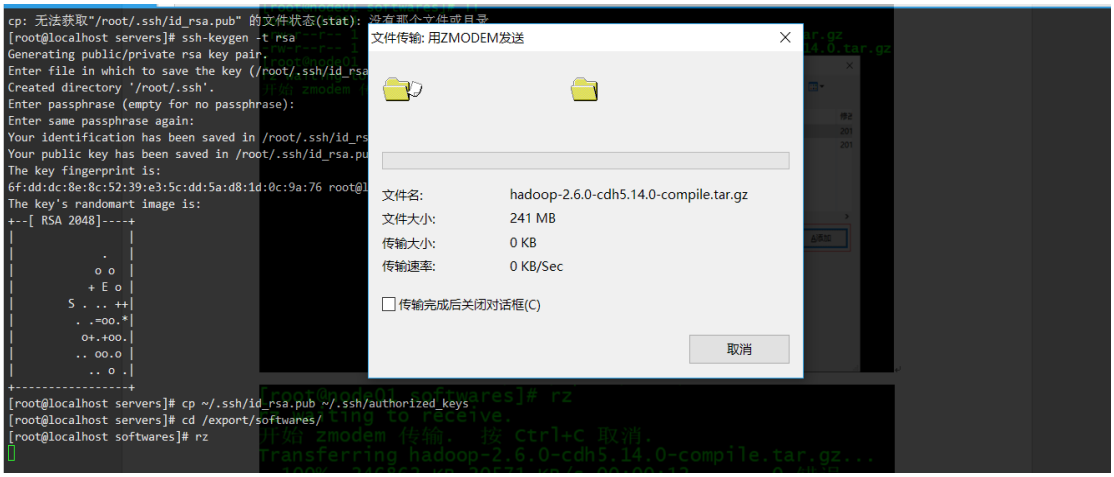
```
[root@localhost servers]# java -version
java version "1.8.0_141"
Java(TM) SE Runtime Environment (build 1.8.0_141-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
[root@localhost servers]#
```

安装 ssh

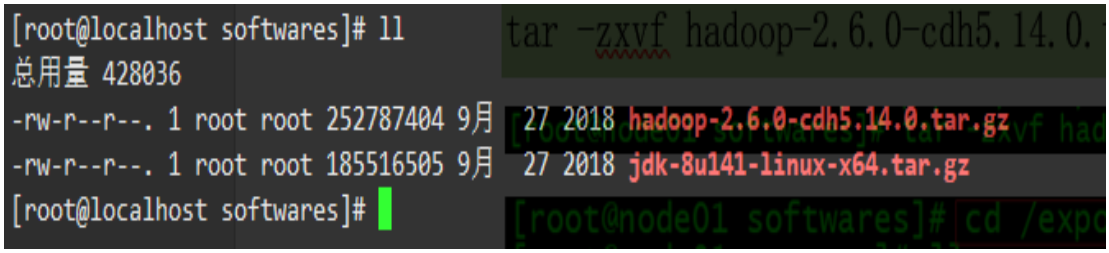
```
sudo yum install ssh

ssh-keygen -t rsa

cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```




```
cd /export/softwares/
rz
```

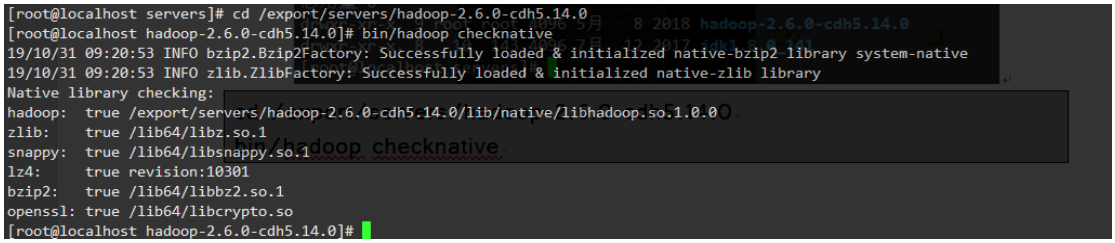



```
mv hadoop-2.6.0-cdh5.14.0-compile.tar.gz hadoop-2.6.0-cdh5.14.0.tar.gz
```




```
tar -zxvf hadoop-2.6.0-cdh5.14.0.tar.gz -C ../servers/
```


```
cd /export/servers/hadoop-2.6.0-cdh5.14.0

bin/hadoop checknative
```



修改 core-site.xml

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

vim core-site.xml
```

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://node01:8020</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/export/servers/hadoop-2.6.0-
cdh5.14.0/hadoopDatas/tempDatas</value>
    </property>

    <property>
```

```
        <name>io.file.buffer.size</name>
        <value>4096</value>
    </property>

    <property>
        <name>fs.trash.interval</name>
        <value>10080</value>
    </property>
</configuration>
```

修改 hdfs-site.xml

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

vim hdfs-site.xml
```

```
<configuration>
     <property>
          <name>dfs.namenode.secondary.http-address</name>
          <value>node01:50090</value>
    </property>

    <property>
        <name>dfs.namenode.http-address</name>
        <value>node01:50070</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///export/servers/hadoop-2.6.0-
cdh5.14.0/hadoopDatas/namenodeDatas</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///export/servers/hadoop-2.6.0-
cdh5.14.0/hadoopDatas/datanodeDatas</value>
    </property>

    <property>
        <name>dfs.namenode.edits.dir</name>
```

```
        <value>file:///export/servers/hadoop-2.6.0-
cdh5.14.0/hadoopDatas/dfs/nn/edits</value>
    </property>
    <property>
        <name>dfs.namenode.checkpoint.dir</name>
        <value>file:///export/servers/hadoop-2.6.0-
cdh5.14.0/hadoopDatas/dfs/snn/name</value>
    </property>
    <property>
        <name>dfs.namenode.checkpoint.edits.dir</name>
        <value>file:///export/servers/hadoop-2.6.0-
cdh5.14.0/hadoopDatas/dfs/nn/snn/edits</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.permissions</name>
        <value>false</value>
    </property>
<property>
        <name>dfs.blocksize</name>
        <value>134217728</value>
    </property>
</configuration>
```

修改 hadoop-env.sh

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

vim hadoop-env.sh
```

```
export JAVA_HOME=/export/servers/jdk1.8.0_141
```

修改 mapred-site.xml

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

vim mapred-site.xml
```

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

    <property>
        <name>mapreduce.job.ubertask.enable</name>
        <value>true</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>node01:10020</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>node01:19888</value>
    </property>
</configuration>
```

修改 yarn-site.xml

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

vim yarn-site.xml
```

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>node01</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>

    <property>
```

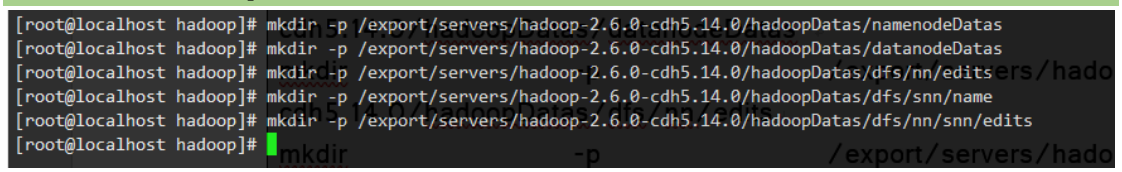
```
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
    </property>
    <property>
        <name>yarn.log-aggregation.retain-seconds</name>
        <value>604800</value>
    </property>
</configuration>
```

修改 slaves 文件

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop

vim slaves
```

创建文件存放目录

```
mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/tempDatas

mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/namenodeDatas

mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/datanodeDatas

mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/nn/edits

mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/snn/name

mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/nn/snn/edits
```

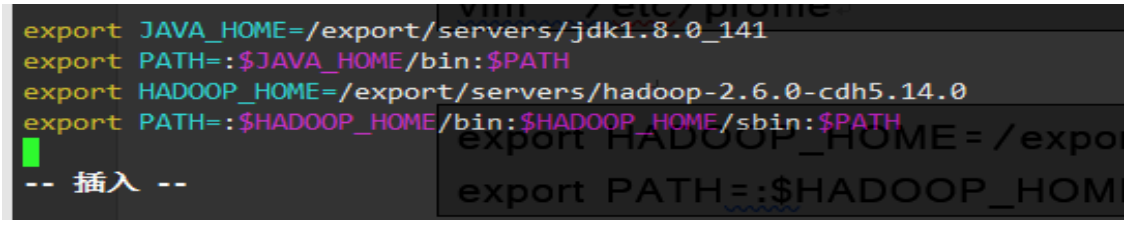


配置 hadoop 的环境变量

```
vim /etc/profile
```

```
export HADOOP_HOME=/export/servers/hadoop-2.6.0-cdh5.14.0
```

```
export PATH=:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```




