实验二：HBase的安装与使用

成员：唐子钰、廖奕凯、刘帅、马天驰

实验目的

了解HBase的基础知识，包括HBase的结构、原理和应用场景。

掌握：

- 伪分布式(Pseudo Distributed)下的单机HBase安装与配置
- 完全分布式下Hbase的安装和配置
- HBASE shell的操作
- JAVA操作HBASE

Hbase Docker

Hbase介绍

HBase是一个开源的、分布式的、面向列的NoSQL数据库系统。它基于Google的Bigtable设计，运行在Hadoop分布式文件系统（HDFS）之上，提供了高可靠性、高性能、可伸缩性和易扩展性等特点。HBase主要用于存储大规模结构化数据，并支持实时读写操作。

以下是HBase的主要特性：

1. 面向列：HBase以列族为单位存储数据，每个列族可以包含多个列。这种面向列的存储方式使得HBase在处理大量数据时非常高效。
2. 分布式：HBase采用分布式架构，可以通过添加节点来扩展集群规模。同时，它还支持数据自动分片和负载均衡等功能。
3. 高可靠性：HBase采用多副本机制来保证数据的可靠性。每个Region都有多个副本，当某个副本出现故障时，系统会自动切换到其他副本。
4. 高性能：由于采用了面向列的存储方式和分布式架构，使得HBase具有非常高的读写性能。
5. 实时读写：HBase支持实时读写操作，并且可以通过添加Region Server来提高读写性能。
6. 灵活的数据模型：HBase支持灵活的数据模型，可以根据需要动态添加列族和列。

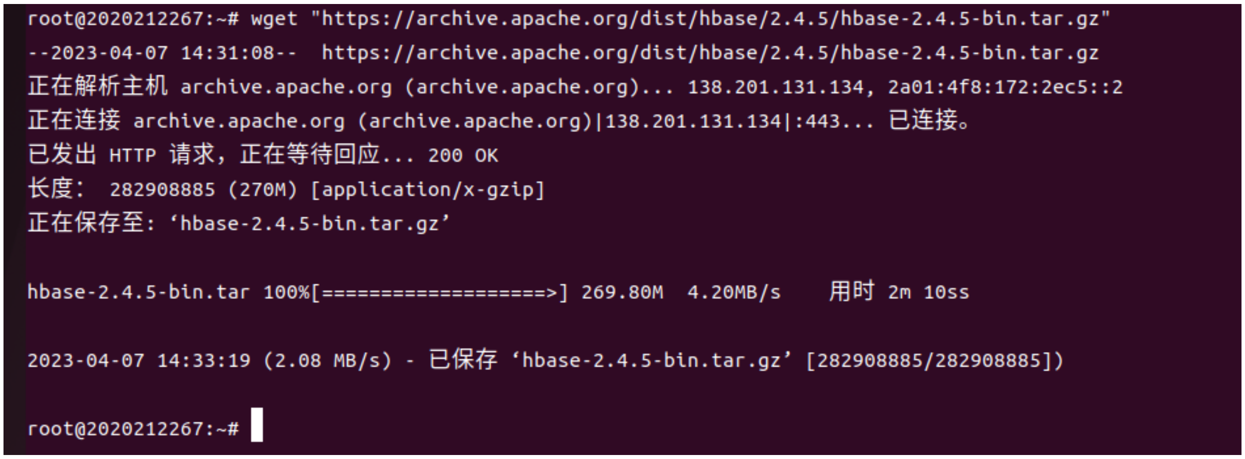
Hbase单机伪分布配置记录

环境

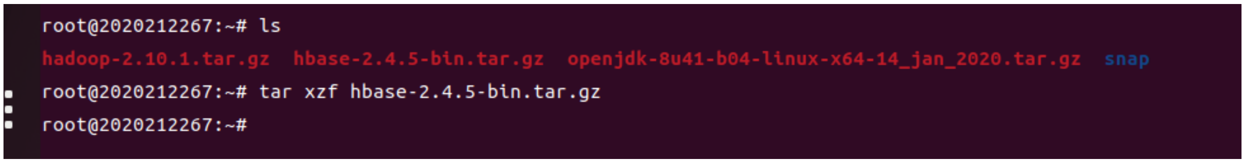
- Ubuntu 18.04
- JDK 1.8
- Hadoop 2.10.1
- HBase 2.4.5

1. 安装Hbase，下载地址：https://archive.apache.org/dist/hbase/2.4.5/hbase-2.4.5-bin.tar.gz

