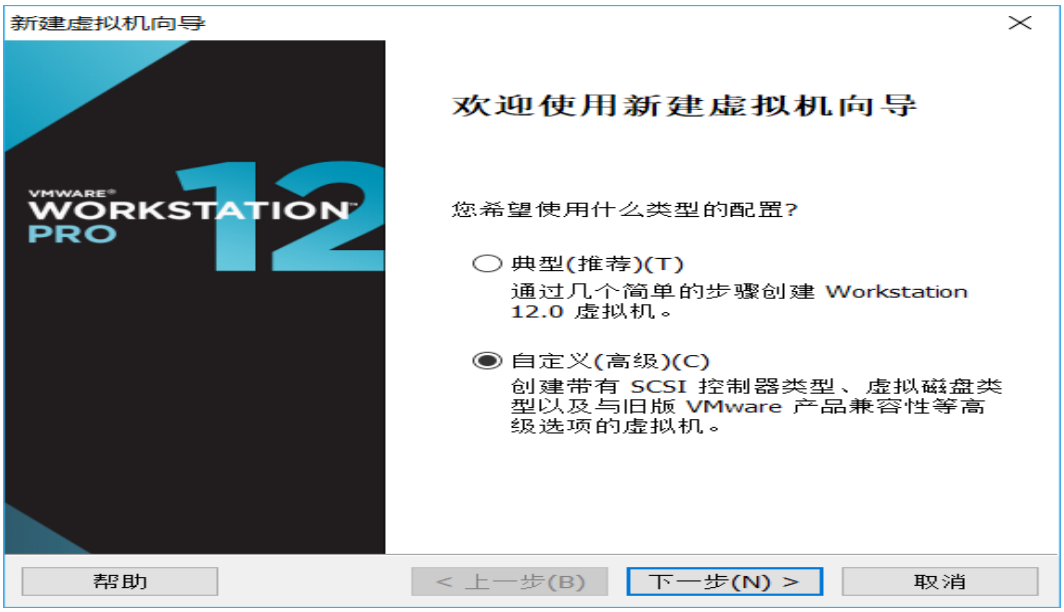
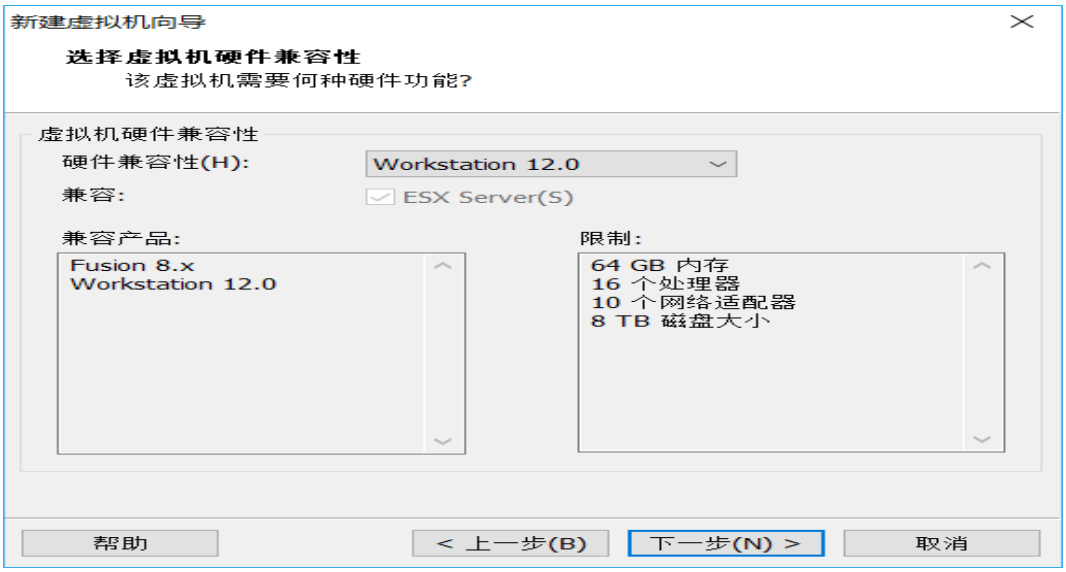
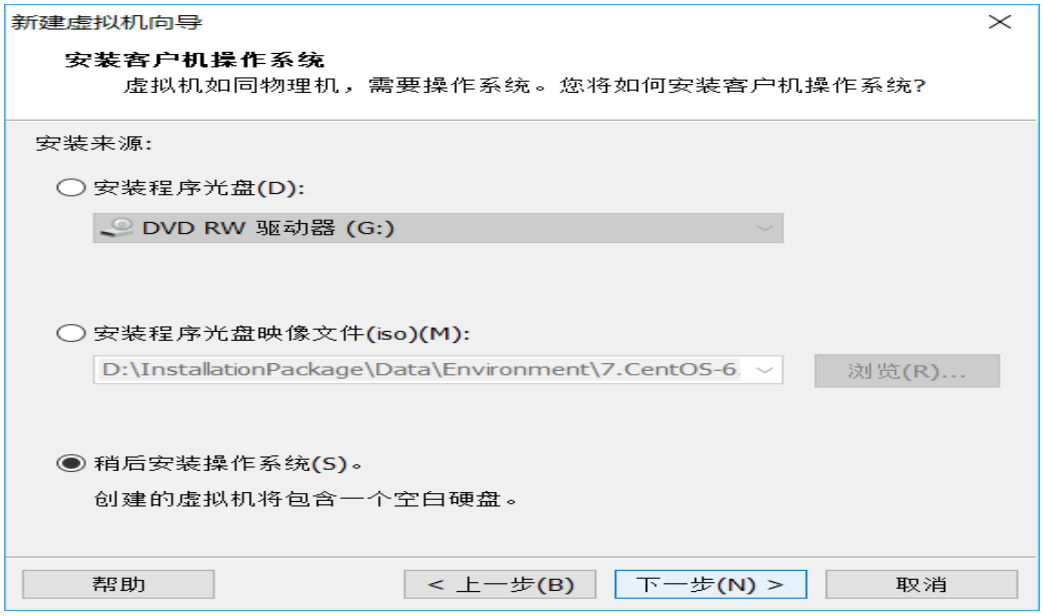
目录

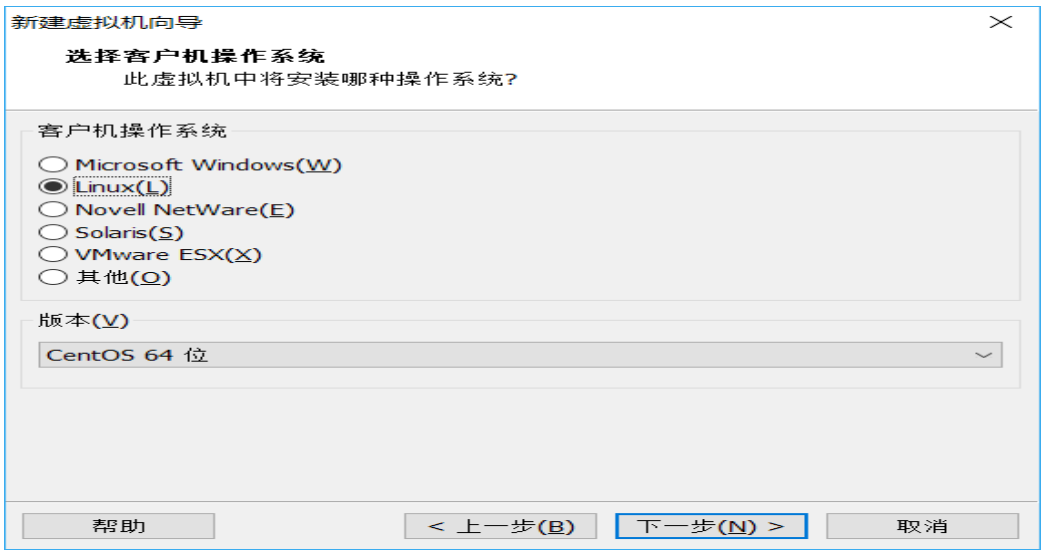
HBase

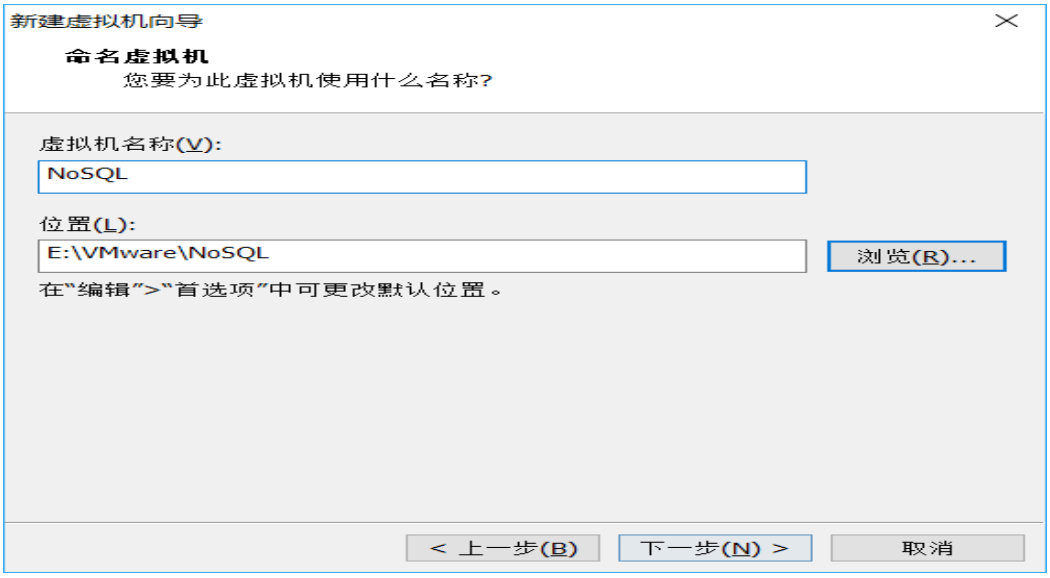
虚拟机安装 CentOS7



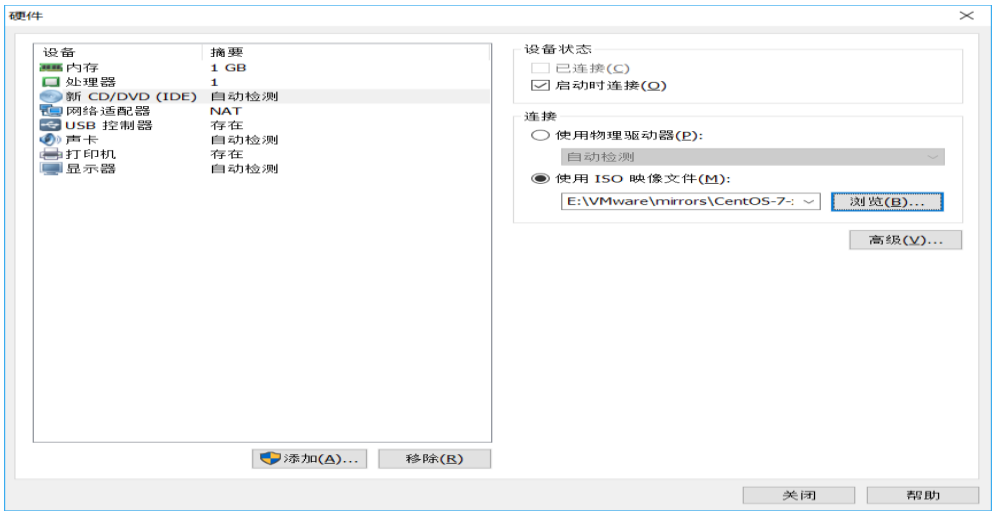

新建虚拟机向导
选择虚拟机硬件兼容性
该虚拟机需要何种硬件功能?
虚拟机硬件兼容性
硬件兼容性(H):
Workstation 12.0
兼容:
ESX Server(S)
兼容产品:
Fusion 8.x
Workstation 12.0
限制:
64 GB 内存
16 个处理器
10 个网络适配器
8 TB 磁盘大小
帮助
< 上一步(B)
下一步(N) >
取消


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

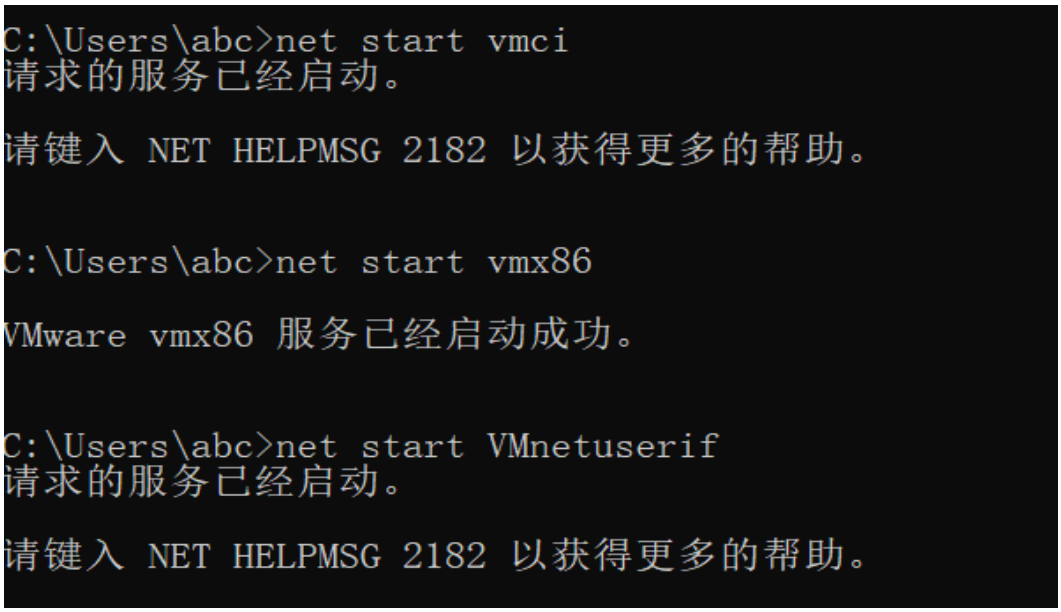

新建虚拟机向导
命名虚拟机
您要为此虚拟机使用什么名称?
虚拟机名称(V):
NoSQL
位置(L):
E:\VMware\NoSQL
浏览(R)...
在"编辑">"首选项"中可更改默认位置。
< 上一步(B)
下一步(N) >
取消