```
source /etc/profile
```

集群启动

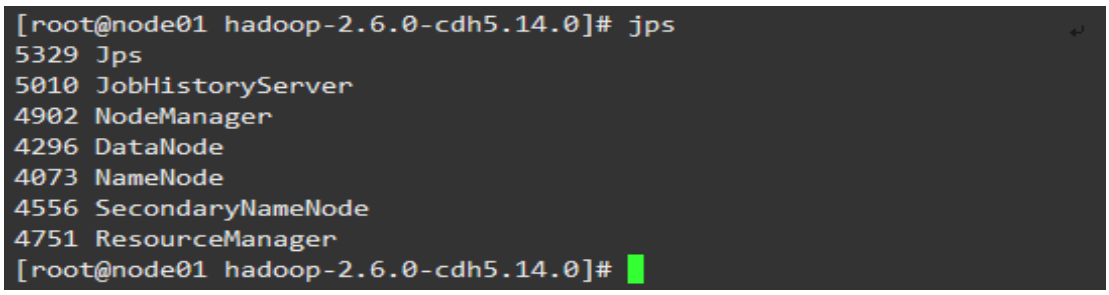
要启动 Hadoop 集群，需要启动 HDFS 和 YARN 两个集群。

注意：首次启动 HDFS 时，必须对其进行格式化操作。本质上是一些清理和准备工作，因为此时的 HDFS 在物理上还是不存在的。

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/

bin/hdfs namenode  -format
```

脚本一键启动

```
cd /export/servers/hadoop-2.6.0-cdh5.14.0/

sbin/start-dfs.sh

sbin/start-yarn.sh

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



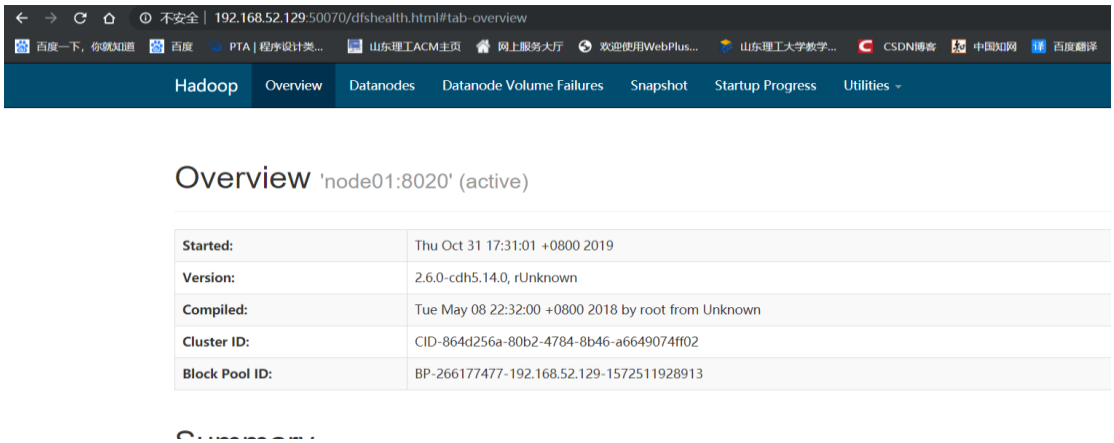
停止集群

```
sbin/stop-dfs.sh

sbin/stop-yarn.sh

sbin/mr-jobhistory-daemon.sh stop historyserver
```

浏览器查看启动页面

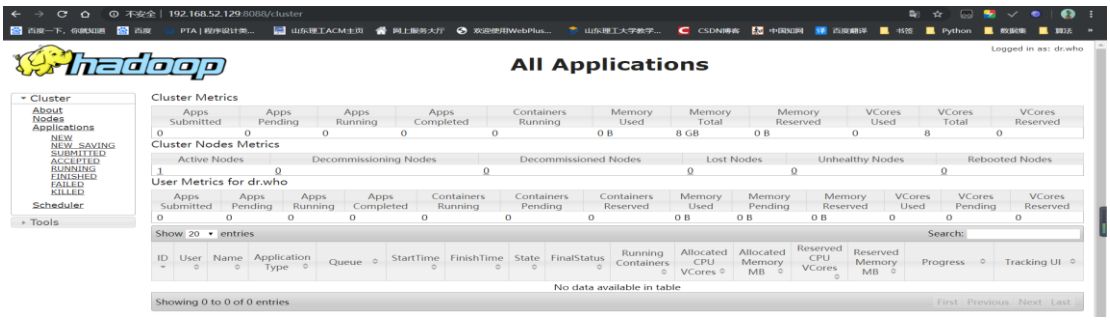
hdfs 集群访问地址

http://192.168.52.129:50070/dfshealth.html#tab-overview



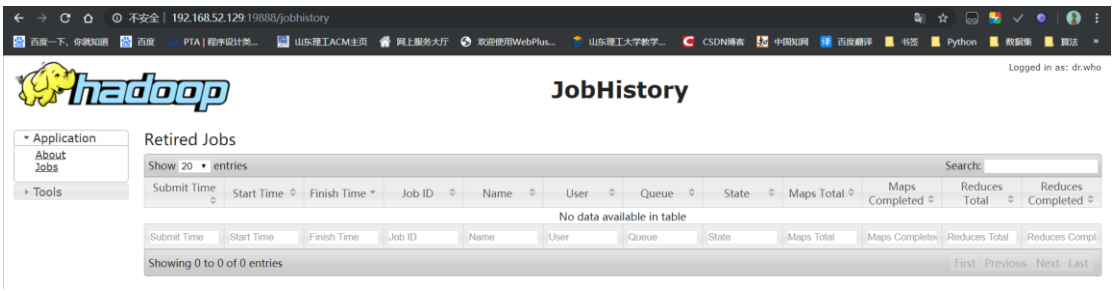
yarn 集群访问地址

http://192.168.52.129:8088/cluster



jobhistory 访问地址

http://192.168.52.129:19888/jobhistory



Hbase 安装

使用 wget 下载 Hbase

```
wget http://archive.apache.org/dist/hbase/1.4.8/hbase-1.4.8-bin.tar.gz
```

```
[root@node01 softwares]# wget http://archive.apache.org/dist/hbase/1.4.8/hbase-1.4.8-bin.tar.gz
--2019-11-01 18:10:35--  http://archive.apache.org/dist/hbase/1.4.8/hbase-1.4.8-bin.tar.gz
正在解析主机 archive.apache.org (archive.apache.org)... 163.172.17.199
正在连接 archive.apache.org (archive.apache.org)|163.172.17.199|:80... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度：113106551 (108M) [application/x-gzip]
正在保存至：“hbase-1.4.8-bin.tar.gz”

29% [=====================================================>
```

解压目录

```
tar -zxvf hbase-1.4.8-bin.tar.gz -C ../servers/
```

```
[root@node01 softwares]# cd ../servers/
[root@node01 servers]# ll
总用量 8
drwxr-xr-x. 11 root root 4096 10月 31 09:38 hadoop-2.6.0-cdh5.14.0
drwxr-xr-x   7 root root  150 11月  1 18:22 hbase-1.4.8
drwxr-xr-x.  8   10  143 4096 7月  12 2017 jdk1.8.0_141
[root@node01 servers]#
```

```
mkdir /export/servers/hbase-1.4.8/zk_data
```

设置 hbase 环境变量

```
vi /etc/profile
```

```
export HBASE_HOME=/export/servers/hbase-1.4.8
export PATH=:$HBASE_HOME/bin:$PATH
```

```
export JAVA_HOME=/export/servers/jdk1.8.0_141
export PATH=:$JAVA_HOME/bin:$PATH
export HADOOP_HOME=/export/servers/hadoop-2.6.0-cdh5.14.0
export PATH=:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export HBASE_HOME=/export/servers/hbase-1.4.8
export PATH=:$HBASE_HOME/bin:$PATH
-- INSERT --
```

```
source /etc/profile
```

配置 hbase-env.sh 文件

```
vi /export/servers/hbase-1.4.8/conf/hbase-env.sh
```

```
export JAVA_HOME=/export/servers/jdk1.8.0_141/
export HBASE_MANAGES_ZK=true #此配置信息，设置由 hbase 自己管理 zookeeper，不需要单独的 zookeeper
export HBASE_PID_DIR=/export/servers/hbase-1.4.8/pids
```

配置 hbase-site.xml

```
vi /export/servers/hbase-1.4.8/conf/hbase-site.xml
```

```
[root@node01 conf]# vi /export/servers/hbase-1.4.8/conf/hbase-env.sh
[root@node01 conf]# vi /export/servers/hbase-1.4.8/conf/hbase-site.xml
```

```
<configuration>
    <property>
        <name>hbase.rootdir</name>
        <value>hdfs://node01:8020/hbase</value>
    </property>
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
    </property>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>node01</value>
    </property>
    <property>
      <name>hbase.zookeeper.property.dataDir</name>
      <value>/export/servers/hbase-1.4.8/zk_data</value>
   </property>
</configuration>
```

启动 hbase

```
start-hbase.sh
```

```
[root@node01 conf]# start-hbase.sh
node01: running zookeeper, logging to /export/servers/hbase-1.4.8/bin/../logs/hbase-root-zookeeper-node01.hadoop.com.out
running master, logging to /export/servers/hbase-1.4.8/logs/hbase-root-master-node01.hadoop.com.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
: running regionserver, logging to /export/servers/hbase-1.4.8/logs/hbase-root-regionserver-node01.hadoop.com.out
: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
```