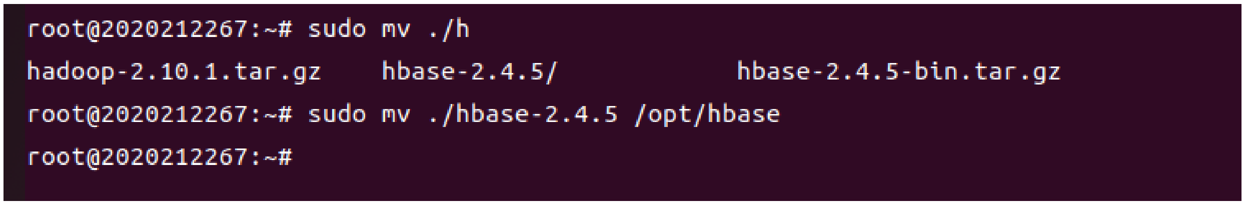
```
tar xzf hbase-2.4.5-bin.tar.gz
sudo mv ./hbase-2.4.5 /opt/hbase # 移动到opt目录下
```



```
root@2020212267:~# wget "https://archive.apache.org/dist/hbase/2.4.5/hbase-2.4.5-bin.tar.gz"
--2023-04-07 14:31:08--  https://archive.apache.org/dist/hbase/2.4.5/hbase-2.4.5-bin.tar.gz
正在解析主机 archive.apache.org (archive.apache.org)... 138.201.131.134, 2a01:4f8:172:2ec5::2
正在连接 archive.apache.org (archive.apache.org)|138.201.131.134|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度： 282908885 (270M) [application/x-gzip]
正在保存至: ‘hbase-2.4.5-bin.tar.gz’

hbase-2.4.5-bin.tar 100%[===================>] 269.80M  4.20MB/s    用时 2m 10ss

2023-04-07 14:33:19 (2.08 MB/s) - 已保存 ‘hbase-2.4.5-bin.tar.gz’ [282908885/282908885])

root@2020212267:~# 
```



```
root@2020212267:~# ls
hadoop-2.10.1.tar.gz  hbase-2.4.5-bin.tar.gz  openjdk-8u41-b04-linux-x64-14_jan_2020.tar.gz  snap
root@2020212267:~# tar xzf hbase-2.4.5-bin.tar.gz
root@2020212267:~#
```



```
root@2020212267:~# sudo mv ./h
hadoop-2.10.1.tar.gz     hbase-2.4.5/             hbase-2.4.5-bin.tar.gz
root@2020212267:~# sudo mv ./hbase-2.4.5 /opt/hbase
root@2020212267:~#
```

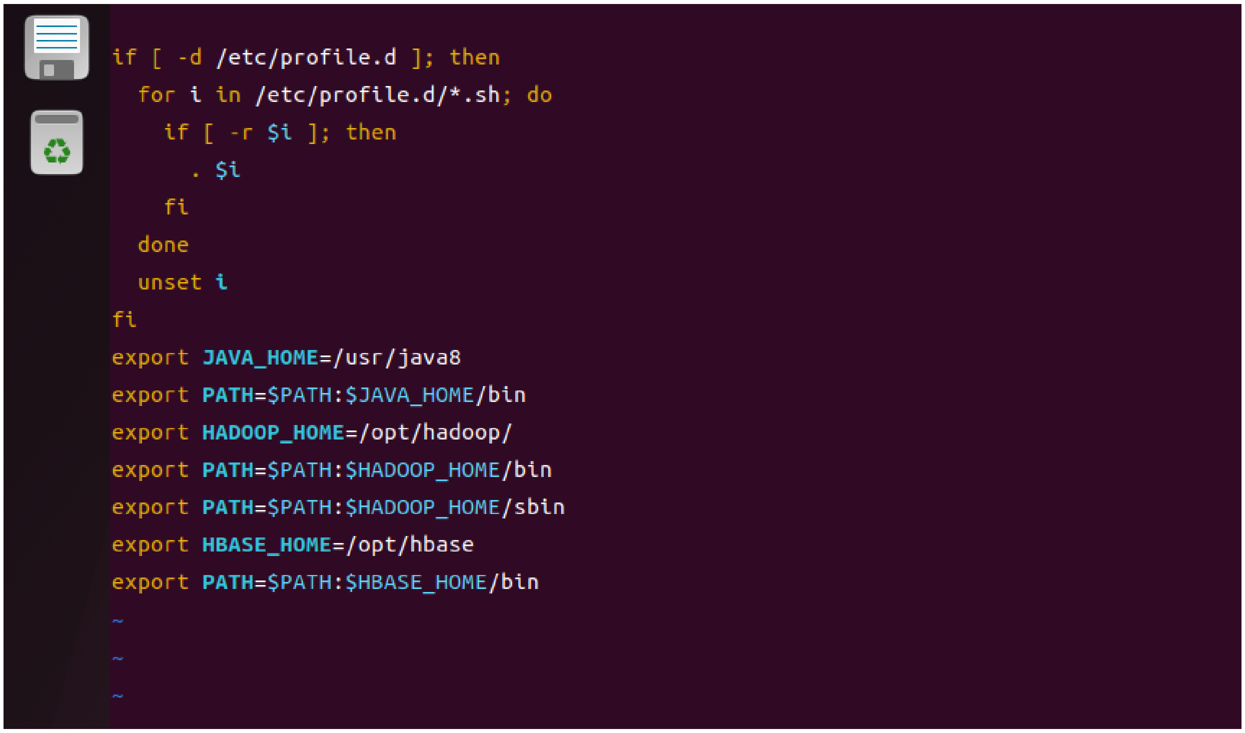
1.配置Hbase环境

```
sudo vim /etc/profile
# /etc/profile添加:
export HBASE_HOME=/opt/hbase
export PATH=$PATH:$HBASE_HOME/bin
source /etc/profile # 更新source
```