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

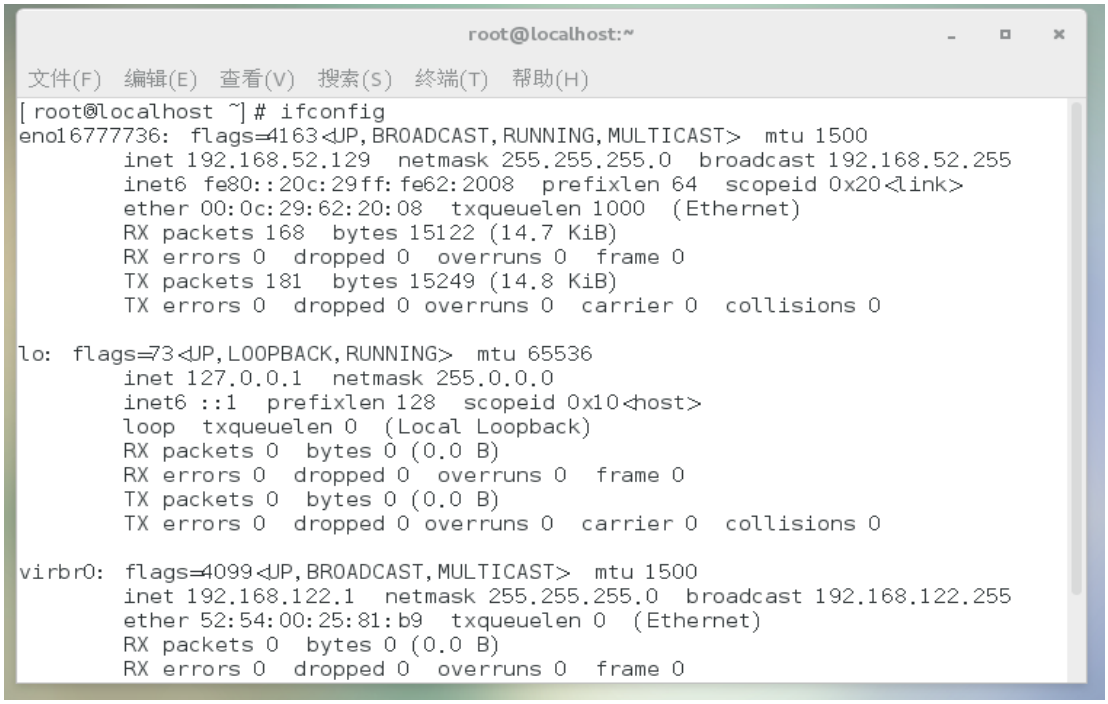
原创 无法打开内核设备“\\.\Global\vmx86”：系统找不到指定的文件。是否在安装VMwar

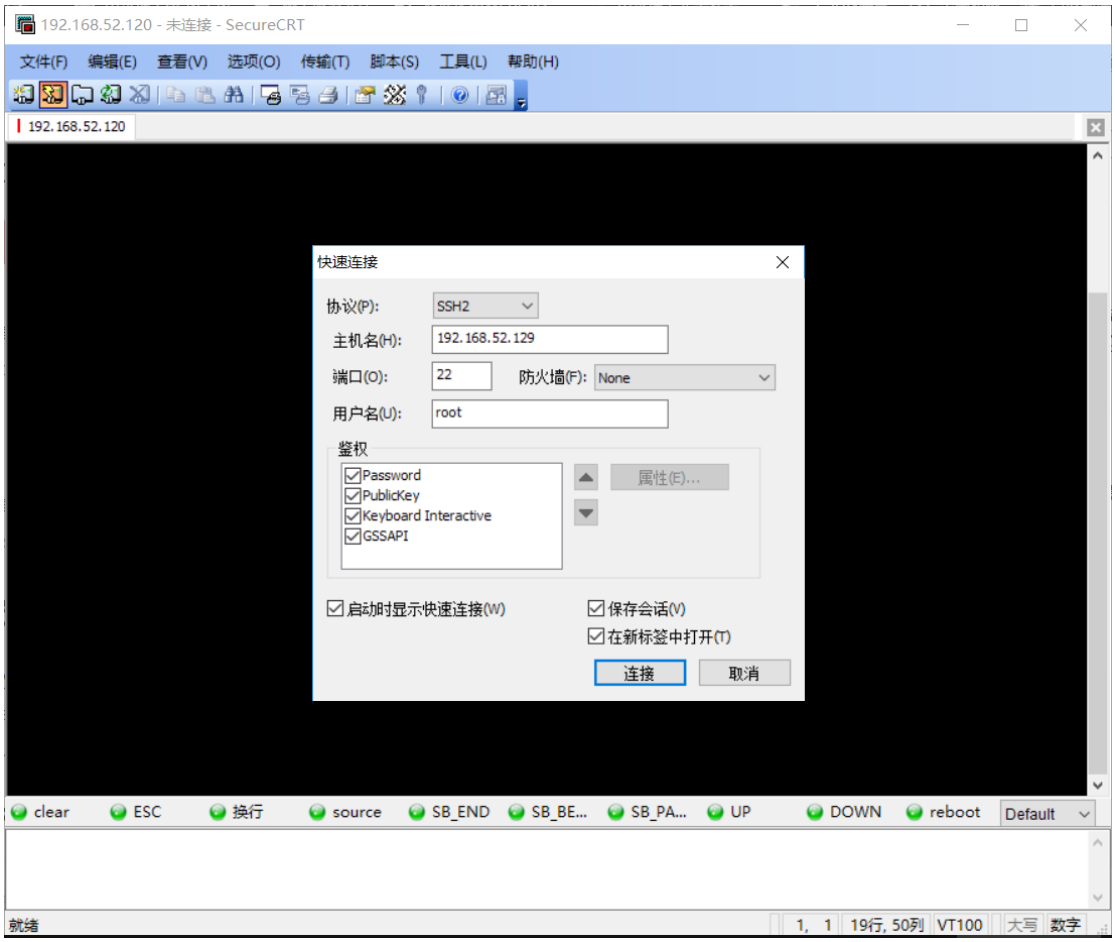
1. 点击“开始→运行”，在运行框中输入 cmd 回车打开命令提示符，然后依次执行以下命令
2. 输入以下的命令并回车
   net start vmci
   net start vmx86
   net start VMnetuserif
3. 改变vmware几种服务的启动方式为：
   sc config vmci start= auto
   sc config vmx86 start= auto
   sc config VMnetuserif start= auto
   这一点儿与win7下面的有所不同，特此提醒，win7下面的是：
   sc config vmci=auto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

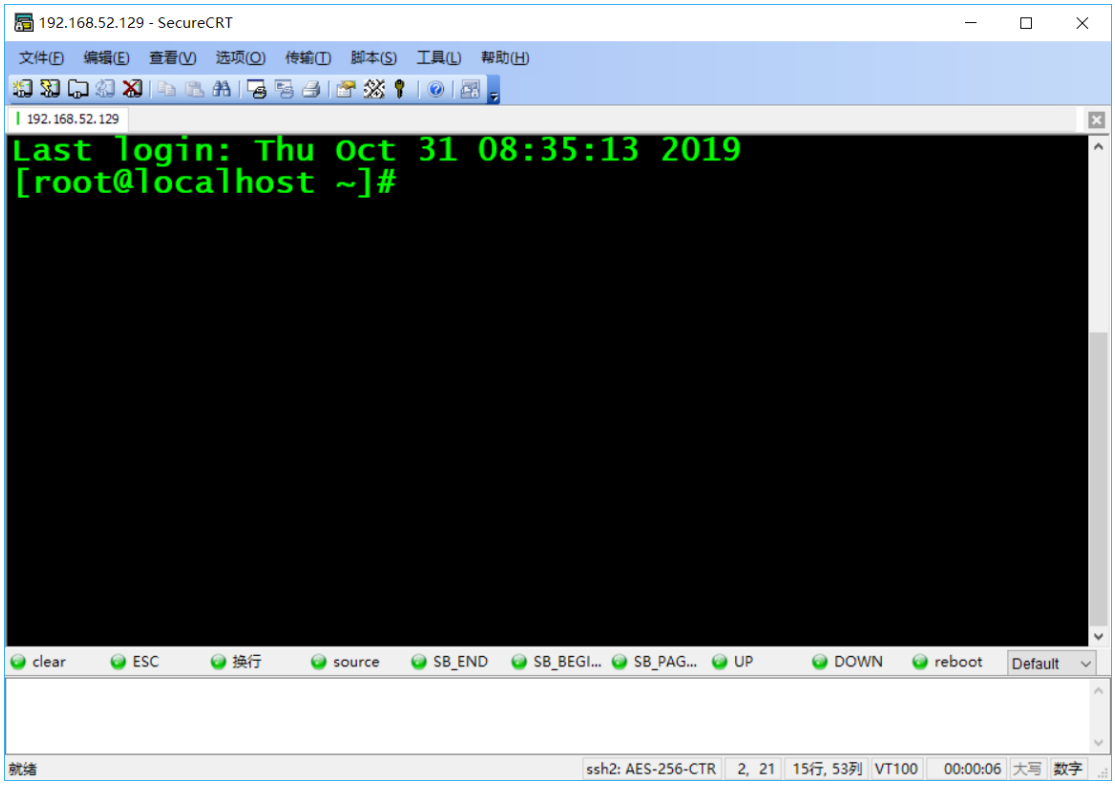


远程连接虚拟机

ifconfig




192.168.52.120 - 未连接 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
192.168.52.120
快速连接
协议(P): SSH2
主机名(H): 192.168.52.129
端口(O): 22
防火墙(F): None
用户名(U): root
鉴权
Password
PublicKey
Keyboard Interactive
GSSAPI
属性(E)...
启动时显示快速连接(W)
保存会话(V)
在新标签中打开(T)
连接
取消
clear ESC 换行 source SB_END SB_BE... SB_PA... UP DOWN reboot Default
就绪
1, 1 19行, 50列 VT100 大写 数字


192.168.52.129 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
192.168.52.129
Last login: Thu Oct 31 08:35:13 2019
[root@localhost ~]#
clear ESC 换行 source SB_END SB_BEGI... SB_PAG... UP DOWN reboot Default
就绪
ssh2: AES-256-CTR 2, 21 15行, 53列 VT100 00:00:06 大写 数字

HDFS 伪分布式搭建

查看自带的 openjdk

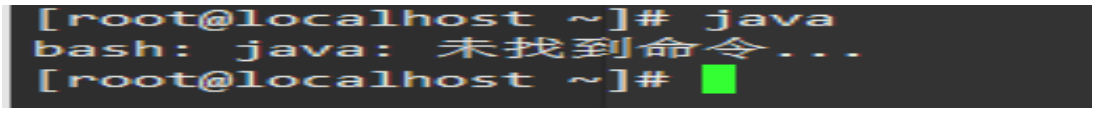
```
rpm -qa | grep java
```

```
[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
```

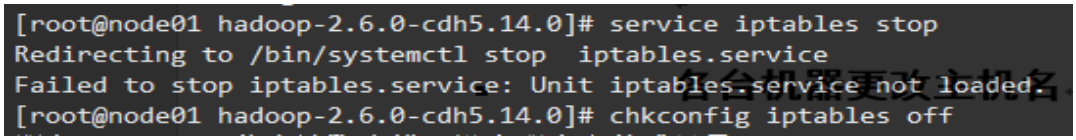
```
[root@localhost ~]# rpm -e java-1.8.0-openjdk-headless-1.8.0.65-3.b17.el7.x86_64 libvirt
-java-0.4.9-4.el7.noarch java-1.7.0-openjdk-devel-1.7.0.91-2.6.2.3.el7.x86_64 java-1.6.0
-openjdk-1.6.0.36-1.13.8.1.el7_1.x86_64 javamail-1.4.6-8.el7.noarch nuxwdog-client-java-
1.0.3-2.el7.x86_64 java-1.7.0-openjdk-1.7.0.91-2.6.2.3.el7.x86_64 javassist-3.16.1-10.el
7.noarch java-1.6.0-openjdk-devel-1.6.0.36-1.13.8.1.el7_1.x86_64 java-1.7.0-openjdk-head
less-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.e
l7.x86_64 python-javapackages-3.4.1-11.el7.noarch java-1.8.0-openjdk-devel-1.8.0.65-3.b1
7.el7.x86_64 --nodeps
```



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
```



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

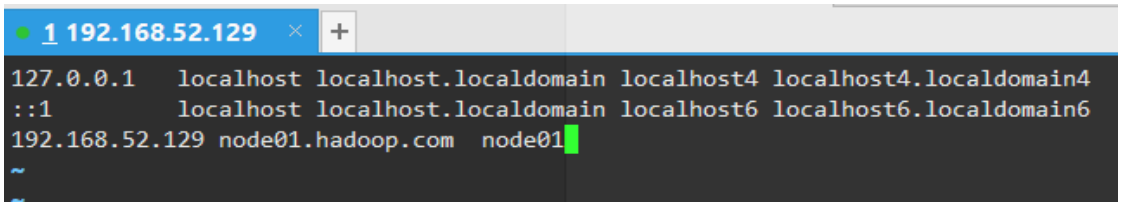
```
vim /etc/sysconfig/network
```


```
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
```

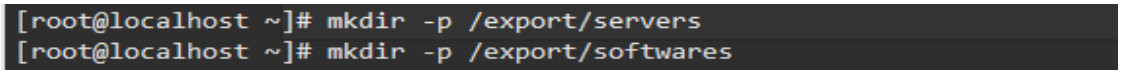


各台机器重启

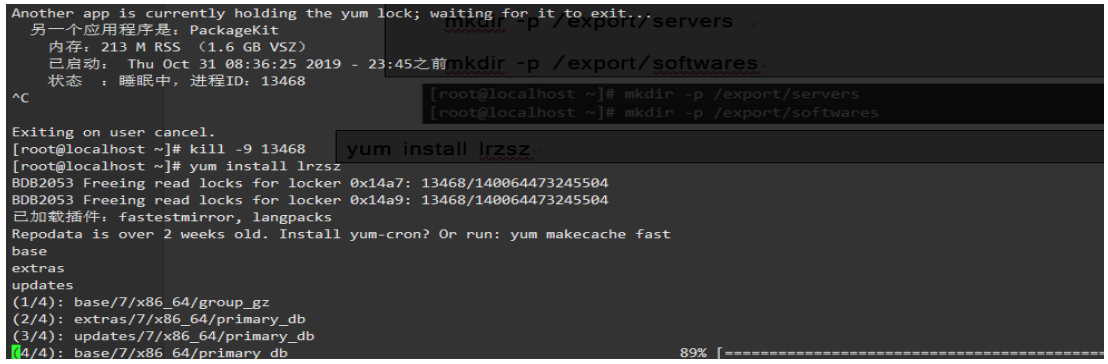
```
reboot  -h  now
```

创建文件夹

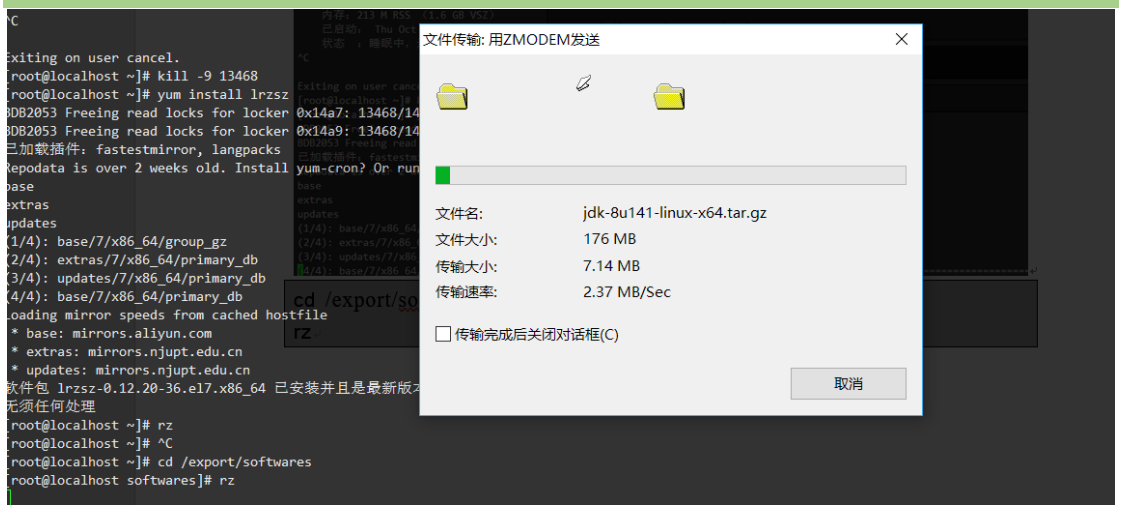
```
mkdir -p /export/servers

mkdir -p /export/softwares
```




```
yum install lrzsz
```



```
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/
```

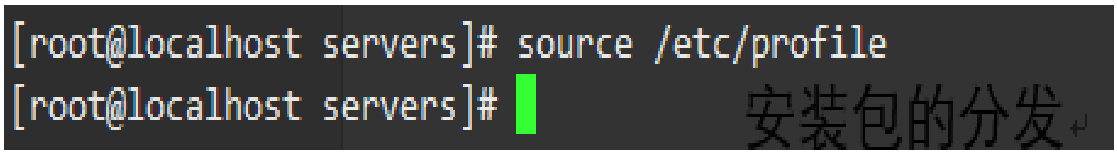
```
[root@localhost softwares]# cd ..
[root@localhost export]# ll
总用量 0
drwxr-xr-x. 3 root root 25 10月 31 09:05 servers
drwxr-xr-x. 2 root root 39 10月 31 09:03 softwares
[root@localhost export]# cd servers/
[root@localhost servers]# ll
总用量 4
drwxr-xr-x. 8 10 143 4096 7月  12 2017 jdk1.8.0_141
[root@localhost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


```
source /etc/profile
```