```
[root@node01 conf]# jps
7728 SecondaryNameNode
7425 NameNode
8036 JobHistoryServer
16196 HRegionServer
7881 ResourceManager
16011 HQuorumPeer
7532 DataNode
16332 Jps
16077 HMaster
7982 NodeManager
```

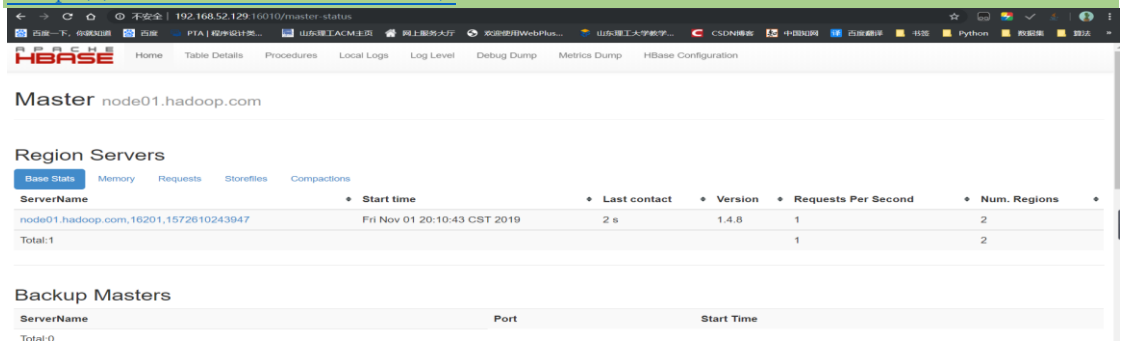
进入 hbase shell

```
hbase shell
```

```
[root@node01 bin]# start-hbase.sh
node01: running zookeeper, logging to /export/servers/hbase-1.4.8/bin/../logs/hbase-root-zookeeper-node01.hadoop.com.out
running master, logging to /export/servers/hbase-1.4.8/logs/hbase-root-master-node01.hadoop.com.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
: running regionserver, logging to /export/servers/hbase-1.4.8/logs/hbase-root-regionserver-node01.hadoop.com.out
[root@node01 bin]# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/export/servers/hbase-1.4.8/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/export/servers/hadoop-2.6.0-cdh5.14.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.8, r91118ce5f10fb4efa03cc8c5a47c7ce97175e85e, Tue Oct  2 11:48:24 PDT 2018

hbase(main):001:0> version
1.4.8, r91118ce5f10fb4efa03cc8c5a47c7ce97175e85e, Tue Oct  2 11:48:24 PDT 2018

hbase(main):002:0>
```

进入 hbase 的 web 页面

```
http://192.168.52.129:16010/
```



HBase shell

```
[root@node01 ~]# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/export/servers/hbase-1.4.8/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/export/servers/hadoop-2.6.0-cdh5.14.0/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.8, r91118ce5f10fb4efa03cc8c5a47c7ce97175e85e, Tue Oct  2 11:48:24 PDT 2018

hbase(main):001:0>
```

version：显示当前 HBase 版本号

```
hbase(main):001:0> version
1.4.8, r91118ce5f10fb4efa03cc8c5a47c7ce97175e85e, Tue Oct  2 11:48:24 PDT 2018
```

status：显示各个主节点状态

```
hbase(main):003:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load
```

whoami：显示当前用户名

```
hbase(main):005:0> whoami
root (auth:SIMPLE)
    groups: root
```

表和列族操作

创建表

```
create 'player','basic'
```

```
hbase(main):001:0> create 'player','basic'
0 row(s) in 3.9300 seconds

=> Hbase::Table - player
```

大小写参数敏感

```
create 'PLAYER','basic'
```

```
hbase(main):002:0> create 'PLAYER','basic'
0 row(s) in 2.3730 seconds

=> Hbase::Table - PLAYER
```

建表时指定列族的参数

```
create 'PLAYER1',{NAME=>'basic',VERSIONS=>5,BLOCKCACHE=>true}
```

```
hbase(main):004:0> create 'PLAYER1',{NAME=>'basic',VERSIONS=>5,BLOCKCACHE=>true}
0 row(s) in 4.3940 seconds

=> Hbase::Table - PLAYER1
```

查看表名列表

```
list
```

```
hbase(main):005:0> list
TABLE
PLAYER
PLAYER1
player
3 row(s) in 0.1830 seconds

=> ["PLAYER", "PLAYER1", "player"]
```

```
exists 'player'
```

```
hbase(main):006:0> exists 'player'
Table player does exist
0 row(s) in 0.0260 seconds
```

描述表结构

```
describe 'player'
```

```
hbase(main):007:0> describe 'player'
Table player is ENABLED
player
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE',
TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.1430 seconds
```

Hbase Web 界面

Tables

User Tables | System Tables | Snapshots

3 table(s) in set. [Details]

Namespace	Table Name	Online Regions	Offline Regions	Failed Regions	Split Regions	Other Regions	Description
default	PLAYER	1	0	0	0	0	'PLAYER', {NAME => 'basic'}
default	PLAYER1	1	0	0	0	0	'PLAYER1', {NAME => 'basic', VERSIONS => '5'}
default	player	1	0	0	0	0	'player', {NAME => 'basic'}

修改表结构

增加列族

```
alter 'player','advanced'
```

```
hbase(main):008:0> alter 'player' , 'advanced'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 4.0740 seconds
```

```
alter 'player','basic',{NAME=>'advanced',IN_MEMORY=>true}
```

```
hbase(main):013:0> alter 'player','basic',{NAME=>'advanced',IN_MEMORY=>true}
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 6.6720 seconds
```

```
hbase(main):014:0> desc 'player'
Table player is ENABLED
player
COLUMN FAMILIES DESCRIPTION
{NAME => 'advanced', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'true', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE'
, TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'basic', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'true', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', T
TL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.0300 seconds
```

修改列族属性

```
alter 'player',{NAME=>'basic',IN_MEMORY=>true}
```

```
hbase(main):009:0> alter 'player',{NAME=>'basic',IN_MEMORY=>true}
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.7810 seconds
```

```
hbase(main):010:0> desc 'player'
Table player is ENABLED
player
COLUMN FAMILIES DESCRIPTION
{NAME => 'advanced', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE
', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'basic', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'true', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', T
TL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.0370 seconds
```

删除列族

```
alter 'player' , 'delete'=>'advanced'
```

```
hbase(main):011:0> alter 'player' , 'delete'=>'advanced'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.9700 seconds
```

```
hbase(main):012:0> desc 'player'
Table player is ENABLED
player
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'true', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', T
TL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.0380 seconds
```

```
alter 'player','basic',{NAME=>'advanced',METHOD=>'delete'}
```

```
hbase(main):015:0> alter 'player','basic',{NAME=>'advanced',METHOD=>'delete'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 3.1430 seconds
```

```
hbase(main):016:0> desc 'player'
Table player is ENABLED
player
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'true', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', T
TL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
1 row(s) in 0.2000 seconds
```

删除表

```
disable 'player'
```

```
hbase(main):017:0> disable 'player'
0 row(s) in 2.5690 seconds
```

```
is_disabled 'player'
```

```
hbase(main):018:0> is_disabled 'player'
true
0 row(s) in 0.0460 seconds
```

```
drop 'player'
```

```
hbase(main):023:0> drop 'player'
0 row(s) in 1.6830 seconds
```

数据清空

```
truncate 'player'
```

数据更新

数据插入

```
put 'player' ,'001','basic:pl','MJ'
```

```
hbase(main):029:0> put 'player' ,'001','basic:pl','MJ'
0 row(s) in 0.8810 seconds
```

```
hbase(main):029:0> put 'player' ,'001','basic:pl','MJ'
0 row(s) in 0.8810 seconds

hbase(main):030:0> get 'player','001'
COLUMN                                CELL
 basic:pl                             timestamp=1573088531791, value=MJ
1 row(s) in 0.1430 seconds
```

数据更新

```
hbase(main):031:0> put 'player' ,'001','basic:pl1','MJ1',1
0 row(s) in 0.0170 seconds
```

数据插入

```
hbase(main):032:0> get 'player','001'
COLUMN                                CELL
 basic:pl                             timestamp=1573088531791, value=MJ
 basic:pl1                            timestamp=1, value=MJ1
1 row(s) in 0.0130 seconds
```

数据更新

```
hbase(main):033:0> put 'player' ,'001','basic:pl1','MJ12'
0 row(s) in 0.0170 seconds

hbase(main):034:0> get 'player','001'
COLUMN                                CELL
 basic:pl                             timestamp=1573088531791, value=MJ
 basic:pl1                            timestamp=1573088711290, value=MJ12
1 row(s) in 0.0070 seconds
```

数据删除

```
delete 'player','001','basic:lastname'
```

```
hbase(main):026:0> get 'player','001'
COLUMN                                CELL
 basic:firstname                      timestamp=1573207729344, value=B
 basic:lastname                       timestamp=1573207742536, value=C
 basic:playername                     timestamp=1573207704211, value=A
1 row(s) in 0.0180 seconds

hbase(main):027:0> delete 'player','001','basic:lastname'
0 row(s) in 0.0060 seconds

hbase(main):028:0> get 'player','001'
COLUMN                                CELL
 basic:firstname                      timestamp=1573207729344, value=B
 basic:playername                     timestamp=1573207704211, value=A
1 row(s) in 0.0170 seconds
```

```
delete 'player','001','basic:lastname1',2
```

所有时间戳小于等于 2 的数据都会删掉

```
deleteall 'player','001'
```

```
hbase(main):057:0> get 'player','001'
COLUMN                                CELL
 basic:firstname                      timestamp=1573207729344, value=B
 basic:lastname1                      timestamp=3, value=C
 basic:playername                     timestamp=1573207704211, value=A
1 row(s) in 0.0150 seconds

hbase(main):058:0> deleteall 'player','001'
0 row(s) in 0.2010 seconds

hbase(main):059:0> get 'player','001'
COLUMN                                CELL
0 row(s) in 0.1670 seconds
```

计数器

```
incr 'player','002','basic:scores',10
```

```
hbase(main):063:0> incr 'player','002','basic:scores',10
COUNTER VALUE = 10
0 row(s) in 0.1680 seconds

hbase(main):064:0> get 'player','002'
COLUMN                                CELL
 basic:scores                         timestamp=1573208664619, value=\x00\x00\x00\x00\x00\x00\x00\x0A
1 row(s) in 0.0150 seconds

hbase(main):065:0> incr 'player','002','basic:scores',10
COUNTER VALUE = 20
0 row(s) in 0.0130 seconds

hbase(main):066:0> get 'player','002'
COLUMN                                CELL
 basic:scores                         timestamp=1573208677494, value=\x00\x00\x00\x00\x00\x00\x00\x14
1 row(s) in 0.0100 seconds
```

```
get_counter 'player' ,'002','basic:scores'
```

```
hbase(main):068:0> get_counter 'player' ,'002','basic:scores'
COUNTER VALUE = 20
```

数据查询

```
get 'player','002'
```

```
hbase(main):069:0> get 'player','002'
COLUMN                                CELL
 basic:scores                         timestamp=1573208677494, value=\x00\x00\x00\x00\x00\x00\x00\x14
1 row(s) in 0.1600 seconds
```

数据扫描

```
scan 'player'
```

```
hbase(main):070:0> scan 'player'
ROW                                   COLUMN+CELL
 002                                  column=basic:scores, timestamp=1573208677494, value=\x00\x00\x00\x00\x00\x00\x00\x14
1 row(s) in 0.5540 seconds

hbase(main):071:0>
```

过滤查询

```
show_filters
```

```
hbase(main):001:0> show_filters
DependentColumnFilter
KeyOnlyFilter
ColumnCountGetFilter
SingleColumnValueFilter
PrefixFilter
SingleColumnValueExcludeFilter
FirstKeyOnlyFilter
ColumnRangeFilter
TimestampsFilter
FamilyFilter
QualifierFilter
ColumnPrefixFilter
RowFilter
MultipleColumnPrefixFilter
InclusiveStopFilter
PageFilter
ValueFilter
ColumnPaginationFilter
```

行键过滤器

```
scan 'player' ,FILTER=>"RowFilter(=,'substring:01')"
```

```
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
3 row(s) in 0.0700 seconds

hbase(main):014:0> scan 'player' ,FILTER=>"RowFilter(=,'substring:01')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
2 row(s) in 0.0470 seconds
```

```
scan 'player' ,FILTER=>"RowFilter(<,'binary:002')"
```

```
hbase(main):018:0> scan 'player' ,FILTER=>"RowFilter(<,'binary:002')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
2 row(s) in 0.0140 seconds
```

```
scan 'player' ,FILTER=>"PrefixFilter('001')"
```

```
hbase(main):022:0> scan 'player' ,FILTER=>"PrefixFilter('001')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
2 row(s) in 0.0580 seconds
```

```
scan 'player' ,{STARTROW=>'001',ENDROW=>'002'}
```

```
hbase(main):028:0> scan 'player' ,{STARTROW=>'001',ENDROW=>'002'}
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
2 row(s) in 0.0650 seconds
```

```
scan 'player' ,{STARTROW=>'001',FILTER=>"InclusiveStopFilter('binary:001')"}
```

```
hbase(main):035:0> scan 'player' ,{STARTROW=>'001',FILTER=>"InclusiveStopFilter('binary:001')"}
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0430 seconds
```

列族和列过滤器

```
scan 'player' ,FILTER=>"FamilyFilter(=,'substring:basic')"
```

```
hbase(main):037:0> scan 'player' ,FILTER=>"FamilyFilter(=,'substring:basic')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0450 seconds
```

```
scan 'player' ,FILTER=>"QualifierFilter(=,'substring:name')"
```

```
hbase(main):038:0> scan 'player' ,FILTER=>"QualifierFilter(=,'substring:name')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0460 seconds
```

```
scan 'player' ,FILTER=>"ColumnPrefixFilter('f')"
```

```
hbase(main):039:0> scan 'player' ,FILTER=>"ColumnPrefixFilter('f')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0400 seconds
```

```
scan 'player' ,FILTER=>"MultipleColumnPrefixFilter('f','l')"
```

```
hbase(main):040:0> scan 'player' ,FILTER=>"MultipleColumnPrefixFilter('f','l')"
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0370 seconds
```

```
scan
'player',{FILTER=>"TimestampsFilter(1573272378490,1573273081747)"}
```

```
hbase(main):045:0> scan 'player',{FILTER=>"TimestampsFilter(1573272378490,1573273081747)"}
ROW                           COLUMN+CELL
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
2 row(s) in 0.0210 seconds
```

```
scan 'player',{FILTER=>"ColumnRangeFilter('f',false,'lastname',true)"}
```

```
hbase(main):058:0> scan 'player',{FILTER=>"ColumnRangeFilter('f',false,'lastname',true)"}
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 001                          column=basic:lastname, timestamp=1573272378490, value=tang
 0010                         column=advanced:firstname, timestamp=1573274784312, value=zhou
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0440 seconds
```

```
scan
'player',{FILTER=>"DependentColumnFilter('basic','firstname',false)"}
```

```
hbase(main):059:0> scan 'player',{FILTER=>"DependentColumnFilter('basic','firstname',false)"}
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 003                          column=basic:firstname, timestamp=1573273081747, value=zou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
5 row(s) in 0.0120 seconds
```

值过滤器

```
scan 'player',{FILTER=>"ValueFilter(=,'binary:zhou')"}
```

```
hbase(main):060:0> scan 'player',{FILTER=>"ValueFilter(=,'binary:zhou')"}
ROW                           COLUMN+CELL
 0010                         column=advanced:firstname, timestamp=1573274784312, value=zhou
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
2 row(s) in 0.0670 seconds
```

```
scan
'player',{COLUMN=>'basic:firstname',FILTER=>"SingleColumnValueFilter(
'basic','firstname',=,'binary:ding')"}
```

```
hbase(main):062:0> scan 'player',{COLUMN=>'basic:firstname',FILTER=>"SingleColumnValueFilter('basic','firstname',
=,'binary:ding')"}
ROW                           COLUMN+CELL
 001                          column=basic:firstname, timestamp=1573272414116, value=ding
1 row(s) in 0.0380 seconds
```

其他过滤器

```
scan 'player',FILTER=>"ColumnPrefixFilter('first') AND ValueFilter(=,'substring:zh')"
```

```
hbase(main):004:0> scan 'player',FILTER=>"ColumnPrefixFilter('first') AND ValueFilter(=,'substring:zh')"
ROW                           COLUMN+CELL
 0010                         column=advanced:firstname, timestamp=1573274784312, value=zhou
 0010                         column=basic:firstname, timestamp=1573272523398, value=zhou
 002                          column=basic:firstname, timestamp=1573272508193, value=zhao
 10010                        column=basic:firstname, timestamp=1573272913875, value=zhou
3 row(s) in 0.2650 seconds
```

快照操作

建立快照

```
snapshot 'player' ,'pl'
```

```
hbase(main):005:0> snapshot 'player' ,'pl'
0 row(s) in 0.4070 seconds
```

显示快照列表

```
list_snapshots
```

```
hbase(main):020:0> list_snapshots
SNAPSHOT                      TABLE + CREATION TIME
 pl                           player (Thu Nov 14 08:24:56 +0800 2019)
1 row(s) in 0.0310 seconds

=> ["pl"]
```

删除快照

```
delete_snapshot 'pl'
```

```
hbase(main):012:0> delete_snapshot 'pl'
0 row(s) in 0.1500 seconds
```

通过快照生成新表

```
clone_snapshot 'pl','play_1'
```

```
hbase(main):021:0> clone_snapshot 'pl','play_1'
0 row(s) in 0.7900 seconds

hbase(main):022:0> list
TABLE
NEWTABLE
play_1
player
3 row(s) in 0.0130 seconds

=> ["NEWTABLE", "play_1", "player"]
```

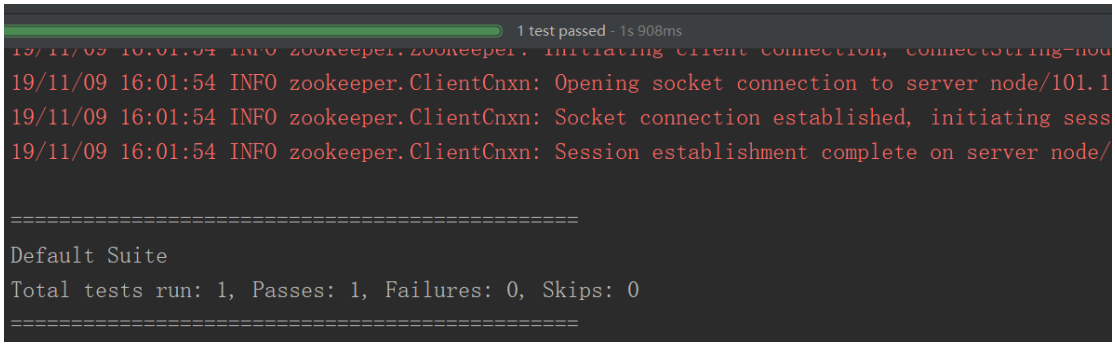
Java 访问 Hbase

pom.xml