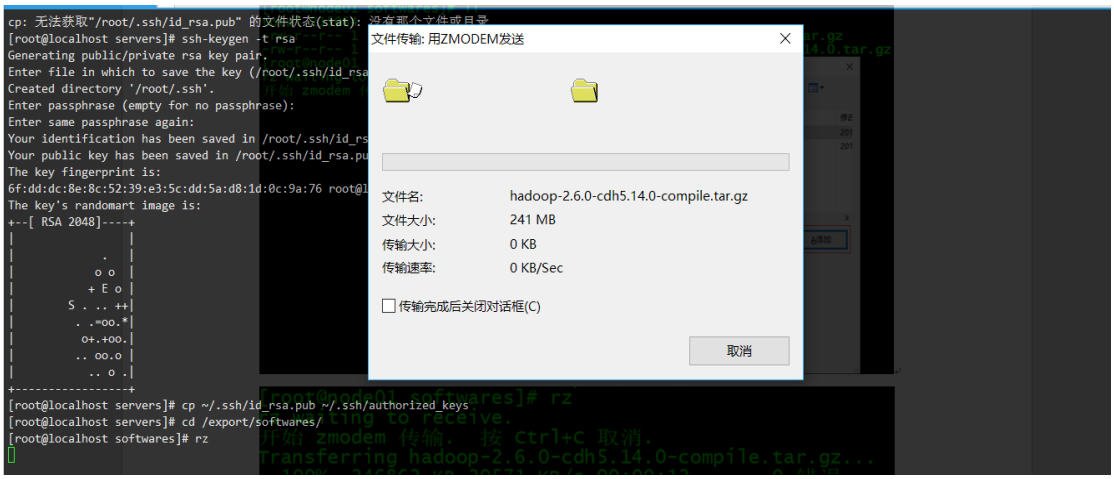

```
[root@localhost servers]# java -version
java version "1.8.0_141"
Java(TM) SE Runtime Environment (build 1.8.0_141-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
[root@localhost servers]#
```

安装 ssh

```
sudo yum install ssh

ssh-keygen -t rsa

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

```
[root@localhost servers]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
6f:dd:dc:8e:8c:52:39:e3:5c:dd:5a:d8:1d:0c:9a:76 root@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|             .   |
|            o o  |
|           + E o |
|        S . .. ++|
|         . .=oo.*|
|          o+.+oo.|
|          .. oo.o |
|            .. o .|
+-----------------+
[root@localhost servers]# cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
[root@localhost servers]#
```

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




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

```
[root@localhost softwares]# ll
总用量 428036
-rw-r--r--. 1 root root 252787404 9月  27 2018 hadoop-2.6.0-cdh5.14.0.tar.gz
-rw-r--r--. 1 root root 185516505 9月  27 2018 jdk-8u141-linux-x64.tar.gz
[root@localhost softwares]#
```

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

```
[root@localhost /]# cd export/servers/
[root@localhost servers]# ll
总用量 8
drwxr-xr-x. 9 root root 4096 5月   8 2018 hadoop-2.6.0-cdh5.14.0
drwxr-xr-x. 8   10  143 4096 7月  12 2017 jdk1.8.0_141
[root@localhost servers]#
```

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

bin/hadoop checknative
```

```
[root@localhost servers]# cd /export/servers/hadoop-2.6.0-cdh5.14.0
[root@localhost hadoop-2.6.0-cdh5.14.0]# bin/hadoop checknative
19/10/31 09:20:53 INFO bzip2.Bzip2Factory: Successfully loaded & initialized native-bzip2 library system-native
19/10/31 09:20:53 INFO zlib.ZlibFactory: Successfully loaded & initialized native-zlib library
Native library checking:
hadoop:  true /export/servers/hadoop-2.6.0-cdh5.14.0/lib/native/libhadoop.so.1.0.0
zlib:    true /lib64/libz.so.1
snappy:  true /lib64/libsnappy.so.1
lz4:     true revision:10301
bzip2:   true /lib64/libbz2.so.1
openssl: true /lib64/libcrypto.so
[root@localhost hadoop-2.6.0-cdh5.14.0]#
```

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
```

```
[root@localhost hadoop]# mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/namenodeDatas
[root@localhost hadoop]# mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/datanodeDatas
[root@localhost hadoop]# mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/nn/edits
[root@localhost hadoop]# mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/snn/name
[root@localhost hadoop]# mkdir -p /export/servers/hadoop-2.6.0-cdh5.14.0/hadoopDatas/dfs/nn/snn/edits
[root@localhost hadoop]#
```

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
```

```
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
-- 插入 --
```

```
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
```

```
[root@node01 hadoop-2.6.0-cdh5.14.0]# sbin/start-dfs.sh
Starting namenodes on [node01]
The authenticity of host 'node01 (192.168.52.129)' can't be established.
ECDSA key fingerprint is b4:51:2f:46:99:63:71:cc:25:1c:d8:fa:28:fa:34:42.
Are you sure you want to continue connecting (yes/no)? yes
node01: Warning: Permanently added 'node01,192.168.52.129' (ECDSA) to the list of known hosts.
node01: starting namenode, logging to /export/servers/hadoop-2.6.0-cdh5.14.0/logs/hadoop-root-namenode-node01.hadoop.com.out
node01: starting datanode, logging to /export/servers/hadoop-2.6.0-cdh5.14.0/logs/hadoop-root-datanode-node01.hadoop.com.out
Starting secondary namenodes [node01]
node01: starting secondarynamenode, logging to /export/servers/hadoop-2.6.0-cdh5.14.0/logs/hadoop-root-secondarynamenode-node01.hadoop.com.out
[root@node01 hadoop-2.6.0-cdh5.14.0]# sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /export/servers/hadoop-2.6.0-cdh5.14.0/logs/yarn-root-resourcemanager-node01.hadoop.com.out
node01: starting nodemanager, logging to /export/servers/hadoop-2.6.0-cdh5.14.0/logs/yarn-root-nodemanager-node01.hadoop.com.out
[root@node01 hadoop-2.6.0-cdh5.14.0]# sbin/mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /export/servers/hadoop-2.6.0-cdh5.14.0/logs/mapred-root-historyserver-node01.hadoop.com.out
[root@node01 hadoop-2.6.0-cdh5.14.0]#
```

```
[root@node01 hadoop-2.6.0-cdh5.14.0]# jps
5329 Jps
5010 JobHistoryServer
4902 NodeManager
4296 DataNode
4073 NameNode
4556 SecondaryNameNode
4751 ResourceManager
[root@node01 hadoop-2.6.0-cdh5.14.0]#
```

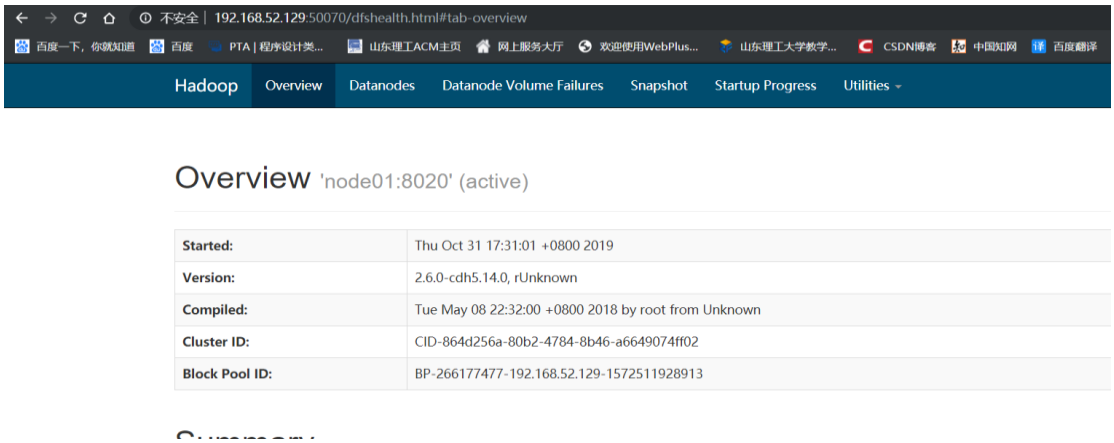
停止集群

```
sbin/stop-dfs.sh

sbin/stop-yarn.sh

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

浏览器查看启动页面

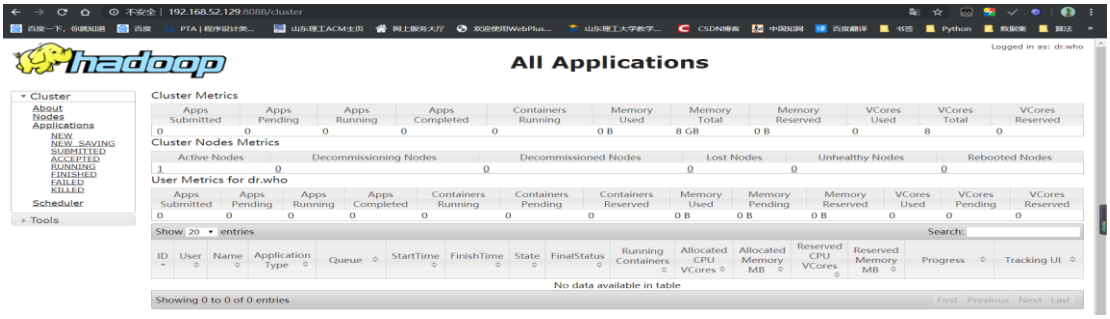
hdfs 集群访问地址

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
```

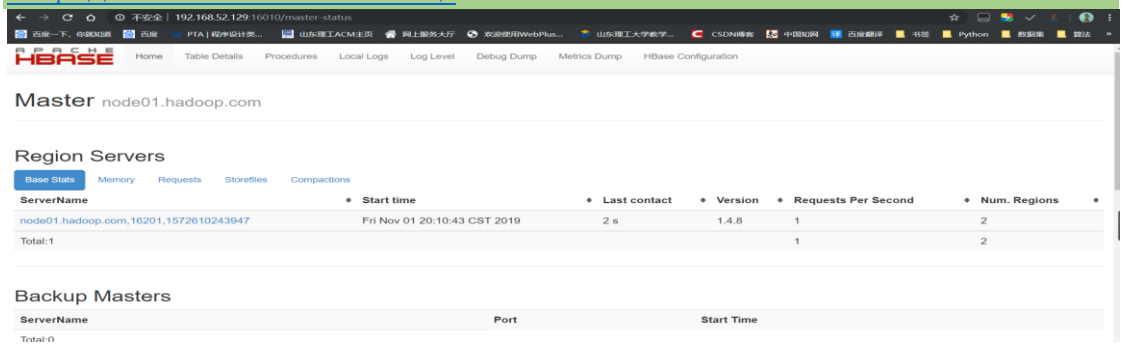
进入 hbase shell

```
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

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

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 数据查询                    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
player' ,FILTER=>"RowFilter(=,'substring:01')"
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"]
```

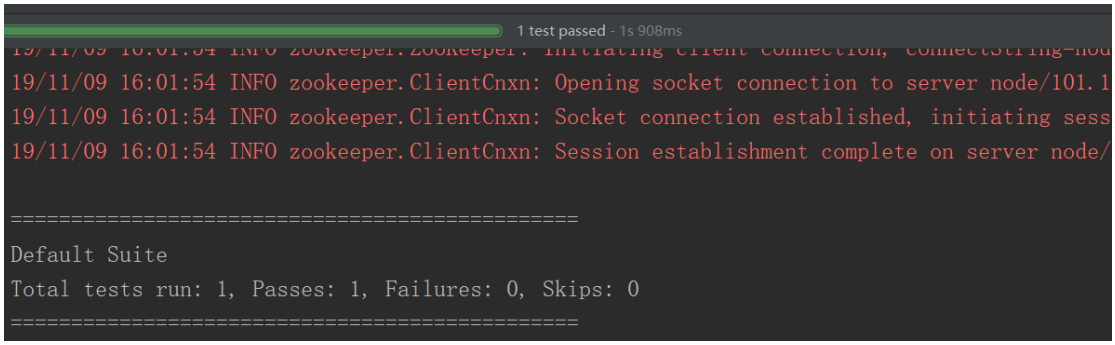
Java 访问 Hbase

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();
    }
```

```
1 test passed - 18s 468ms
19/11/14 08:33:59 INFO client.HBaseAdmin: Started disable of NEWTABLE
19/11/14 08:34:03 INFO client.HBaseAdmin: Disabled NEWTABLE
19/11/14 08:34:05 INFO client.HBaseAdmin: Deleted NEWTABLE
NEWTABLEis exists, delete it.............