```
        <dependency>
            <groupId>org.apache.hbase</groupId>
            <artifactId>hbase-client</artifactId>
            <version>1.4.8</version>
        </dependency>
```

建立连接

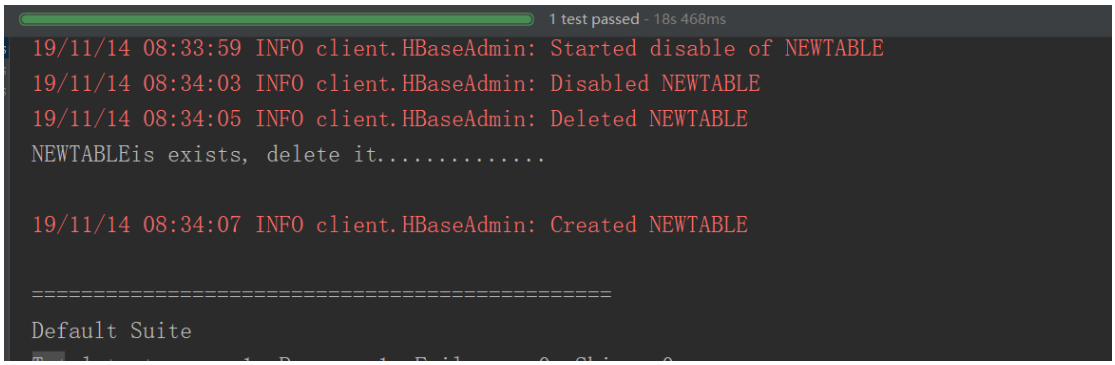
```
public static Configuration conf;
public static Connection connection;
public void getconncet(){
        conf= HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum","node");
        conf.set("hbase.zookeeper.property.clientPort","2181");
        conf.set("zookeeper.znode.parent","/hbase");
        conf.set("hbase.master", "node:16000");
        try{
            connection=ConnectionFactory.createConnection(conf);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```



建立和删除表

```
public void createtable() throws IOException {
        getconncet();
```

```
        TableName tableName =TableName.valueOf("NEWTABLE");
        Admin admin = connection.getAdmin();
        if (admin.tableExists(tableName)){
            admin.disableTable(tableName);
            admin.deleteTable(tableName);
            System.out.println(tableName.toString()+"is exists, delete it.....");
        }
        HTableDescriptor descriptor = new HTableDescriptor(tableName);
        HColumnDescriptor columnDescriptor = new HColumnDescriptor("cf1");
        columnDescriptor.setBloomFilterType(BloomType.ROWCOL);
        descriptor.addFamily(columnDescriptor);
        descriptor.addFamily(new HColumnDescriptor("cf2"));
        admin.createTable(descriptor);
        admin.close();
    }
```




```
hbase(main):004:0> list
TABLE
NEWTABLE
player
2 row(s) in 0.0240 seconds

=> ["NEWTABLE", "player"]
hbase(main):005:0> desc 'NEWTABLE'
Table NEWTABLE is ENABLED
NEWTABLE
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', BLOOMFILTER => 'ROWCOL', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DA
TA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true',
BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'cf2', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_
BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLO
CKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.2180 seconds
```

描述表结构

```
public void desctable() throws IOException {
            getconncet();
            TableName tableName =TableName.valueOf("player");
            Admin admin = connection.getAdmin();
            HTableDescriptor
descriptor=admin.getTableDescriptor(tableName);
            System.out.println(descriptor.toString());


     }
```

```
19/11/09 16:01:03 INFO zookeeper.ClientCnxn: Socket connection established, initiating session, client:
19/11/09 16:01:03 INFO zookeeper.ClientCnxn: Session establishment complete on server node/101.132.96.2
'player', {NAME => 'advanced', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETE

===============================================
Default Suite
Total tests run: 1, Passes: 1, Failures: 0, Skips: 0
===============================================
```

```
public void desctable() throws IOException {
            getconncet();
            TableName tableName =TableName.valueOf("NEWTABLE");
            Admin admin = connection.getAdmin();
            HTableDescriptor
descriptor=admin.getTableDescriptor(tableName);
            System.out.println(descriptor.toString());
            System.out.println("table information:.....");

System.out.println("getNameAsString:"+descriptor.getNameAsString());

System.out.println("getMaxFileSize:"+descriptor.getMaxFileSize());

System.out.println("getMemStoreFlushSize:"+descriptor.getMemStoreFlush
Size());

System.out.println("getRegionSplitPolicyClassName:"+descriptor.getRegion
SplitPolicyClassName());
```

```
System.out.println("getRegionSplitPolicyClassName:"+descriptor.getRegion
SplitPolicyClassName());

Collection<HColumnDescriptor>families=descriptor.getFamilies();
            System.out.println("Column family infomation.....................");
            for (HColumnDescriptor result:families){

System.out.println("getNameAsString:"+result.getNameAsString());

System.out.println("getBloomFilterType:"+result.getBloomFilterType());
                    System.out.println("getBlocksize:"+result.getBlocksize());

System.out.println("getMaxVersions:"+result.getMaxVersions());

System.out.println("getMinVersions:"+result.getMinVersions());
            }
            admin.close();
      }
```

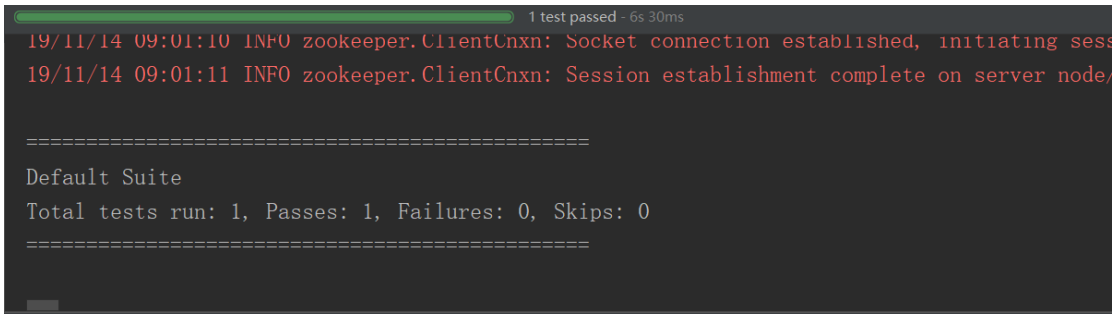
```
1 test passed - 4s 897ms
'NEWTABLE', {NAME => 'cf1', BLOOMFILTER => 'ROWCOL', VERSIONS => '1', IN_MEMORY => 'fa
table information:.....................
getNameAsString:NEWTABLE
getMaxFileSize:-1
getMemStoreFlushSize:-1
getRegionSplitPolicyClassName:null
getRegionSplitPolicyClassName:null
Column family infomation...................
getNameAsString:cf1
getBloomFilterType:ROWCOL
getBlocksize:65536
getMaxVersions:1
getMinVersions:0
getNameAsString:cf2
getBloomFilterType:ROW
getBlocksize:65536
getMaxVersions:1
getMinVersions:0

==============================================
Default Suite
```

数据更新

```
public void addData() throws IOException{
```

```
            getconncet();
            HTable
table=(HTable)connection.getTable(TableName.valueOf("NEWTABLE"));
            table.setWriteBufferSize(6*1024*1024);
            table.setAutoFlushTo(false);
            Put put=new Put(Bytes.toBytes("row1"));
            put.setDurability(Durability.SKIP_WAL);

put.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col0"),Bytes.toBytes(
"value0"));

put.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col1"),Bytes.toBytes("
value1"));

put.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col2"),Bytes.toBytes(
"value2"));

put.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col3"),Bytes.toBytes(
"value3"));
            table.put(put);
            table.flushCommits();
            Put put2=new Put("row2".getBytes());
            put2.setDurability(Durability.SKIP_WAL);

put2.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col0"),Bytes.toBytes
("value4"));

put2.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col4"),Bytes.toBytes
("value5"));

put2.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col3"),Bytes.toBytes
("value6"));

put2.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col5"),Bytes.toBytes
("value7"));
            table.put(put2);
            table.flushCommits();
            Put put3=new Put("row3".getBytes());
```

```
        put3.setDurability(Durability.SKIP_WAL);

put3.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col0"),Bytes.toBytes
("value4"));

put3.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col6"),Bytes.toBytes
("value8"));

put3.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col3"),Bytes.toBytes
("value9"));

put3.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col7"),Bytes.toBytes
("value10"));

        Put put4=new Put("row4".getBytes());
        put4.setDurability(Durability.SKIP_WAL);

put4.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col0"),Bytes.toBytes
("value11"));

put4.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col8"),Bytes.toBytes
("value8"));

put4.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col3"),Bytes.toBytes
("value9"));

put4.addColumn(Bytes.toBytes("cf2"),Bytes.toBytes("col9"),Bytes.toBytes
("value12"));

        List<Put>putList=new ArrayList<Put>();
        putList.add(put3);
        putList.add(put4);
        table.put(putList);
        table.flushCommits();
        table.close();
    }
```