19/11/14 08:34:07 INFO client.HBaseAdmin: Created NEWTABLE

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

```
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

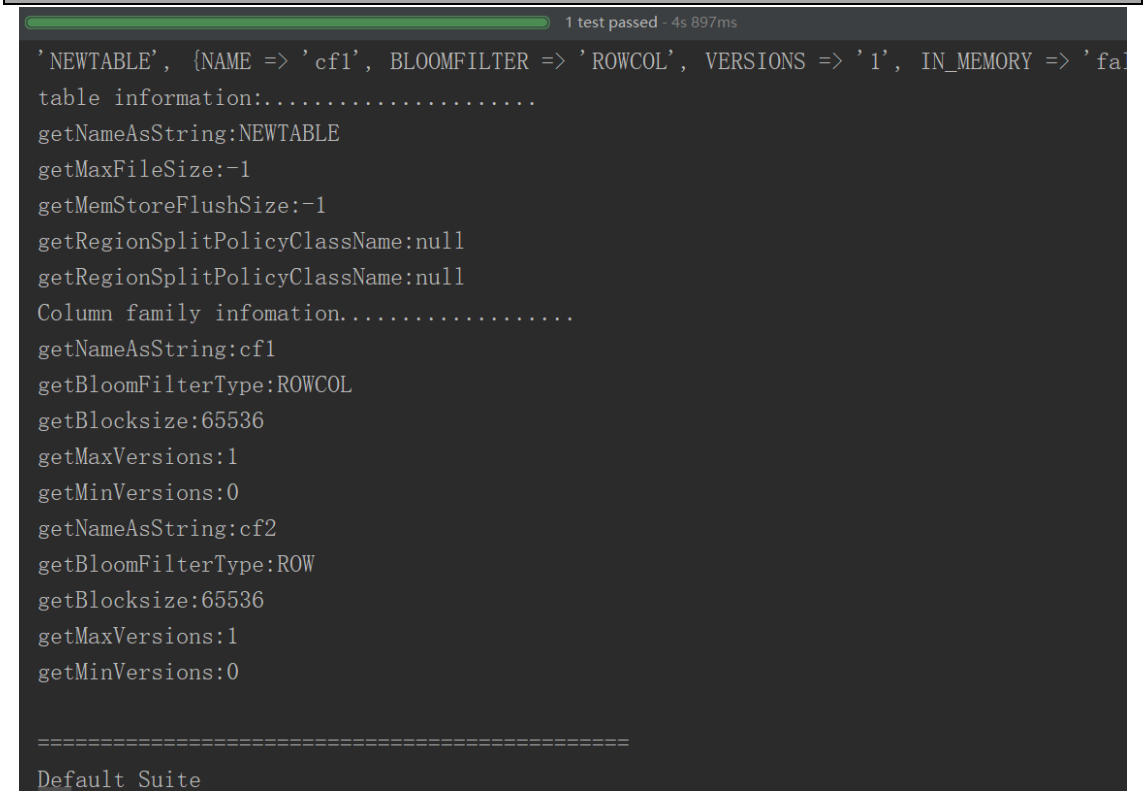
```
public void desctable() throws IOException {
        getconncet();
        TableName tableName =TableName.valueOf("player");
        Admin admin = connection.getAdmin();
        HTableDescriptor
descriptor=admin.getTableDescriptor(tableName);
        System.out.println(descriptor.toString());
```

```
    }
```

```
19/11/09 16:01:03 INFO zookeeper.ClientCnxn: Opening socket connection to server node/101.132.96.221:21
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();
        }
```

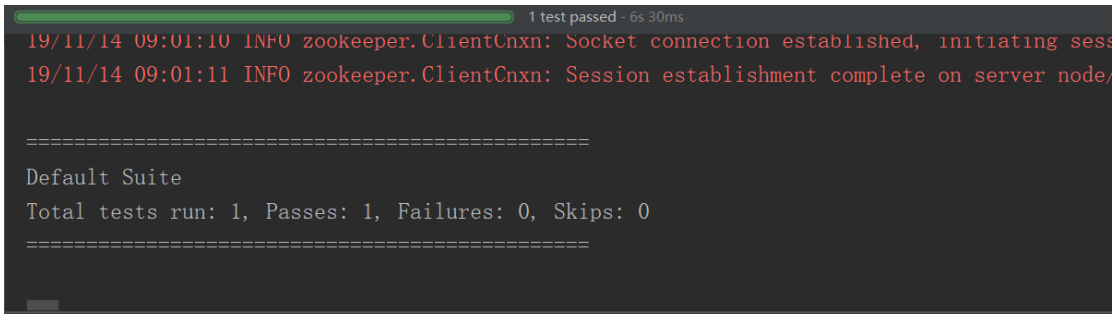


数据更新

```
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();
    }
```

```
1 test passed - 4s 827ms
19/11/14 09:10:44 INFO zookeeper.ClientCnxn: Session establishment complete on serve
row1/cf1:col0/1573693274029/Put/vlen=6/seqid=0:cf1:value0:1573693274029
row1/cf1:col1/1573693274029/Put/vlen=6/seqid=0:cf1:value1:1573693274029
row1/cf2:col2/1573693274029/Put/vlen=6/seqid=0:cf2:value2:1573693274029
row1/cf2:col3/1573693274029/Put/vlen=6/seqid=0:cf2:value3:1573693274029

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

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());
            }
        }
    }
```

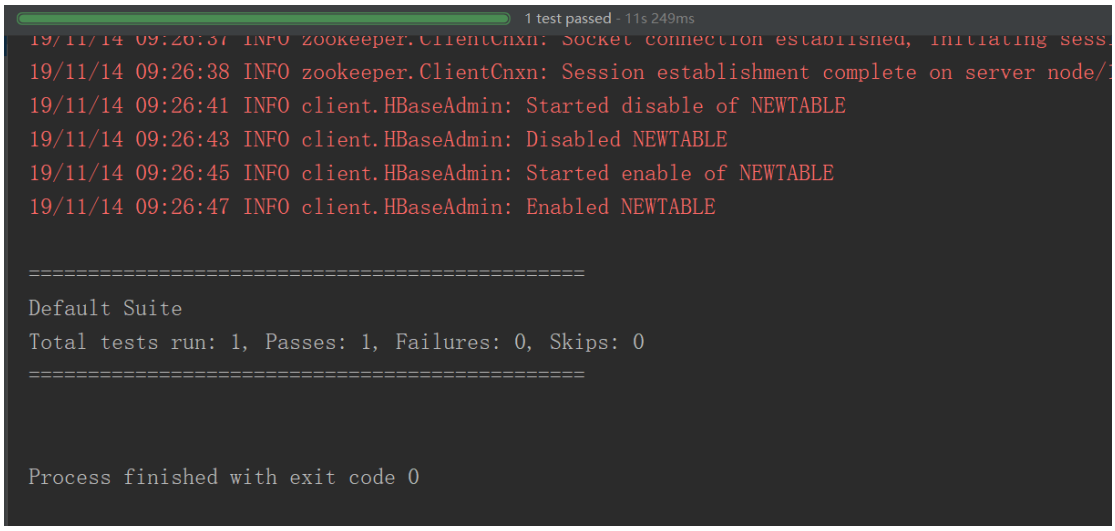
```
1 test passed - 5s 894ms
19/11/14 09:19:12 INFO zookeeper.ClientCnxn: Socket connection established, initiating s
19/11/14 09:19:13 INFO zookeeper.ClientCnxn: Session establishment complete on server no
row1/cf1:col0/1573693274029/Put/vlen=6/seqid=0:cf1:value0:1573693274029
row1/cf1:col1/1573693274029/Put/vlen=6/seqid=0:cf1:value1:1573693274029
row1/cf2:col2/1573693274029/Put/vlen=6/seqid=0:cf2:value2:1573693274029
row1/cf2:col3/1573693274029/Put/vlen=6/seqid=0:cf2:value3:1573693274029
row2/cf1:col0/1573693274379/Put/vlen=6/seqid=0:cf1:value4:1573693274379
row2/cf1:col4/1573693274379/Put/vlen=6/seqid=0:cf1:value5:1573693274379
row2/cf2:col3/1573693274379/Put/vlen=6/seqid=0:cf2:value6:1573693274379
row2/cf2:col5/1573693274379/Put/vlen=6/seqid=0:cf2:value7:1573693274379
row3/cf1:col0/1573693274740/Put/vlen=6/seqid=0:cf1:value4:1573693274740
row3/cf1:col6/1573693274740/Put/vlen=6/seqid=0:cf1:value8:1573693274740
row3/cf2:col3/1573693274740/Put/vlen=6/seqid=0:cf2:value9:1573693274740
row3/cf2:col7/1573693274740/Put/vlen=7/seqid=0:cf2:value10:1573693274740
row4/cf1:col0/1573693274740/Put/vlen=7/seqid=0:cf1:value11:1573693274740
row4/cf1:col8/1573693274740/Put/vlen=6/seqid=0:cf1:value8:1573693274740
row4/cf2:col3/1573693274740/Put/vlen=6/seqid=0:cf2:value9:1573693274740
row4/cf2:col9/1573693274740/Put/vlen=7/seqid=0:cf2:value12:1573693274740

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

删除行和列

删除列族和列

```
private void removecol()throws IOException{
```

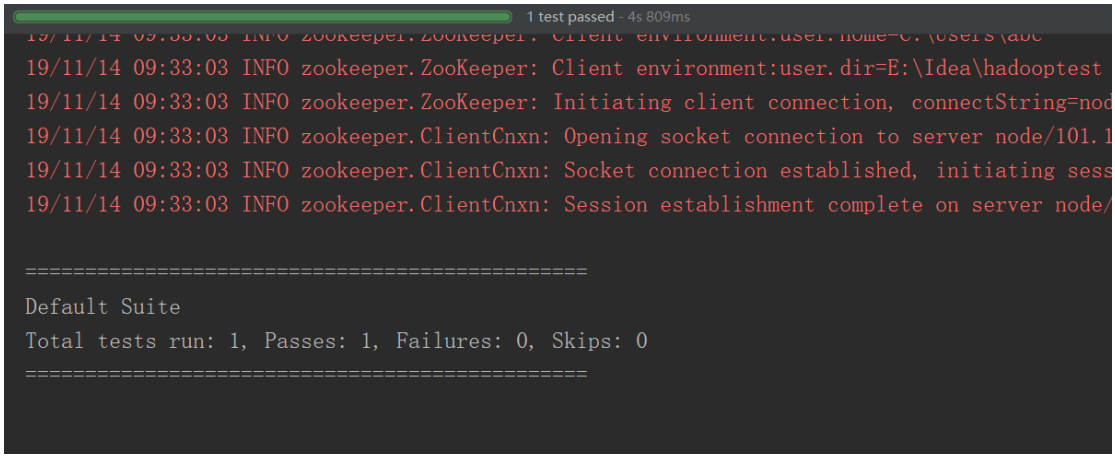
```
getconncet();
            Admin admin=connection.getAdmin();
            HTableDescriptor
descriptor=admin.getTableDescriptor(TableName.valueOf("NEWTABLE"));
            TableName tableName =TableName.valueOf("NEWTABLE");
            descriptor.removeFamily(Bytes.toBytes("col0"));
            admin.disableTable(tableName);
            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.locale

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

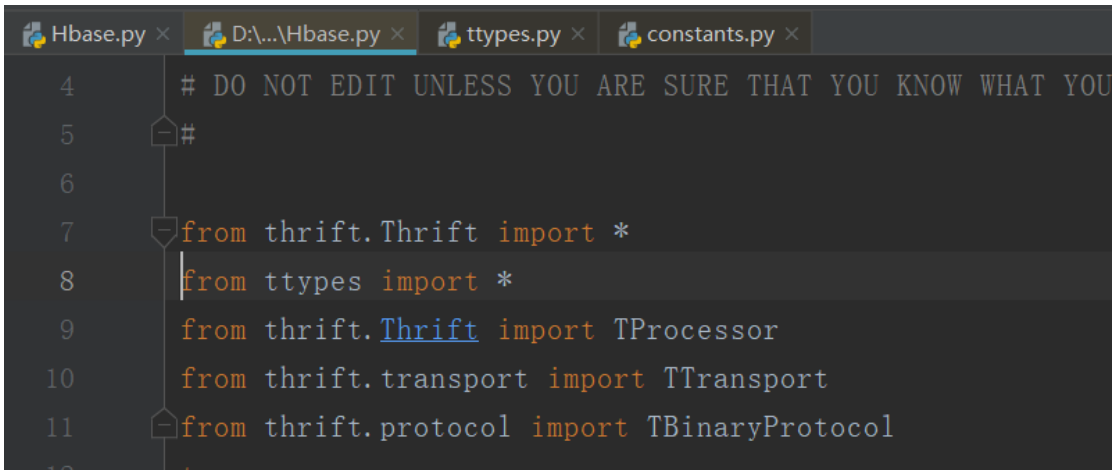
```
Traceback (most recent call last):
  File "E:/PycharmWorkspaces/NoSql/Hbase.py", line 3, in <module>
    from hbase import Hbase
  File "D:\ProgramData\Anaconda3\envs\nosql\lib\site-packages\hbase\Hbase.py", line 2066
    except IOError, io:
                  ^
SyntaxError: invalid syntax
```

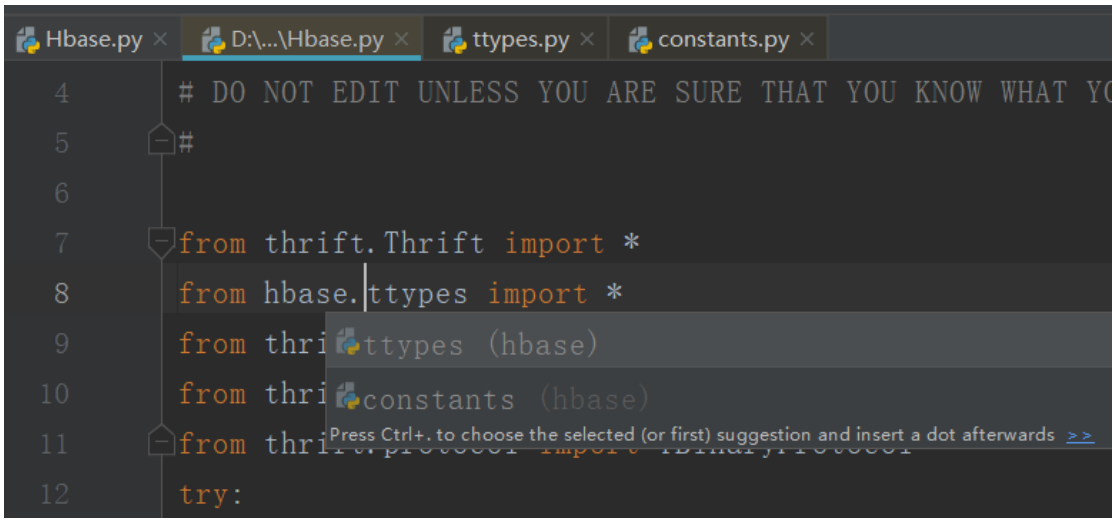
改为

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

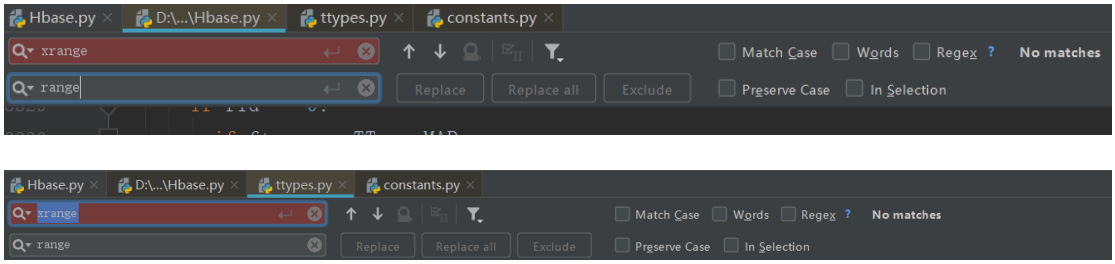
```
Traceback (most recent call last):
  File "E:/PycharmWorkspaces/NoSql/Hbase.py", line 3, in <module>
    from hbase import Hbase
  File "D:\ProgramData\Anaconda3\envs\nosql\lib\site-packages\hbase\Hbase.py", line 8, in <module>
    from ttypes import *
ModuleNotFoundError: No module named 'ttypes'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

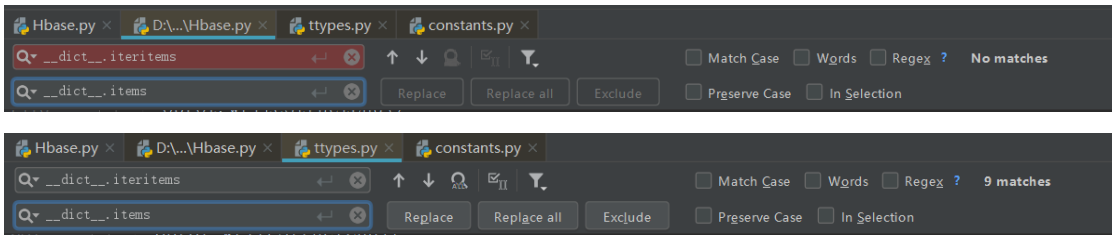
改为





替换处理



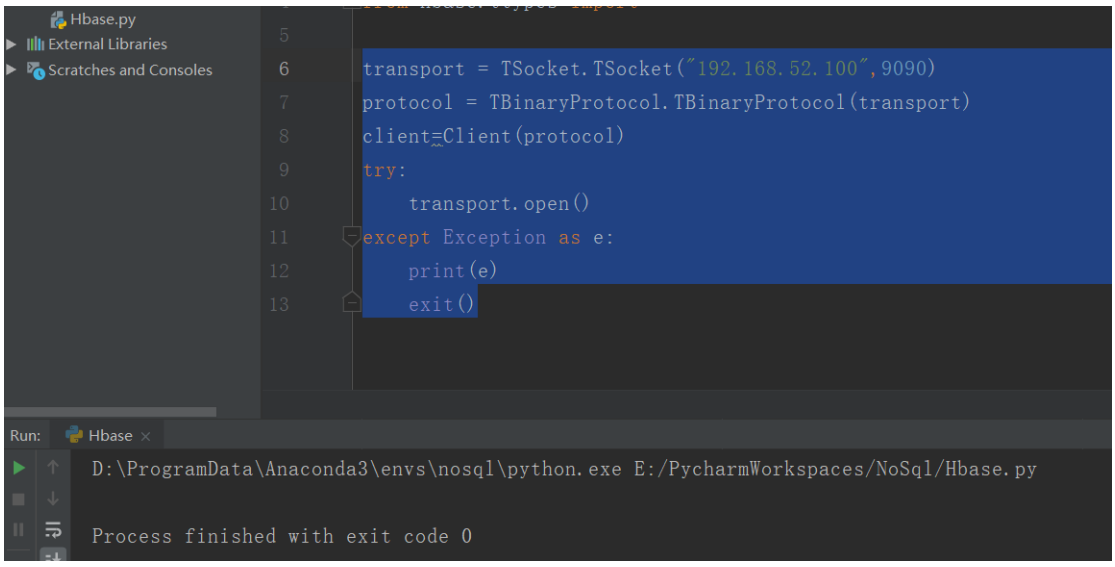


引用的类库

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

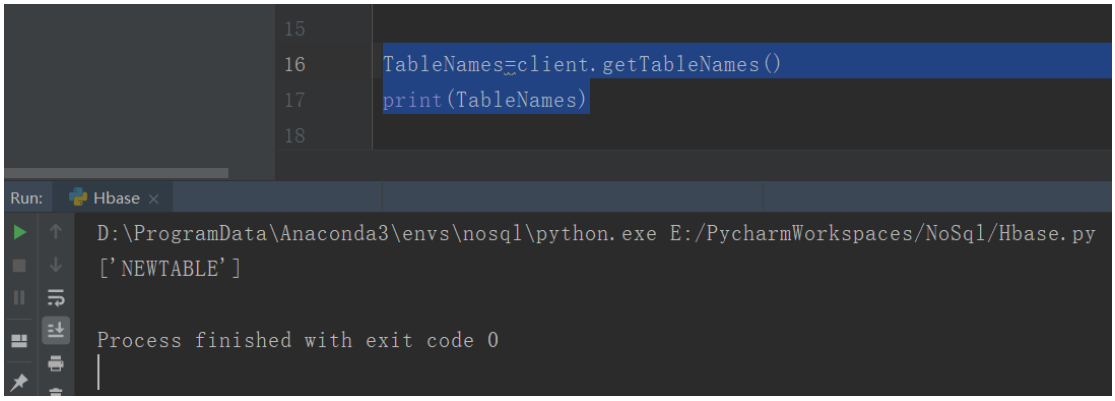
建立连接

```
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',valu
e='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)
```

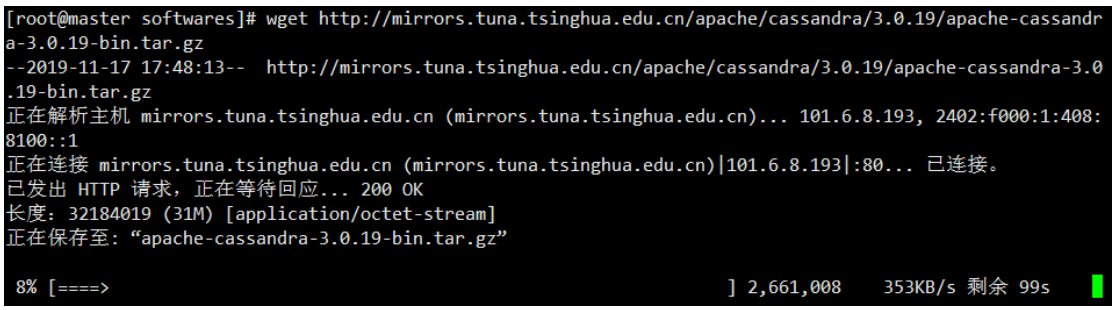
```
rowkey: row-key1
column qualifier: cf1:a
value: 1
timestamp: 1573911335534
column qualifier: cf1:c
value: 20
timestamp: 1573911335534
```

删除数据

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

安装 Cassandra

```
cd /export/softwares/
wget
http://mirrors.tuna.tsinghua.edu.cn/apache/cassandra/3.0.19/apache-cassandra-3.0.19-bin.tar.gz
```




```
tar zxvf apache-cassandra-3.0.19-bin.tar.gz -C ../servers
```

```
vi /etc/profile
export CASSANDRA_HOME=/export/servers/apache-cassandra-3.0.19
export PATH=:$CASSANDRA_HOME/bin:$PATH
source /etc/profile
```

修改配置文件 cassandra.yaml

```
#进入$CASSANDRA_HOME/conf 配置文件所在的目录
```

```
cd $CASSANDRA_HOME/conf
```

a.修改 cassandra 集群的名字(默认是 Test Cluster)

```
# The name of the cluster. This is mainly used to prevent machines in
# one logical cluster from joining another.
cluster_name: 'Test Cluster'
```

b.设置集群种子节点 IP，如果多个用逗号分隔

```
# seeds is actually a comma-delimited list of addresses.
# Ex: "<ip1>,<ip2>,<ip3>"
- seeds: "192.168.52.100,192.168.52.110,192.168.52.120"
```

c.设置监听地址(本机的 IP)，是为了其他节点能与节点进行通信(默认是 localhost)，每台机器填自己机器的 IP

```
# Setting listen_address to 0.0.0.0 is always wrong.
listen_address: 192.168.52.100
```

d.开启 thrift rpc 服务(默认是 false)

```
# Whether to start the thrift rpc server.
start_rpc: true
```

e.设置 rpc 的地址(默认是 localhost)

```
# For security reasons, you should not expose this port to the internet.
Firewall it if needed.
rpc_address: 192.168.52.100
```

f.设置数据文件所在路径(默认是 $CASSANDRA_HOME/data/data)

```
# If not set, the default directory is $CASSANDRA_HOME/data/data.
data_file_directories:
     - /data1/cassandradata/data
     - /data2/cassandradata/data
     - /data3/cassandradata/data
     - /data4/cassandradata/data
```

```
    - /data5/cassandradata/data
```

g. 设置 commitlog 文件所在路径(默认是 $CASSANDRA_HOME/data/commitlog)

```
# If not set, the default directory is $CASSANDRA_HOME/data/commitlog.
commitlog_directory: /data6/cassandradata/commitlog
```

问：为什么要设置 data_file_directories 和 commitlog_directory?

答：因为这两个文件很大，分散集群中磁盘 I/O 压力，前者是 cassandra 实际数据存放的目录，后者是数据写入 commitlog 的文件目录

分发安装包

```
scp -r apache-cassandra-3.0.19/ node02:$PWD
scp -r apache-cassandra-3.0.19/ node03:$PWD
```

修改 $CASSANDRA_HOME/conf/cassandra.yaml 中的 listen_address 和 rpc_address 将其设置成自己的 IP

启动 cassandra

```
cassandra -r
```

```
ERROR 13:12:11 Port already in use: 7199; nested exception is:
        java.net.BindException: Address already in use (Bind failed)
java.net.BindException: Address already in use (Bind failed)
        at java.net.PlainSocketImpl.socketBind(Native Method) ~[na:1.8.0_141]
        at java.net.AbstractPlainSocketImpl.bind(AbstractPlainSocketImpl.java:387) ~[na:1.8.0_141]
        at java.net.ServerSocket.bind(ServerSocket.java:375) ~[na:1.8.0_141]
```

```
netstat -tunlp |grep 7199
cassandra -p cassandra.pid
pkill -F cassandra.pid
```

```
nodetool status
```

```
[root@node01 apache-cassandra-3.0.19]# nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address         Load       Tokens       Owns (effective)  Host ID                               Rack
UN  192.168.52.120  106.97 KB  256          68.8%             968f2af4-79f6-4343-b93c-25123d7ba8a4  rack1
UN  192.168.52.110  86.84 KB   256          67.0%             da84a290-c346-442f-9afe-7855b26df331  rack1
UN  192.168.52.100  69.38 KB   256          64.2%             84a82b2f-757c-40e1-9dbb-16f2e7edb655  rack1

[root@node01 apache-cassandra-3.0.19]#
```

CentOS 6.9 下将 python2.6.6 升级为 Python2.7.13

查看当前系统中的 Python 版本

```
python --version
```

返回 Python 2.6.6 为正常。

检查 CentOS 版本

```
cat /etc/redhat-release
```

返回 CentOS release 6.9 (Final) 为正常。

安装所有的开发工具包

```
yum groupinstall -y "Development tools"
```

安装其它的必需包

```
yum install -y zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel
```

下载、编译和安装 Python 2.7.13

```
wget https://www.python.org/ftp/python/2.7.13/Python-2.7.13.tgz
tar zxf Python-2.7.13.tgz
cd Python-2.7.13
./configure
make && make install
```

默认 Python 2.7.13 会安装在 /usr/local/bin 目录下。

```
ll -tr /usr/local/bin/python*
```

```
/usr/local/bin/python2.7
/usr/local/bin/python2.7-config
/usr/local/bin/python -> python2
/usr/local/bin/python2 -> python2.7
/usr/local/bin/python2-config -> python2.7-config
/usr/local/bin/python-config -> python2-config
```

而系统自带的 Python 是在 /usr/bin 目录下。

```
ll -tr /usr/bin/python*
```

```
/usr/bin/python2.6-config
/usr/bin/python2.6
```

```
/usr/bin/python
/usr/bin/python2 -> python
/usr/bin/python-config -> python2.6-config
```

更新系统默认 Python 版本

先把系统默认的旧版 Python 重命名。

```
mv /usr/bin/python /usr/bin/python.old
```

再删除系统默认的 python-config 软链接。

```
rm -f /usr/bin/python-config
```

最后创建新版本的 Python 软链接。

```
ln -s /usr/local/bin/python /usr/bin/python
ln -s /usr/local/bin/python-config /usr/bin/python-config
ln -s /usr/local/include/python2.7/ /usr/include/python2.7
```

以上步骤做完以后，目录 /usr/bin 下的 Python 应该是

```
ll -tr /usr/bin/python*
```

```
/usr/bin/python2.6-config
/usr/bin/python2.6
/usr/bin/python.old
/usr/bin/python2 -> python
/usr/bin/python -> /usr/local/bin/python
/usr/bin/python-config -> /usr/local/bin/python-config
```

查看新的 Python 版本

```
python --version
```

返回 Python 2.7.13 为正常。

以下步骤还是有必要的

为新版 Python 安装 setuptools

```
wget https://bootstrap.pypa.io/ez_setup.py -O - | python
```

setuptools 正确安装完成后，easy_install 命令就会被安装在 /usr/local/bin 目录下了。

```
wget https://pypi.python.org/packages/source/d/distribute/distribute-0.6.10.tar.gz
tar xf distribute-0.6.10.tar.gz
cd distribute-0.6.10
python2.7 setup.py install
```

```
wget http://curl.haxx.se/ca/cacert.pem
mv cacert.pem ca-bundle.crt
cp ca-bundle.crt /etc/pki/tls/certs/
```

为新版 Python 安装 pip

```
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
```

至此，新版 Python 即算安装完毕了。

注意：这可能会导致以前安装过的 Python 程序运行不了或者无法重启之类的（比如著名的 Shadowsocks Python 版）。原因是旧版的 pkg_resources 位于 /usr/lib/python2.6/site-packages 下。而新版的则是在 /usr/local/lib/python2.7/site-packages 下。

所以，也许你需要重新安装一下程序。

再次注意：升级 Python 可能会导致 yum 命令不可用。解决方法如下：

编辑 /usr/bin/yum 文件，将开头第一行的

```
#!/usr/bin/python
```

改为

```
#!/usr/bin/python2.6
```

但是，这种改法，万一哪天你 yum update 了一下，yum 被升级了后，又变回老样子了。

记住旧版本 Python 2.6.6 的重要路径如下所示，在运行 yum 命令的时候，会提示你哪个 module 不存在，不存在的我们就去旧版本的路径下找，一定能找到的。找到后，复制到新版本 Python 的路径 /usr/local/lib/python2.7/site-packages/ 下即可。

/usr/lib/python2.6/site-packages/

/usr/lib64/python2.6/site-packages/

我的复制过程是这样的：

```
cp -r /usr/lib/python2.6/site-packages/yum /usr/local/lib/python2.7/site-packages/
cp -r /usr/lib/python2.6/site-packages/rpmUtils /usr/local/lib/python2.7/site-packages/
cp -r /usr/lib/python2.6/site-packages/iniparse /usr/local/lib/python2.7/site-packages/
```

```
cp -r /usr/lib/python2.6/site-packages/urlgrabber /usr/local/lib/python2.7/site-packages/
cp -r /usr/lib64/python2.6/site-packages/rpm /usr/local/lib/python2.7/site-packages/
cp -r /usr/lib64/python2.6/site-packages/curl /usr/local/lib/python2.7/site-packages/
cp -p /usr/lib64/python2.6/site-packages/pycurl.so /usr/local/lib/python2.7/site-packages/
cp -p /usr/lib64/python2.6/site-packages/_sqlitecache.so /usr/local/lib/python2.7/site-packages/
cp -p /usr/lib64/python2.6/site-packages/sqlitecachec.py /usr/local/lib/python2.7/site-packages/
cp -p /usr/lib64/python2.6/site-packages/sqlitecachec.pyc /usr/local/lib/python2.7/site-packages/
cp -p /usr/lib64/python2.6/site-packages/sqlitecachec.pyo /usr/local/lib/python2.7/site-packages/
```

cqlsh 基本用法

进入 shell

```
cqlsh
```

```
[root@master ~]# cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.0.19 | CQL spec 3.4.0 | Native protocol v4]
Use HELP for help.
cqlsh>
```

查看版本信息

```
show version
```

```
cqlsh> show version
[cqlsh 5.0.1 | Cassandra 3.0.19 | CQL spec 3.4.0 | Native protocol v4]
cqlsh>
```

描述集群信息

```
describe cluster
```

```
cqlsh> describe cluster

Cluster: Test Cluster
Partitioner: Murmur3Partitioner
```

查看空间列表

```
desc keyspaces
```

```
cqlsh> desc keyspaces

system_traces  system_schema  system_auth  system  system_distributed

cqlsh>
```

键空间管理

创建键空间

简单复制策略(SimpleStrategy)

```
create keyspace ks1 with replication={'class':'SimpleStrategy','replication_factor':'1'};
```

网络拓扑复制策略(NetworkTopologyStrategy)

```
create keyspace ks1 with replication={'class':'NetworkTopologyStrategy','dc1':3,'dc2':2 } AND DURABLE_WRITES=false;
```

删除键空间

```
drop keyspace ks1
```

查看键空间列表

```
describe keyspaces
```

```
cqlsh> describe keyspaces

ks1  system_schema  system_auth  system  system_distributed  system_traces

cqlsh>
```