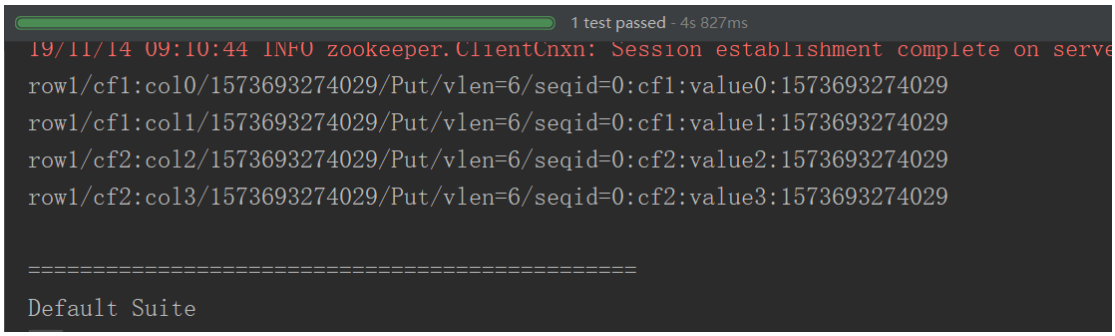



```
hbase(main):023:0> scan 'NEWTABLE'
ROW                           COLUMN+CELL
 row1                         column=cf1:col0, timestamp=1573693274029, value=value0
 row1                         column=cf1:col1, timestamp=1573693274029, value=value1
 row1                         column=cf2:col2, timestamp=1573693274029, value=value2
 row1                         column=cf2:col3, timestamp=1573693274029, value=value3
 row2                         column=cf1:col0, timestamp=1573693274379, value=value4
 row2                         column=cf1:col4, timestamp=1573693274379, value=value5
 row2                         column=cf2:col3, timestamp=1573693274379, value=value6
 row2                         column=cf2:col5, timestamp=1573693274379, value=value7
 row3                         column=cf1:col0, timestamp=1573693274740, value=value4
 row3                         column=cf1:col6, timestamp=1573693274740, value=value8
 row3                         column=cf2:col3, timestamp=1573693274740, value=value9
 row3                         column=cf2:col7, timestamp=1573693274740, value=value10
 row4                         column=cf1:col0, timestamp=1573693274740, value=value11
 row4                         column=cf1:col8, timestamp=1573693274740, value=value8
 row4                         column=cf2:col3, timestamp=1573693274740, value=value9
 row4                         column=cf2:col9, timestamp=1573693274740, value=value12
4 row(s) in 0.0520 seconds
```

数据查询

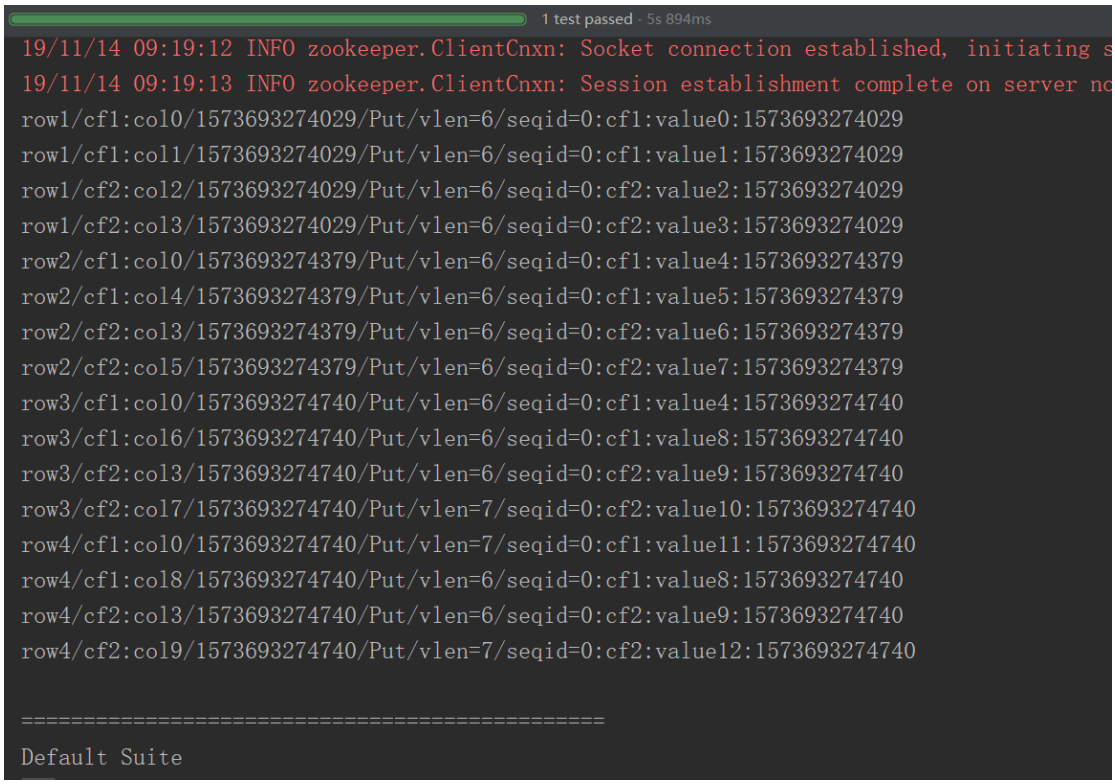
get 方法

```
private void getData() throws IOException{
        getconncet();
        Table                                                    table
=connection.getTable(TableName.valueOf("NEWTABLE"));
        Get get=new Get(Bytes.toBytes("row1"));
        Result result=table.get(get);
        for (Cell cell:result.rawCells()){
            System.out.println(new
String(CellUtil.getCellKeyAsString(cell))+":"+new
String(CellUtil.cloneFamily(cell))+":"+new
String(CellUtil.cloneValue(cell))+":"+cell.getTimestamp());
        }
        table.close();
    }
```



scan 方法

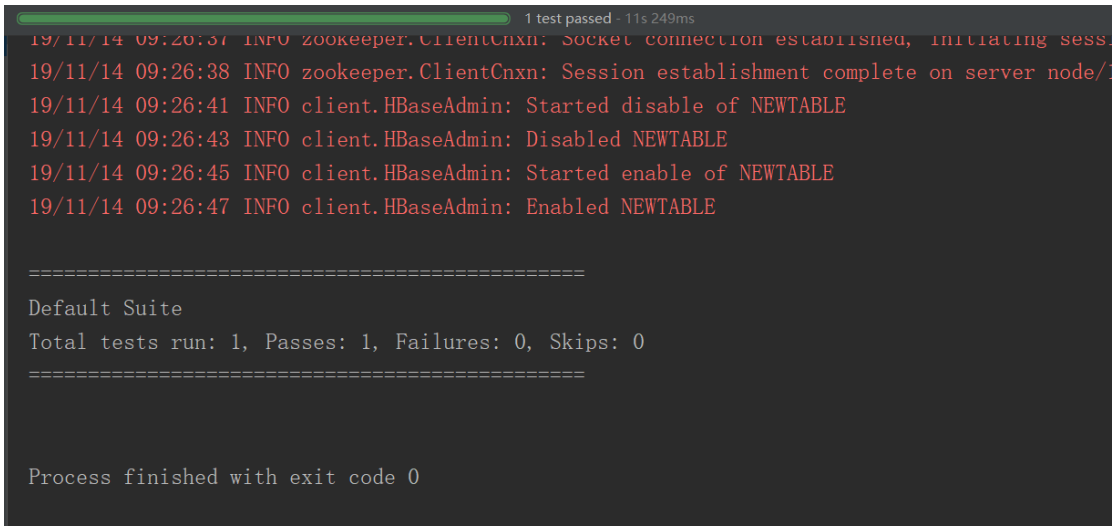
```
private void ScanData()throws IOException{
        getconncet();
        Table                                                    table
=connection.getTable(TableName.valueOf("NEWTABLE"));
        Scan scan=new Scan();
        ResultScanner results=table.getScanner(scan);
        for (Result result:results){
            for (Cell cell:result.rawCells()){
                System.out.println(new
String(CellUtil.getCellKeyAsString(cell))+":"+new
String(CellUtil.cloneFamily(cell))+":"+new
String(CellUtil.cloneValue(cell))+":"+cell.getTimestamp());
            }
        }
    }
```



删除行和列

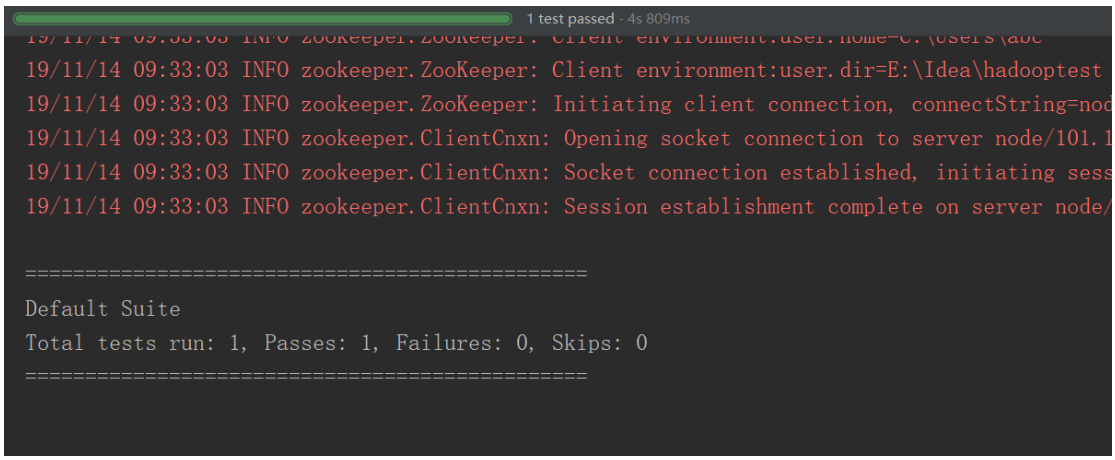
删除列族和列

```
private void removecol()throws IOException{
getconncet();
        Admin admin=connection.getAdmin();
        HTableDescriptor
descriptor=admin.getTableDescriptor(TableName.valueOf("NEWTABLE"));
        TableName tableName =TableName.valueOf("NEWTABLE");
        descriptor.removeFamily(Bytes.toBytes("col0"));
        admin.disableTable(tableName);
        admin.deleteColumn(tableName, Bytes.toBytes("cf2"));
        admin.enableTable(tableName);
        admin.close();
    }
```



删除行和键值对

```
public void deleteRow()throws IOException{
            getconncet();
            HTable table=null;
            try {

table=(HTable)connection.getTable(TableName.valueOf("NEWTABLE"));
                    Delete delete1=new Delete(Bytes.toBytes("row1"));
                    Delete delete2=new Delete(Bytes.toBytes("row2"));
                    Delete delete3=new Delete(Bytes.toBytes("row3"));
                    delete2.addFamily(Bytes.toBytes("cf1"));

delete3.addColumn(Bytes.toBytes("cf1"),Bytes.toBytes("col6"));
                    table.delete(delete1);
                    table.delete(delete2);
                    table.delete(delete3);
                    table.close();
            }catch (Exception e){

            }
     }
```



过滤器

```
public void filter()throws IOException{
            getconncet();
            TableName tableName =TableName.valueOf("NEWTABLE");
            Table table=connection.getTable(tableName);
            Scan scan=new Scan();
            FilterList                                          filterList=new
FilterList(FilterList.Operator.MUST_PASS_ALL);
            filterList.addFilter(new
RowFilter(CompareFilter.CompareOp.LESS,new
BinaryComparator(Bytes.toBytes("row3"))));
            filterList.addFilter(new KeyOnlyFilter());
            scan.setFilter(filterList);
            ResultScanner results=table.getScanner(scan);
            for (Result result:results){
                  for (Cell cell:result.rawCells()){
                        System.out.println(new
String(CellUtil.getCellKeyAsString(cell))+":"+new
String(CellUtil.cloneFamily(cell))+":"+new
String(CellUtil.cloneValue(cell))+":"+cell.getTimestamp());
                  }
            }
      }
```

解决 Java API 不能远程访问 HBase 的问题

查看发现 HBase 绑定的是本地 IP：127.0.0.1，这当然访问不了

```
netstat -anp|grep 16000
```

```
[root@localhost ~]# netstat -anp|grep 16000
tcp        0      0 127.0.0.1:16000         0.0.0.0:*               LISTEN      16321/java
tcp        0      0 127.0.0.1:16000         127.0.0.1:35893         ESTABLISHED 16321/java
tcp        0      0 127.0.0.1:35893         127.0.0.1:16000         ESTABLISHED 16412/java
```

配置 Linux 的 hostname

```
vim /etc/sysconfig/network
```

```
NETWORKING=yes
HOSTNAME=master
```

这里配置的 hostname 要 Linux 重启才生效，为了不重启就生效，我们可以执行：hostname master 命令，暂时设置 hostname

```
# Created by anaconda
NETWORKING=yes
HOSTNAME=master
```

配置 Linux 的 hosts，映射 ip 的 hostname 的关系

```
vi /etc/hosts
```

```
172.19.71.150   master
```

```
#::1     localhost       localhost.localdomain
#127.0.0.1      localhost       localhost.local

172.19.71.150  master  node01 localhost
```

```
netstat -anp|grep 16000
```

```
[root@localhost ~]# netstat -anp|grep 16000
tcp        0      0 172.19.71.150:16000     0.0.0.0:*               LISTEN      7788/java
tcp        0      0 172.19.71.150:16000     172.19.71.150:4069      ESTABLISHED 7788/java
tcp        0      0 172.19.71.150:4069      172.19.71.150:16000     ESTABLISHED 7941/java
```

配置访问 windows 的 hosts

路径为：C:\Windows\System32\drivers\etc\hosts

```
172.19.71.150   master
```

配置完这三项 Java API 就可以远程访问 HBase 了，切记最后配置 windows 的 hosts 也是必须的

Python 访问 Hbase

CentOS 安装 Thrift

安装依赖

```
yum -y install automake libtool flex bison pkgconfig gcc-c++ boost-devel libevent-devel zlib-devel python-devel ruby-devel openssl-devel
```

安装 boost 包

```
cd /export/softwares/
wget http://sourceforge.net/projects/boost/files/boost/1.53.0/boost_1_53_0.tar.gz
tar xvf boost_1_53_0.tar.gz
```

```
cd boost_1_53_0
./bootstrap.sh
./b2 install
```

安装 thrift

升级 bison:

```
wget http://ftp.gnu.org/gnu/bison/bison-2.5.1.tar.gz
tar xvf bison-2.5.1.tar.gz
cd bison-2.5.1
./configure
make
make install
```

```
wget http://mirrors.hust.edu.cn/apache/thrift/0.9.3/thrift-0.9.3.tar.gz
tar xzvf thrift-0.9.3.tar.gz
cd thrift-0.9.3
./configure
make
```

```
make install
```

验证是否安装成功

```
thrift -version
```

打开 HBase 的 Thrift 服务

```
hbase-daemon.sh start thrift
hbase-daemon.sh start thrift2
```

客户端配置 Python 环境 pip 安装 Thrift

```
conda create --name nosql python=3.7
conda activate nosql
conda install thrift
pip install hbase-thrift
```

修改代码文件

将 Python2 风格代码改为 Python3

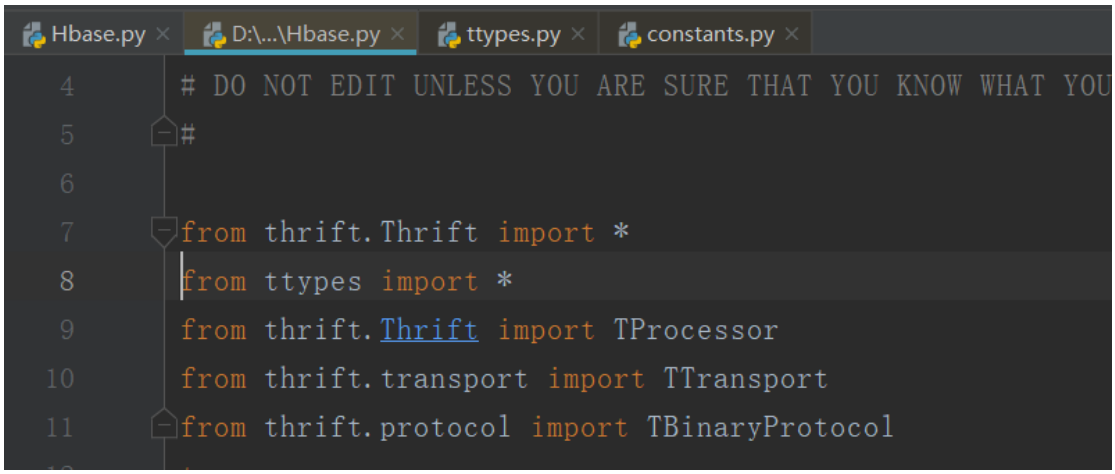
```
Traceback (most recent call last):
  File "E:/PycharmWorkspaces/NoSql/Hbase.py", line 3, in <module>
    from hbase import Hbase
  File "D:\ProgramData\Anaconda3\envs\nosql\lib\site-packages\hbase\Hbase.py", line 2066
    except IOError, io:
                  ^
SyntaxError: invalid syntax
```