修改键空间属性

```
alter keyspace ks1 with replication ={'class':'SimpleStrategy','replication_factor':'2'};
```

系统键空间

```
select * from system_schema.keyspaces;
```

```
cqlsh> select * from system_schema.keyspaces;

 keyspace_name      | durable_writes | replication
--------------------+----------------+-----------------------------------------------------------------------------------------
        system_auth |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
      system_schema |           True |                                {'class': 'org.apache.cassandra.locator.LocalStrategy'}
               ks1 |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}
 system_distributed |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
             system |           True |                                {'class': 'org.apache.cassandra.locator.LocalStrategy'}
      system_traces |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}

(6 rows)
cqlsh>
```

```
select * from system_schema.tables where keyspace_name='ks1';
```

```
cqlsh> select * from system_schema.tables where keyspace_name='ks1';

 keyspace_name | table_name | bloom_filter_fp_chance | caching | comment | compaction | compression | crc_check_chance | dclocal_read_repair_chance | default_time_to_live | extensions | flags | gc_grace_seconds |
 id | max_index_interval | memtable_flush_period_in_ms | min_index_interval | read_repair_chance | speculative_retry
---------------+------------+------------------------+---------+---------+------------+-------------+------------------+----------------------------+----------------------+------------+-------+------------------+
----+--------------------+-----------------------------+--------------------+--------------------+-------------------

(0 rows)
```

```
select * from system_schema.columns where keyspace_name='ks1' AND table_name='address';
```

```
cqlsh> select * from system_schema.columns where keyspace_name='ks1' AND table_name='address';

 keyspace_name | table_name | column_name | clustering_order | column_name_bytes | kind | position | type
---------------+------------+-------------+------------------+-------------------+------+----------+------
```

```
select * from system_schema.types;
```

```
cqlsh> select * from system_schema.types;

 keyspace_name | type_name | field_names | field_types
---------------+-----------+-------------+-------------
```

数据表管理

建立数据表

```
create table address(name text PRIMARY KEY,phone list<text>);
```

```
cqlsh> use ks1;
cqlsh:ks1> create table address(name text PRIMARY KEY,phone list<text>);
cqlsh:ks1>
```

```
describe ks1;
```

```
cqlsh:ks1> describe ks1;

CREATE KEYSPACE ks1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '2'}  AND durable_writes = true;

CREATE TABLE ks1.address (
    name text PRIMARY KEY,
    phone list<text>
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

cqlsh:ks1>
```

```
desc tables;
```

```
cqlsh:ks1> desc tables;

address
```

设置复合型主键

```
create table address2(name text,phone list<text> , primary key(name));
```

修改表结构

```
alter table address add age int;
alter table address with bloom_filter_fp_chance=0.01;
```

删除数据并重建表

```
truncate address;
```

用户自定义类型

```
create type scores(subject text,score int);
```

```
drop type scores;
```

CQL 数据查询

```
create table ks1.testtable1(
col1 text,
col2 int,
```

```
col3 tuple<text,text>,
PRIMARY KEY (col1,col2)
);
```

```
select * from ks1.testtable1;
```

```
cqlsh:ks1> select * from ks1.testtable1;

 col1 | col2 | col3
------+------+------

(0 rows)
```

条件查询

```
create table test(
key int,
col1 int,
col2 int,
col3 int,
col4 int,
primary key((key),col1,col2,col3,col4)
);
```

```
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,1,1,1);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,1,1,2);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,1,1,3);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,2,2,1);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,2,2,2);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,2,2,3);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,2,2,1);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,1,2,2);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,1,2,3);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,1,1,1);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,1,1,2);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,1,1,3);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,2,2,1);
insert into ks1.test(key,col1,col2,col3,col4) values(100,2,2,2,2);
insert into ks1.test(key,col1,col2,col3,col4) values(100,1,2,2,3);
```

```
cqlsh:ks1> select * from ks1.test;

 key | col1 | col2 | col3 | col4
-----+------+------+------+------
 100 |    1 |    1 |    1 |    1
 100 |    1 |    1 |    1 |    2
 100 |    1 |    1 |    1 |    3
 100 |    1 |    2 |    2 |    1
 100 |    1 |    2 |    2 |    2
 100 |    1 |    2 |    2 |    3
 100 |    2 |    1 |    1 |    1
 100 |    2 |    1 |    1 |    2
 100 |    2 |    1 |    1 |    3
 100 |    2 |    1 |    2 |    2
 100 |    2 |    1 |    2 |    3
 100 |    2 |    2 |    2 |    1
 100 |    2 |    2 |    2 |    2

(13 rows)
cqlsh:ks1>
```

```
select * from test where key =100 and col1 in (1,2) and col2=1 and col3=1 and col4<=2;
```

```
cqlsh:ks1> select * from test where key =100 and col1 in (1,2) and col2=1 and col3=1 and col4<=2;

 key | col1 | col2 | col3 | col4
-----+------+------+------+------
 100 |    1 |    1 |    1 |    1
 100 |    1 |    1 |    1 |    2
 100 |    2 |    1 |    1 |    1
 100 |    2 |    1 |    1 |    2

(4 rows)
cqlsh:ks1>
```

切片查询

```
select * from test where key=100 and col1=1 and col2=1 and (col3,col4)>=(1,2) and (col3,col4)<(2,3);
```

```
cqlsh:ks1> select * from test where key=100 and col1=1 and col2=1 and (col3,col4)>=(1,2) and (col3,col4)<
(2,3);

 key | col1 | col2 | col3 | col4
-----+------+------+------+------
 100 |    1 |    1 |    1 |    2
 100 |    1 |    1 |    1 |    3

(2 rows)
cqlsh:ks1>
```

索引机制

建立索引

```
create index indexofaddress on address(age) ;
```

删除索引

```
drop index indexofaddress;
```

使用标量函数

```
select writetime(col3) from ks1.testtable1;
```

```
cqlsh:ks1> select writetime(col3) from ks1.testtable1;

 writetime(col3)
------------------
 1574645253837295
```

```
select token(col3) from ks1.testtable1;
```

```
cqlsh:ks1> select token(col3) from ks1.testtable1;

 system.token(col3)
----------------------
 -4070488616938442618
```

数据更新

插入更新删除

数据插入

```
insert into ks1.testtable1(col1,col2,col3) values('some text',1,('the ket','the value'));
insert into ks1.testtable1(col1,col2,col3) values('other text',1,('another ket','another value'));
```

```
cqlsh:ks1> select * from testtable1 ;

 col1       | col2 | col3
------------+------+---------------------------------
 other text |    1 | ('another ket', 'another value')
          1 |    2 |                     ('3', None)
  some text |    1 |          ('the ket', 'the value')

(3 rows)
```

```
insert into ks1.testtable1(col1,col2,col3) values('some text',1,('a','b'));
insert into ks1.testtable1(col1,col2,col3) values('sther text',1,('c','d'));
```

```
cqlsh:ks1> select * from testtable1 ;

 col1       | col2 | col3
------------+------+---------------------------------
 other text |    1 | ('another ket', 'another value')
          1 |    2 |                     ('3', None)
  some text |    1 |          ('the ket', 'the value')

(3 rows)
cqlsh:ks1> insert into ks1.testtable1(col1,col2,col3) values('some text',1,('a','b'));
cqlsh:ks1> insert into ks1.testtable1(col1,col2,col3) values('sther text',1,('c','d'));
cqlsh:ks1> select * from testtable1 ;

 col1       | col2 | col3
------------+------+---------------------------------
 other text |    1 | ('another ket', 'another value')
 sther text |    1 |                     ('c', 'd')
          1 |    2 |                     ('3', None)
  some text |    1 |                     ('a', 'b')

(4 rows)
cqlsh:ks1>
```

数据更新

```
update ks1.testtable1 set col3 = ('anykey','new value') where col1='some text' and col2=1;
```

```
cqlsh:ks1> select * from testtable1 ;

 col1       | col2 | col3
------------+------+---------------------------------
 other text |    1 | ('another ket', 'another value')
 sther text |    1 |                     ('c', 'd')
          1 |    2 |                   ('3', None)
  some text |    1 |                     ('a', 'b')

(4 rows)
cqlsh:ks1> update ks1.testtable1 set col3 = ('anykey','new value') where col1='some text' and col2=1;
cqlsh:ks1> select * from testtable1 ;

 col1       | col2 | col3
------------+------+---------------------------------
 other text |    1 | ('another ket', 'another value')
 sther text |    1 |                     ('c', 'd')
          1 |    2 |                   ('3', None)
  some text |    1 |          ('anykey', 'new value')

(4 rows)
```

数据删除

```
DELETE col3 FROM ks1.testtable1 WHERE col1='some test'and col2=1;
DELETE from ks1.testtable1 where col1='other text' and col2=10;
```

json 格式插入数据

```
insert into ks1.testtable1 JSON '{"col1":"json test","col2":1000}';
```

读写一致性

查看一致性设置

```
CONSISTENCY ;
```

```
cqlsh:ks1> CONSISTENCY ;
Current consistency level is ONE.
cqlsh:ks1>
```

if 轻量级事物

```
insert into ks1.testtable1(col1,col2,col3) values('some test',1,('another key','another value')) if not exists;
```

集合列操作

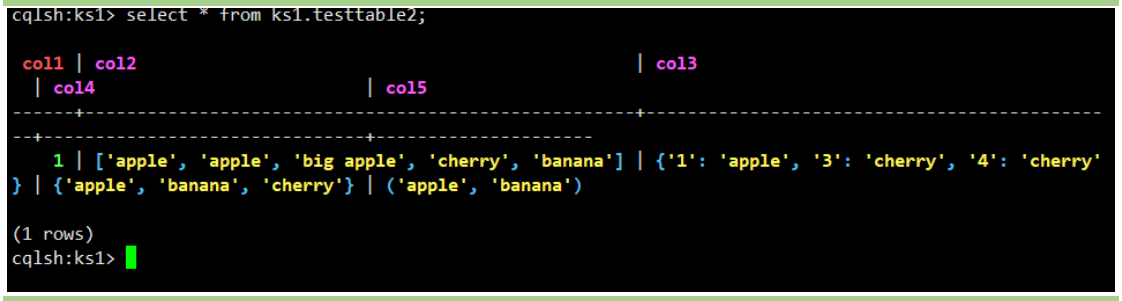
```
create table testtable2(
col1 int PRIMARY KEY,
```

```
col2 list<text>,
col3 map<text,text>,
col4 set<text>,
col5 frozen<tuple<text,text>>
);
```

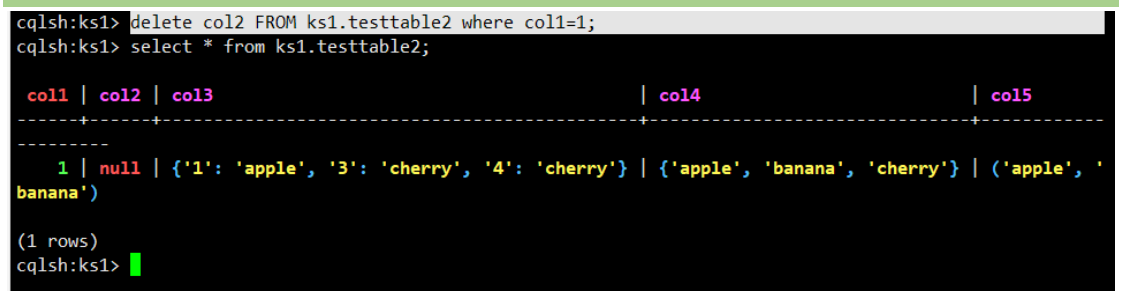
```
INSERT INTO ks1.testtable2(col1,col2,col3,col4,col5) VALUES (1,['apple','apple','banana','cherry','banana'],{'1':'apple','1':'banana','3':'cherry','4':'cherry'},{'apple','banana','cherry','apple'},('apple','banana'));
```

list 类型更新删除

```
update ks1.testtable2 set col2[2] = 'big apple' where col1 = 1 ;
```




```
delete col2[2] from ks1.testtable2 WHERE col1 =1;
delete col2 FROM ks1.testtable2 where col1=1;
```



set 类型更新和删除

```
UPDATE ks1.testtable2 set col4=col4+{'big apple','small apple'} where col1=1;
DELETE col4 FROM ks1.testtable2 WHERE col1=1;
```

```
cqlsh:ks1> select * from ks1.testtable2;

 col1 | col2 | col3                                                | col4 | col5
------+------+-----------------------------------------------------+------+---------------------
    1 | null | {'1': 'apple', '3': 'cherry', '4': 'cherry'} | null | ('apple', 'banana')

(1 rows)
cqlsh:ks1>
```

nodetool 工具

查看集群状态

```
nodetool version
```

```
[root@master ~]# nodetool version
ReleaseVersion: 3.0.19
```

```
nodetool ring
```

```
[root@master ~]# nodetool ring

Datacenter: datacenter1
==========
Address    Rack        Status State   Load            Owns                Token

                                                                          9196837031363529877
127.0.0.1  rack1       Up     Normal  177.83 KB       100.00%             -9155348372780184599
127.0.0.1  rack1       Up     Normal  177.83 KB       100.00%             -8997042927709325019
127.0.0.1  rack1       Up     Normal  177.83 KB       100.00%             -8875639460139531777
```

查看 compation 信息

```
nodetool compactionstats
```

JAVA 访问 Cassandra

修改 pom.xml

```
        <!--
https://mvnrepository.com/artifact/com.datastax.cassandra/cassandra-driver-core -->
        <dependency>
            <groupId>com.datastax.cassandra</groupId>
            <artifactId>cassandra-driver-core</artifactId>
```