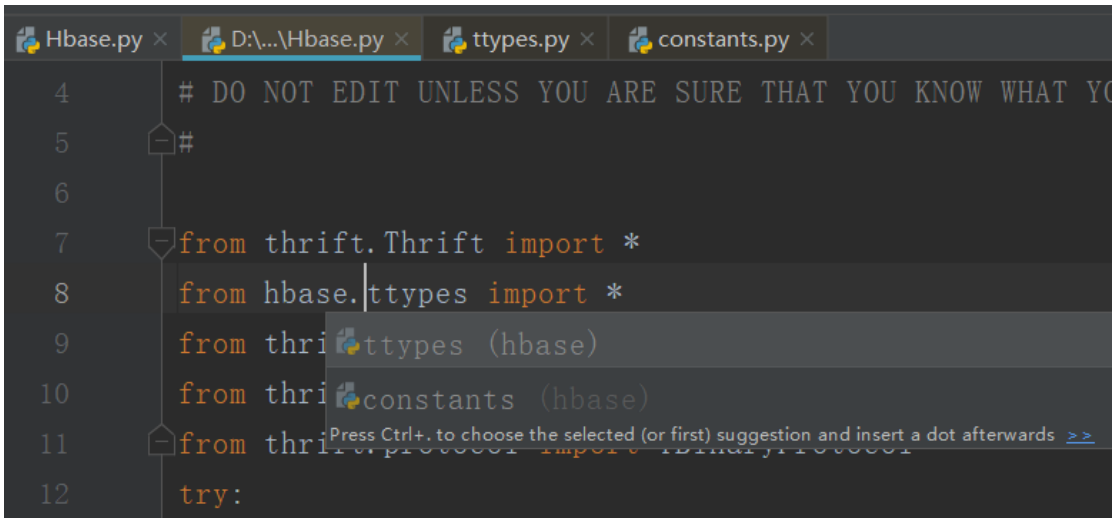
改为

```
try:
    result.success = self._handler.getVer(args.
except IOError as io:
    result.io = io
```

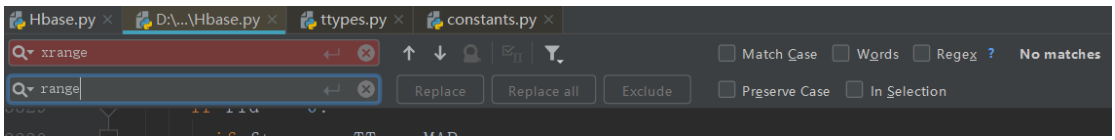
```
Traceback (most recent call last):
  File "E:/PycharmWorkspaces/NoSql/Hbase.py", line 3, in <module>
    from hbase import Hbase
  File "D:\ProgramData\Anaconda3\envs\nosql\lib\site-packages\hbase\Hbase.py", line 8, in <module>
    from ttypes import *
ModuleNotFoundError: No module named 'ttypes'
```

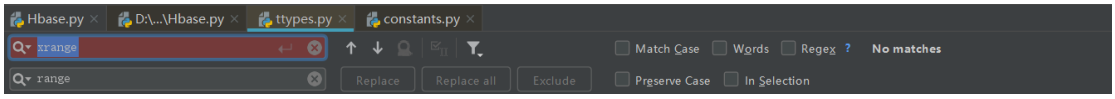
改为

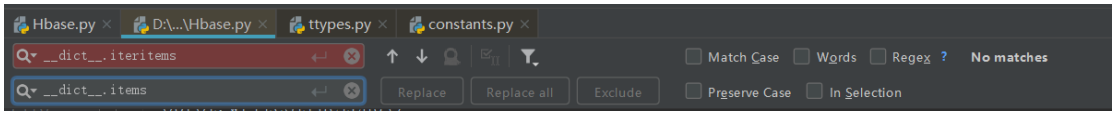


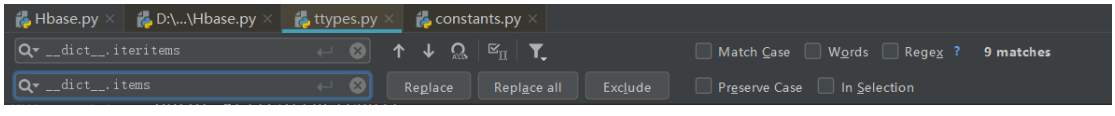


替换处理









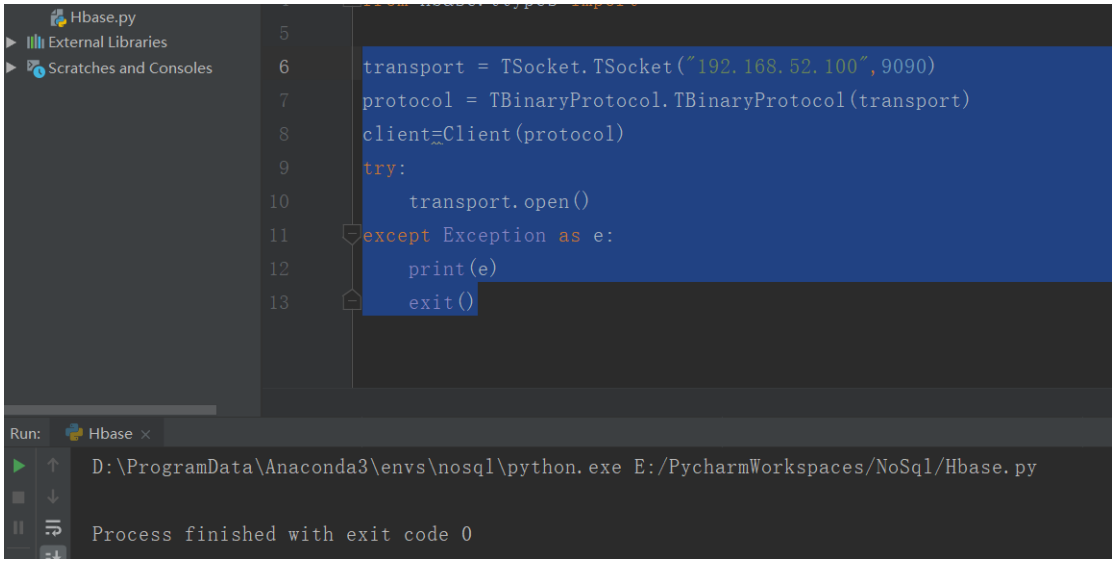
引用的类库

```
from thrift.transport import TSocket
```

```
from thrift.protocol import TBinaryProtocol
from hbase.Hbase import *
from hbase.ttypes import *
```

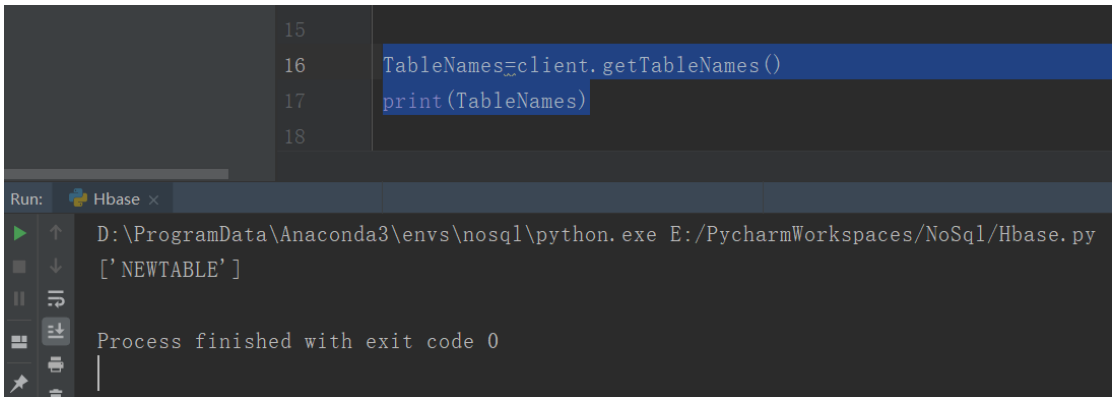
建立连接

```
transport = TSocket.TSocket("192.168.52.100",9090)
protocol = TBinaryProtocol.TBinaryProtocol(transport)
client=Client(protocol)
try:
    transport.open()
except Exception as e:
    print(e)
    exit()
# 关闭连接
transport.close()
```



列举所有表名

```
TableNames=client.getTableNames()
print(TableNames)
```



表的建立

```
content1=ColumnDescriptor(name='cf1',maxVersions=1)
content2=ColumnDescriptor(name='cf2',)
client.createTable('testtable',[content1,content2])
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/Hbase.py
['NEWTABLE', 'testtable']

Process finished with exit code 0
```

表的禁用删除

```
try:
    client.disableTable('testtable')
    client.deleteTable('testtable')
except:
    pass
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/Hbase.py
['NEWTABLE']

Process finished with exit code 0
```

查看表结构

```
ColumnDescriptors=client.getColumnDescriptors('testtable')
print(ColumnDescriptors)
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/Hbase.py
{'cf1:': ColumnDescriptor(name='cf1:', maxVersions=1, compression='NONE', inMemory=False, bloomFilterType='NONE', bloomFilterVectorSize=0, bloomFil

Process finished with exit code 0
```

插入数据

```
mutations=[Mutation(column='cf1:a',value='1'),Mutation(column='cf1:c',value='20')]
client.mutateRow('testtable','row-key1',mutations)
```

检索数据

```
result=client.getRow('testtable','row-key1')
print(result)
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/Hbase.py
[TRowResult(row='row-key1', columns={'cf1:a': TCell(value='1', timestamp=1573911335534), 'cf1:c': TCell(value='20', timestamp=1573911335534)})]

Process finished with exit code 0
```

```
for r in result:
    print('rowkey:',r.row)
    for c in r.columns.keys():
        print("column qualifier:",c)
        print('value:',r.columns[c].value)
        print('timestamp:', r.columns[c].timestamp)
```

```
rowkey: row-key1
column qualifier: cf1:a
value: 1
timestamp: 1573911335534
column qualifier: cf1:c
value: 20
timestamp: 1573911335534
```

删除数据

```
client.deleteAllRow('testtable',row='row-key1')
client.deleteAllRowTs('testtable','row-key1',1573912214382)
```