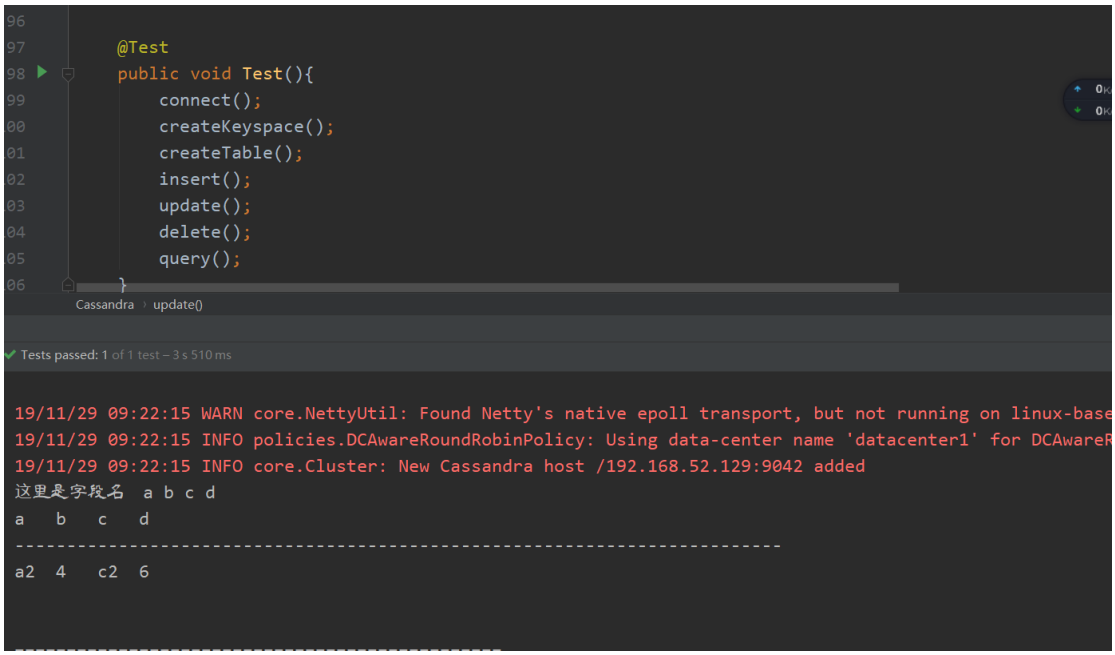
```
            <version>3.7.2</version>
        </dependency>
```

```
package cassandra;
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ColumnDefinitions.Definition;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Row;
import com.datastax.driver.core.Session;
import org.testng.annotations.Test;

public class Cassandra {
    public Cluster cluster;

    public Session session;

    public void connect(){
        Cluster culster=Cluster.builder().withClusterName("Test
Cluster").addContactPoint("192.168.52.129").build();
        session=culster.connect();
    }

    /**
     * 创建键空间
     */
    public void createKeyspace()
    {
        /**单数据中心 复制策略 : 1**/
        String cql = "CREATE KEYSPACE if not exists mydb WITH
replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}";
        session.execute(cql);
    }

    /**
     * 创建表
     */
    public void createTable()
```

```
    {
        /** a,b 为复合主键 a：分区键，b：集群键**/
        String cql = "CREATE TABLE if not exists mydb.test (a text,b
int,c text,d int,PRIMARY KEY (a, b))";
        session.execute(cql);
    }
    /**
     * 插入
     */
    public void insert()
    {
        String cql = "INSERT INTO mydb.test (a , b , c , d ) VALUES
( 'a2',4,'c2',6);";
        session.execute(cql);
    }

    /**
     * 修改
     */
    public void update()
    {
        // a,b 是复合主键 所以条件都要带上，少一个都会报错，而且
update 不能修改主键的值，这应该和 cassandra 的存储方式有关
        String cql = "UPDATE mydb.test SET d = 1234 WHERE a='aa'
and b=2;";
        // 也可以这样 cassandra 插入的数据如果主键已经存在，其实
就是更新操作
        String cql2 = "INSERT INTO mydb.test (a,b,d) VALUES
( 'aa',2,1234);";
        // cql 和 cql2 的执行效果其实是一样的
        session.execute(cql);
    }

    /**
     * 删除
     */
    public void delete()
```

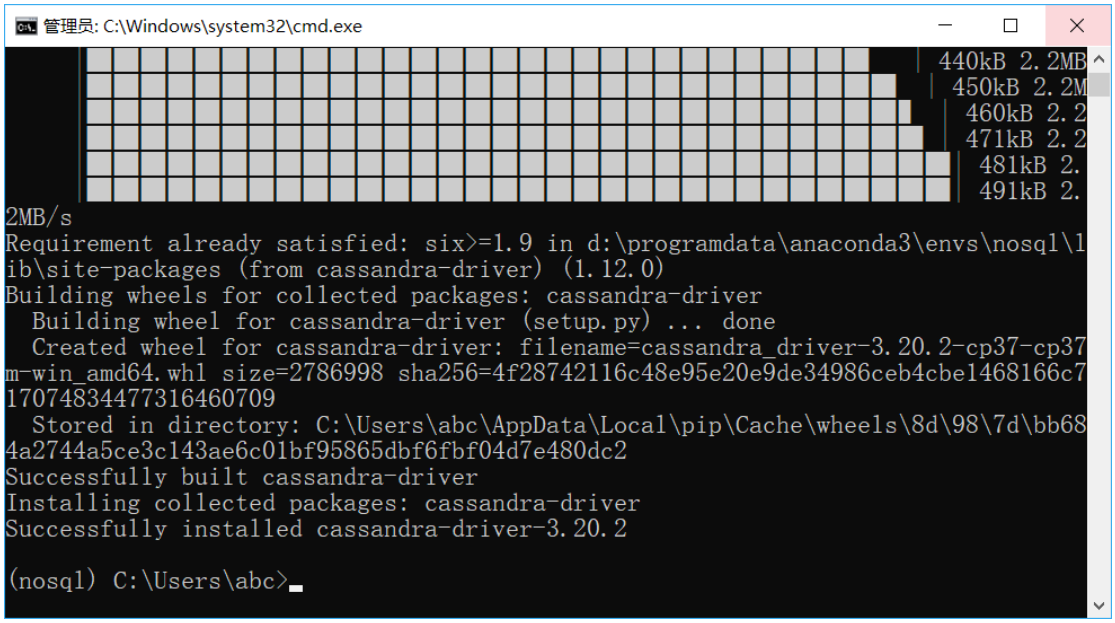
```
    {
        // 删除一条记录里的单个字段 只能删除非主键，且要带上主
键条件
        String cql = "DELETE d FROM mydb.test WHERE a='aa' AND
b=2;";
        // 删除一张表里的一条或多条记录 条件里必须带上分区键
        String cql2 = "DELETE FROM mydb.test WHERE a='aa';";
        session.execute(cql);
        session.execute(cql2);
    }

    /**
     * 查询
     */
    public void query()
    {
        String cql = "SELECT * FROM mydb.test;";
        String cql2 = "SELECT a,b,c,d FROM mydb.test;";

        ResultSet resultSet = session.execute(cql);
        System.out.print("这里是字段名： ");
        for (Definition definition : resultSet.getColumnDefinitions())
        {
            System.out.print(definition.getName() + " ");
        }
        System.out.println();
        System.out.println(String.format("%s\t%s\t%s\t%s\t\n%s",
"a", "b", "c", "d",
                "-----------------------------------------------
-------------------"));
        for (Row row : resultSet)
        {
            System.out.println(String.format("%s\t%d\t%s\t%d\t",
row.getString("a"), row.getInt("b"),
                    row.getString("c"), row.getInt("d")));
        }
    }
```

```
    @Test
    public void Test(){
        connect();
        createKeyspace();
        createTable();
        insert();
        update();
        delete();
        query();
    }
}
```



Python 访问 Cassandra

```
pip install cassandra-driver
```

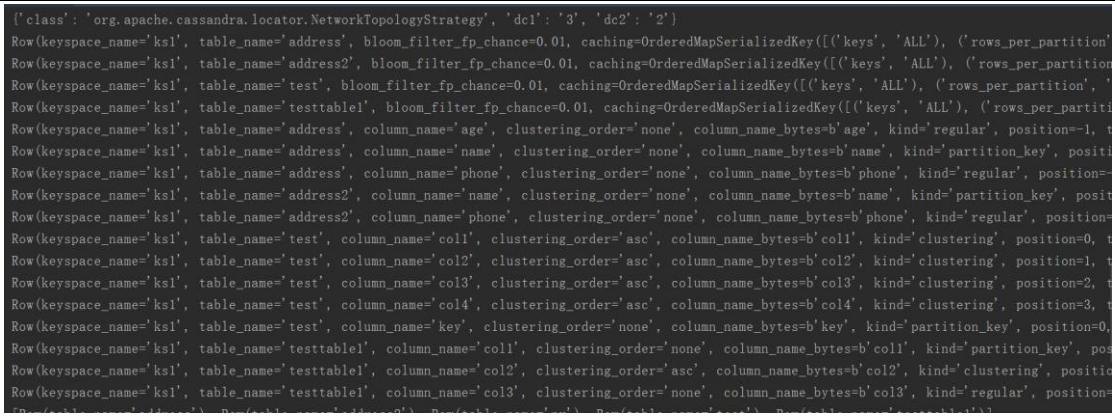



```
# encoding=UTF-8
from cassandra.cluster import Cluster
cluster = Cluster(['192.168.52.129'])
cluster.port=9042
session = cluster.connect()#创建连接
session_keyspace = cluster.connect("ks1") #直接连接 keyspace
'''
对 Cassandra 进行操作
'''
#1. 创建键空间
#1.1 使用 SimpleStrategy 策略创建键空间
'''
简单复制策略是指在一个数据中心的情况下使用简单的策略.该策略中,第一个副本被放置在所选择的节点上,剩下的节点采用 Dynamo 论文中的副本策略,不考虑机架位置
'''
#'replication':n 用于指示副本数量
#session.execute("create keyspace test with replication={'class':'SimpleStrategy','replication_factor':'1'};")

#1.2 网络拓补复制策略,在该策略下,数据副本采用二级机架感知策略
#Durable_writes 默认为 true,表示数据再写入时,先持久化在预写日志中,便于故障恢复.再网络拓补复制策略下,该选项可设置为 false,但有数据丢失
```

```
的风险
#session.execute("create keyspace ks3 with replication={'class':'NetworkTopologyStrategy','dc1':3,'dc2':2} and durable_writes=false;")

#1.4 查看键空间列表
#在 cqlsh 里可使用 describe spaces;查询键空间列表,但 execute 不支持 describe 命令
print(cluster.metadata.keyspaces)
print("\n")
# 因为 cassandra 为键值对的存储方式,所以,可使用类似显示字典内容的方式输出键空间
for k,v in cluster.metadata.keyspaces.items():
    print(k,v)
#1.5 修改键空间属性
session.execute("alter keyspace ks1 with replication={'class':'SimpleStrategy','replication_factor':2}")
#1.7 系统键空间
'''
    system_schema 表中存储当前所有的键空间和数据表的局部信息(schema 信息)
'''
result_keySpace=session.execute("select * from system_schema.keyspaces;")
for i in result_keySpace:
    print(i)
    for j in i:
        print(j)
'''
所有数据表的信息存储于系统表 system_shema.tables 中
'''
result_ks1_tables=session.execute("select * from system_schema.tables where keyspace_name = 'ks1';")
for i in result_ks1_tables:
    print(i)
'''
所有数据表的列信息存储于 system_schema.columns 中
```

```
'''
result_ks1_columns=session.execute("select * from system_schema.columns where keyspace_name = 'ks1';")
for i in result_ks1_columns:
    print(i)
'''
所有用户自定义数据类型都存储在 system_schema.types 中
'''
result_ks1_udf = session.execute("select * from system_schema.types")
for i in result_ks1_udf:
    print(i)



'''
创建表,在此之前需要保证是先使用目标键空间
'''
session.execute("use ks1;")
#2.1 建表操作
# session.execute("create table py(name text primary key,phone list<text>);")
print([i for i in session.execute("select table_name from system_schema.tables where keyspace_name='ks1';")])

# 几种特殊的数据类型：Map（键值对）、Set、list、Frozen（非具体类型，强调对被 frozen 限制的整体的操作，例如:frozen<tuple<text,text>>）

#2.2 删除表
session.execute("drop table address;")
#2.3 设置复合型主键
'''
可直接在单一主键后加 primary key，也可单独设置。
默认情况下，复合型主键的第一个主键为分区键(列)，其他为分簇键(列)
分簇列可单独设定升序(ASC)与降序(DESC)
'''
session.execute("create table address(firstname text,lastname text,No int,phone list<text>,primary key((firstname,lastname),No)) with clustering order by(No DESC);")
```

```
#2.4 修改表结构
'''
alter 仅支持 add drop rename 操作
'''
session.execute("alter table address add age int;")
print([i for i in session.execute("select column_name,type from system_schema.columns where keyspace_name='ks1' and table_name='address';")])
session.execute("alter table address drop age;")
print([i for i in session.execute("select column_name,type from system_schema.columns where keyspace_name='ks1' and table_name='address';")])


#间接查询，注意：若未设置索引，查询需添加 ALLOW FILTER 子句作为条件(性能无法预测)
print([i for i in session.execute("select * from system_schema.tables where table_name='stu' allow filtering;")])

cluster.shutdown()#关闭连接
```



MongoDB 安装

yum 安装

创建 yum 源文件：

```
cd /etc/yum.repos.d
```

```
vim mongodb-org-4.0.repo
```

添加以下内容：

```
[mngodb-org]
name=MongoDB Repository
baseurl=http://mirrors.aliyun.com/mongodb/yum/redhat/7Server/mongodb-org/4.0/x86_64/
gpgcheck=0
enabled=1
```

安装 MongoDB

安装命令：

```
yum -y install mongodb-org
```

安装完成后,查看 mongo 安装位置

```
whereis mongod
```

```
[root@master ~]# whereis mongod
mongod: /usr/bin/mongod /etc/mongod.conf /export/servers/mongodb-linux-x86_64-4.0.2/bin/mongod /usr/share/man/man1/mongod.1
```

```
systemctl start mongod.service
```

解压安装

MongoDB 官网 https://www.mongodb.com

```
cd /export/softwares/
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-4.0.2.tgz
```

```
tar zxvf mongodb-linux-x86_64-4.0.2.tgz -C ../servers/
```

```
vi /etc/profile
```

```
export MONGODB_HOME=/export/servers/mongodb-linux-x86_64-4.0.2
export PATH=:$MONGODB_HOME/bin:$PATH
```

```
cd /export/servers/mongodb-linux-x86_64-4.0.2
mkdir -p /export/servers/mongodb-linux-x86_64-4.0.2/db
```

开启 mongodb

```
mongod --dbpath /export/servers/mongodb-linux-x86_64-4.0.2/db
```

关闭 mongodb

```
mongod -shutdown
```

进入 shell

```
mongo --host 127.0.0.1:27017
```

```
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] ** WARNING: This server is bound to loc
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **          Remote systems will be unab
ct to this server.
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **          Start the server with --bir
ss> to specify which IP
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **          addresses it should serve r
om, or with --bind_ip_all to
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **          bind to all interfaces. If
or is desired, start the
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **          server with --bind_ip 127.0
able this warning.
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten]
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten]
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_
abled is 'always'.
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **        We suggest setting it to 'nev
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten]
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/mm/transparent_
frag is 'always'.
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten] **        We suggest setting it to 'nev
2019-11-29T16:39:32.544+0800 I CONTROL  [initandlisten]
>
```

配置文件

```
vi /etc/mongod.conf
```

```
bindIp: 0.0.0.0
```

shell

```
mongo
```

数据库和集合操作

数据库操作

查看当前连接服务器

```
db.getMongo()
```

```
> db.getMongo()
connection to 127.0.0.1:27017
```

查看数据库列表

```
show dbs
```

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

切换数据库

```
use test
```

```
> use test
switched to db test
>
```

查看数据库中所有集合

```
show collections
```

```
> show collections
mycol
>
```

json 显示集合名称

```
db.getCollectionNames()
```

```
> db.getCollectionNames()
[ "mycol" ]
```

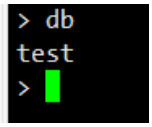
集合详细信息

```
db.getCollectionInfos()
```

```
> db.getCollectionInfos()
[
        {
                "name" : "mycol",
                "type" : "collection",
                "options" : {

                },
                "info" : {
                        "readOnly" : false,
                        "uuid" : UUID("8edc1f82-437c-449c-9a27-fb73e418dcb2")
                },
                "idIndex" : {
                        "v" : 2,
                        "key" : {
                                "_id" : 1
                        },
                        "name" : "_id_",
                        "ns" : "test.mycol"
                }
        }
]
>
```

显示当前数据库名

```
db
```



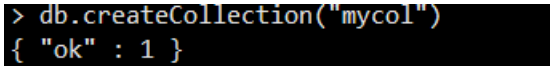
删除数据库

```
db.dropDatabase()
```

集合操作

新建集合

```
db.createCollection("mycol")
```



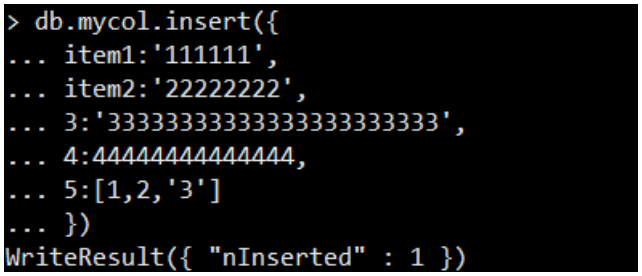
删除集合

```
db.myCol.drop()
```

基本增删改查操作

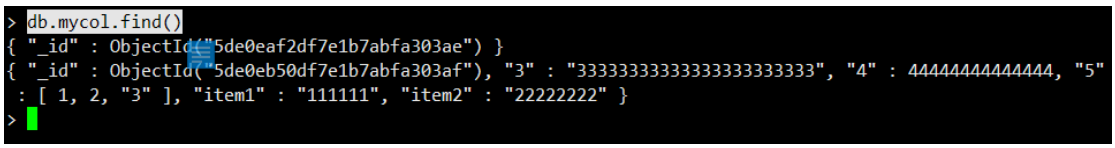
文档插入

```
db.mycol.insert({
item1:'111111',
item2:'22222222',
3:'3333333333333333333333',
4:44444444444444,
5:[1,2,'3']
})
```



查看文档

```
db.mycol.find()
```




```
db.mycol.find().pretty()
```

```
> db.mycol.find().pretty()
{ "_id" : ObjectId("5de0eaf2df7e1b7abfa303ae") }
{
        "_id" : ObjectId("5de0eb50df7e1b7abfa303af"),
        "3" : "333333333333333333333",
        "4" : 44444444444444,
        "5" : [
                1,
                2,
                "3"
        ],
        "item1" : "111111",
        "item2" : "22222222"
}
>
```

文档更新

```
db.mycol.update({'4':{$gt:0}},{$set:{'item2':'OK'}});
```

```
> db.mycol.update({'4':{$gt:0}},{$set:{'item2':'OK'}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.mycol.find().pretty()
{ "_id" : ObjectId("5de0eaf2df7e1b7abfa303ae") }
{
        "_id" : ObjectId("5de0eb50df7e1b7abfa303af"),
        "3" : "333333333333333333333",
        "4" : 44444444444444,
        "5" : [
                1,
                2,
                "3"
        ],
        "item1" : "111111",
        "item2" : "OK"
}
>
```

文档删除

```
db.mycol.remove({'4':{$gt:0}},true)
```

```
> db.mycol.remove({'4':{$gt:0}},true)
WriteResult({ "nRemoved" : 1 })
>
```

```
db.mycol.remove({})
```

聚合和管道

```
db.mycol.aggregate({$group:{_id:"item1",num:{$sum:1}}})
```

```
> db.mycol.aggregate({$group:{_id:"item1",num:{$sum:1}}})
{ "_id" : "item1", "num" : 1 }
>
```

```
db.mycol.aggregate({$project:{item1:1, item2:1, item3:1}})
```

```
> db.mycol.aggregate({$project:{item1:1, item2:1, item3:1}})
{ "_id" : ObjectId("5de0eaf2df7e1b7abfa303ae") }
```

```
db.mycol.aggregate({$match:{"item3":{$gt:1,$gt:15}}})

db.mycol.aggregate({$sort:{"item2":1}})
```

```
> db.mycol.aggregate({$sort:{"item2":1}})
{ "_id" : ObjectId("5de0f0f77190fd288caa07c6"), "item1" : "111111", "item2" : "22222222", "item3" : 1, "it
em4" : [ 1, 2, 3, 4, 5 ] }
>
```

```
db.mycol.aggregate({$unwind:'$item4'})
```

```
> db.mycol.aggregate({$unwind:'$item4'})
{ "_id" : ObjectId("5de0f0f77190fd288caa07c6"), "item1" : "111111", "item2" : "22222222", "item3" : 1, "it
em4" : 1 }
{ "_id" : ObjectId("5de0f0f77190fd288caa07c6"), "item1" : "111111", "item2" : "22222222", "item3" : 1, "it
em4" : 2 }
{ "_id" : ObjectId("5de0f0f77190fd288caa07c6"), "item1" : "111111", "item2" : "22222222", "item3" : 1, "it
em4" : 3 }
{ "_id" : ObjectId("5de0f0f77190fd288caa07c6"), "item1" : "111111", "item2" : "22222222", "item3" : 1, "it
em4" : 4 }
{ "_id" : ObjectId("5de0f0f77190fd288caa07c6"), "item1" : "111111", "item2" : "22222222", "item3" : 1, "it
em4" : 5 }
>
```

索引操作

```
db.mycol.createIndex({"item1":1})
```

```
> db.mycol.createIndex({"item1":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>
```

查看索引

```
db.mycol.getIndexes()
```

```
> db.mycol.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.mycol"
        },
        {
                "v" : 2,
                "key" : {
                        "item1" : 1
                },
                "name" : "item1_1",
                "ns" : "test.mycol"
        }
]
>
```

删除索引

```
db.mycol.dropIndex("myindex")
```

```
> db.mycol.dropIndex("myindex")
{ "nIndexesWas" : 3, "ok" : 1 }
>
```

```
db.mycol.dropIndexes()
```

全文索引

```
db.mycol.createIndex({"item1":"text"})
```

```
> db.mycol.createIndex({"item1":"text"})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
>
```

python 访问 MongoDB

安装

```
pip install pymongo
```

导包

```
from pymongo import MongoClient
```

建立连接

```
client=MongoClient("192.168.52.129:27017")
```

切换数据库

```
db=client.get_database("testdb")
db=client.testdb
```

切换集合

```
col=db.testcol
```

定义 JSON 文档

```
item={
      "name":"fruits",
      "count_of":3,
      "varieties":['banana',"cherry","orange"]
}
item1={
      "name":"fruits",
      "count_of":4,
      "varieties":['banana',"cherry","orange"]
}
item2={
      "name":"fruits",
      "count_of":5,
      "varieties":['banana',"cherry","orange"]
}
```

插入记录

```
col.insert_one(item)
```

```
col.insert_one(item1)
col.insert_one(item2)
```

查看数据

```
print(col.find_one())
```

```
{'_id': ObjectId('5de10e2e33b628133501c150'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}

Process finished with exit code 0
```

更新数据

```
col.update_many({"name":"fruits"},{"$push":{"varieties":"lemen"}})
print(col.find_one())
```

```
{'_id': ObjectId('5de10e5531eba9306ae55a95'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10e5531eba9306ae55a95'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen']}
```

$push $each 插入多个元素

```
col.update_many({"name":"fruits"},{"$push":{"varieties":{"$each":["1",'2','3']}}})
print(col.find_one())
```

```
{'_id': ObjectId('5de10e6fd7c603fdc75becca'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10e6fd7c603fdc75becca'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen']}
{'_id': ObjectId('5de10e6fd7c603fdc75becca'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}

Process finished with exit code 0
```

累加方式更新数字类型元素

```
col.update_many({"name":"fruits"},{"$inc":{"count_of":1}})
print(col.find_one())
```

```
{'_id': ObjectId('5de10e84b1a96b9e546c6de9'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10e84b1a96b9e546c6de9'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen']}
{'_id': ObjectId('5de10e84b1a96b9e546c6de9'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}
{'_id': ObjectId('5de10e84b1a96b9e546c6de9'), 'name': 'fruits', 'count_of': 4, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}

Process finished with exit code 0
```

$pop 删除数据

```
col.update_many({"name":"fruits"},{"$pop":{"varieties":-1}})
print(col.find_one())
```

```
{'_id': ObjectId('5de10e9834302bbec48df59f'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10e9834302bbec48df59f'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen']}
{'_id': ObjectId('5de10e9834302bbec48df59f'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}
{'_id': ObjectId('5de10e9834302bbec48df59f'), 'name': 'fruits', 'count_of': 4, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}
{'_id': ObjectId('5de10e9834302bbec48df59f'), 'name': 'fruits', 'count_of': 4, 'varieties': ['cherry', 'orange', 'lemen', '1', '2', '3']}

Process finished with exit code 0
```

更新数据

```
col.update_many({"name":"fruits"},{"$set":{"count_of":10}})
print(col.find_one())
```

```
{'_id': ObjectId('5de10eafbbb5826a9c3272d6'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10eafbbb5826a9c3272d6'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen']}
{'_id': ObjectId('5de10eafbbb5826a9c3272d6'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}
{'_id': ObjectId('5de10eafbbb5826a9c3272d6'), 'name': 'fruits', 'count_of': 4, 'varieties': ['banana', 'cherry', 'orange', 'lemen', '1', '2', '3']}
{'_id': ObjectId('5de10eafbbb5826a9c3272d6'), 'name': 'fruits', 'count_of': 4, 'varieties': ['cherry', 'orange', 'lemen', '1', '2', '3']}
{'_id': ObjectId('5de10eafbbb5826a9c3272d6'), 'name': 'fruits', 'count_of': 10, 'varieties': ['cherry', 'orange', 'lemen', '1', '2', '3']}

Process finished with exit code 0
```

查看数据

```
print(col.find_one({"name":"fruits"}))
```

排序 限制 跳过

```
for r in col.find({"name":"fruits"}).sort("count_of").limit(3).skip(2):
    print(r)
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/MongoDB.py
{'_id': ObjectId('5de10ef3e43c8711512164a9'), 'name': 'fruits', 'count_of': 5, 'varieties': ['banana', 'cherry', 'orange']}

Process finished with exit code 0
```

控制显示的列

```
for r in col.find({"name":"fruits"},projection={'count_of':False}):
    print(r)
```

```
{'_id': ObjectId('5de10f0afd44728751973e5e'), 'name': 'fruits', 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10f0afd44728751973e5f'), 'name': 'fruits', 'varieties': ['banana', 'cherry', 'orange']}
{'_id': ObjectId('5de10f0afd44728751973e60'), 'name': 'fruits', 'varieties': ['banana', 'cherry', 'orange']}

Process finished with exit code 0
```

聚合查询

```
print(col.find({"name":"fruits"}).count())
print(col.count_documents({"name":"fruits"}))
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/MongoDB.p
3

Process finished with exit code 0
```

比较运算符

```
for r in col.find({"count_of":{"$lt":4}}):
    print(r)
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/MongoDB.py
{'_id': ObjectId('5de10f3720eb6a69fff616f4'), 'name': 'fruits', 'count_of': 3, 'varieties': ['banana', 'cherry', 'orange']}

Process finished with exit code 0
```

地理索引查询

```
from pymongo import GEO2D
db.places.create_index([("loc",GEO2D)])
#插入经纬度信息
result = db.places.insert_many([{"loc": [2, 5]},{"loc":[30, 5]},{"loc":
[1, 2]},{"loc": [4, 4]}])
# $near  查询
for doc in db.places.find({"loc": {"$near": [3, 6]}}).limit(3):
    print(doc)
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/MongoDB.py
{'_id': ObjectId('5de10f579147d80cf7197b0d'), 'loc': [2, 5]}
{'_id': ObjectId('5de10f579147d80cf7197b10'), 'loc': [4, 4]}
{'_id': ObjectId('5de10f579147d80cf7197b0f'), 'loc': [1, 2]}

Process finished with exit code 0
```

Gridfs 操作

```
import gridfs
fs=gridfs.GridFS(db)
fileid=fs.put(b"hello word",filename="testfile")
print(fs.exists({'filename':'testfile'}))
print(fs.list())
for doc in fs.find({"filename":"testfile"}):
    print(doc._id)
    print(doc.filename)
    print(doc.read())
    #删除
    fs.delete(doc._id)
```

```
D:\ProgramData\Anaconda3\envs\nosql\python.exe E:/PycharmWorkspaces/NoSql/MongoDB.py
True
['testfile']
5de10f705e0eaa1382012d4e
testfile
b'hello word'

Process finished with exit code 0
```

删除数据

```
col.delete_one({"name":"fruits"})
print(col.find_one())
```

删除集合

```
db.places.delete_many({})
db.drop_collection("testcol")
```

Neo4j 安装

下载 Neo4j

1. 访问 neo4j 官网下载，官网地址：https://neo4j.com/

2. 访问微云数据主页：http://we-yun.com/index.php/blog/releases-56.html，提供国内 ftp 下载地址：ftp://neo4j.55555.io/neo4j/3.5.6/

安装

```
tar zxvf neo4j-community-3.5.6-unix.tar.gz -C ../servers/
```

```
cd /export/servers/neo4j-community-3.5.6/
vim conf/neo4j.conf
```

```
dbms.connectors.default_listen_address=0.0.0.0
dbms.connectors.default_advertised_address=0.0.0.0
```

```
54 dbms.connectors.default_listen_address=0.0.0.0
55
56 # You can also choose a specific network interface, and configure a r
57 # port for each connector, by setting their individual listen_address
58
59 # The address at which this server can be reached by its clients. Thi
60 # it may be the address of a reverse proxy which sits in front of the
61 # individual connectors below.
62 dbms.connectors.default_advertised_address=0.0.0.0
63
```

```
vi /etc/profile
```

```
export NEO4J_HOME=/export/servers/neo4j-community-3.5.6
export PATH=:$NEO4J_HOME/bin:$PATH
```

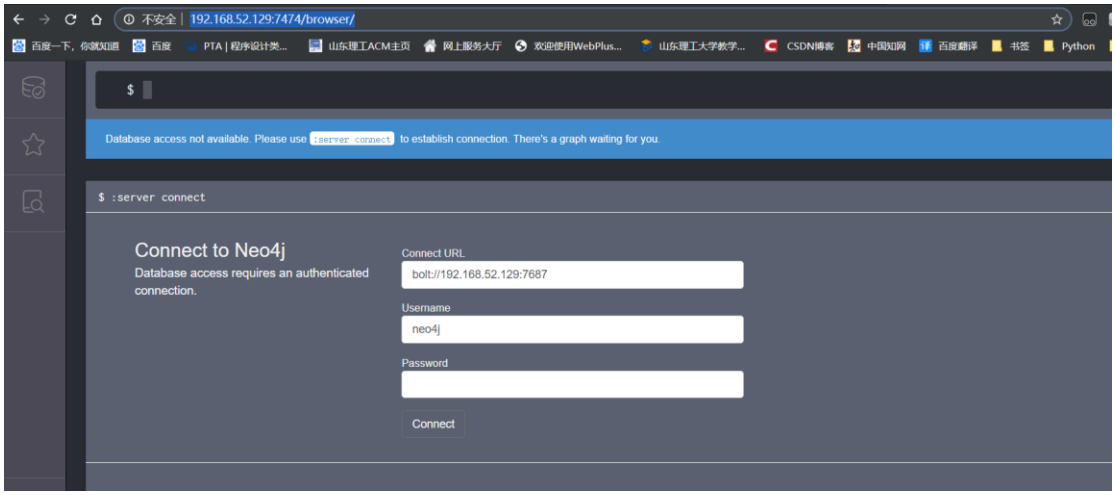
```
source /etc/profile
```

启动 neo4j 服务

```
neo4j start
```

访问 web

http://192.168.52.129:7474/browser/



登录界面默认填充了用户名 neo4j。

输入默认密码 neo4j 后，修改密码 123456，进入 neo4j

Python 访问 Neo4j

安装驱动

```
pip install neo4j-driver
```

导包

```
    from neo4j import GraphDatabase
```

建立连接

```
    driver=GraphDatabase.driver("bolt://192.168.52.129",auth=("neo4j",
"123456"))
    session = driver.session()
```

传递 Cypher 语句

```
    session.run("CREATE (a:Persion {name:'alice'})")
result=session.run("MATCH (a:Persion) RETURN a.name")
for i in result:
    print(i['a.name'])
```

```
result = session.run("MATCH (a:Persion) RETURN a")
for i in result:
    print(i['a'])
```

关闭连接

```
session.close()
```