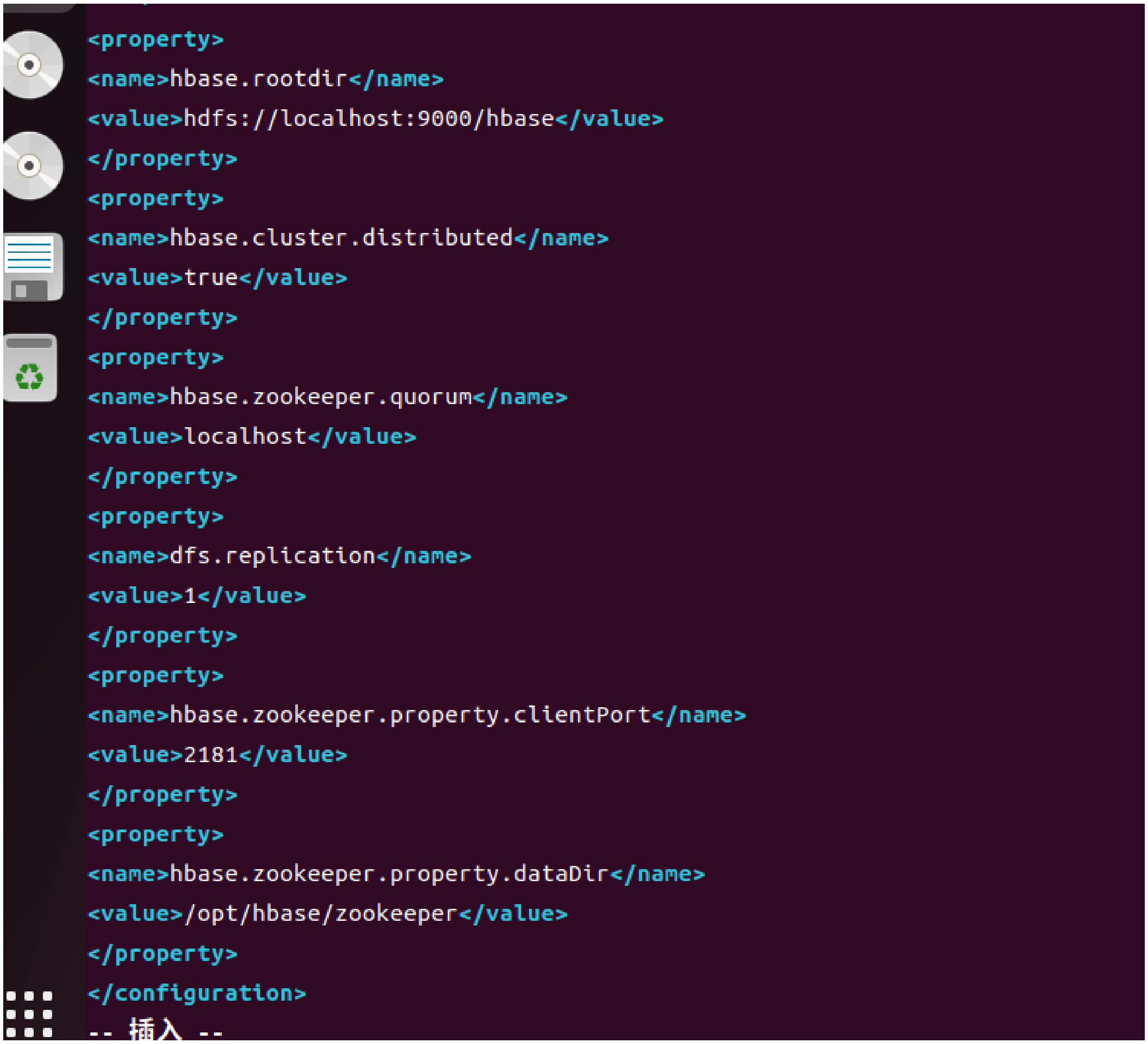
2.修改Hbase伪分布site配置文件

`hbase.rootdir` 属性的值应为 `hdfs://localhost:9000/hbase`；

伪分布式,所以`hbase.cluster.distributed`为 true

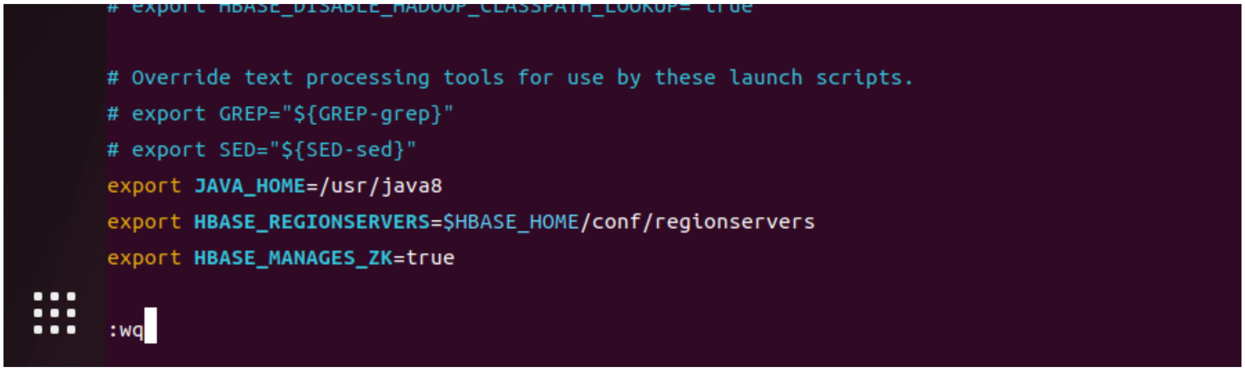


```
if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
export JAVA_HOME=/usr/java8
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/opt/hadoop/
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HBASE_HOME=/opt/hbase
export PATH=$PATH:$HBASE_HOME/bin
~
~
~
```



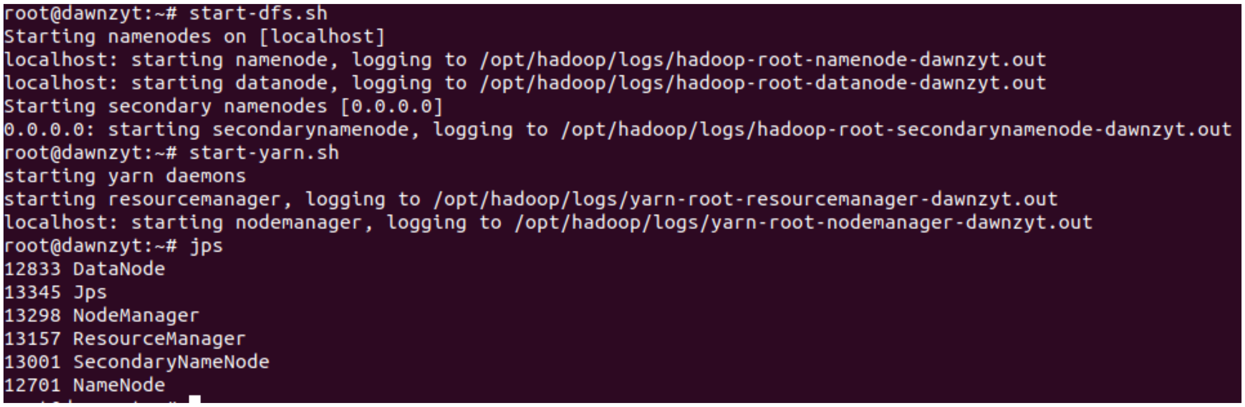
```
<property>
<name>hbase.rootdir</name>
<value>hdfs://localhost:9000/hbase</value>
</property>
<property>
<name>hbase.cluster.distributed</name>
<value>true</value>
</property>
<property>
<name>hbase.zookeeper.quorum</name>
<value>localhost</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>hbase.zookeeper.property.clientPort</name>
<value>2181</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/opt/hbase/zookeeper</value>
</property>
</configuration>
-- 插入 --
```

2.1 修改`hbase-env.sh`，配置区域服务器的路径,并启动自动管理zookeeper



```
# export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"

# Override text processing tools for use by these launch scripts.
# export GREP="${GREP-grep}"
# export SED="${SED-sed}"
export JAVA_HOME=/usr/java8
export HBASE_REGIONSERVERS=$HBASE_HOME/conf/regionservers
export HBASE_MANAGES_ZK=true

:wq
```

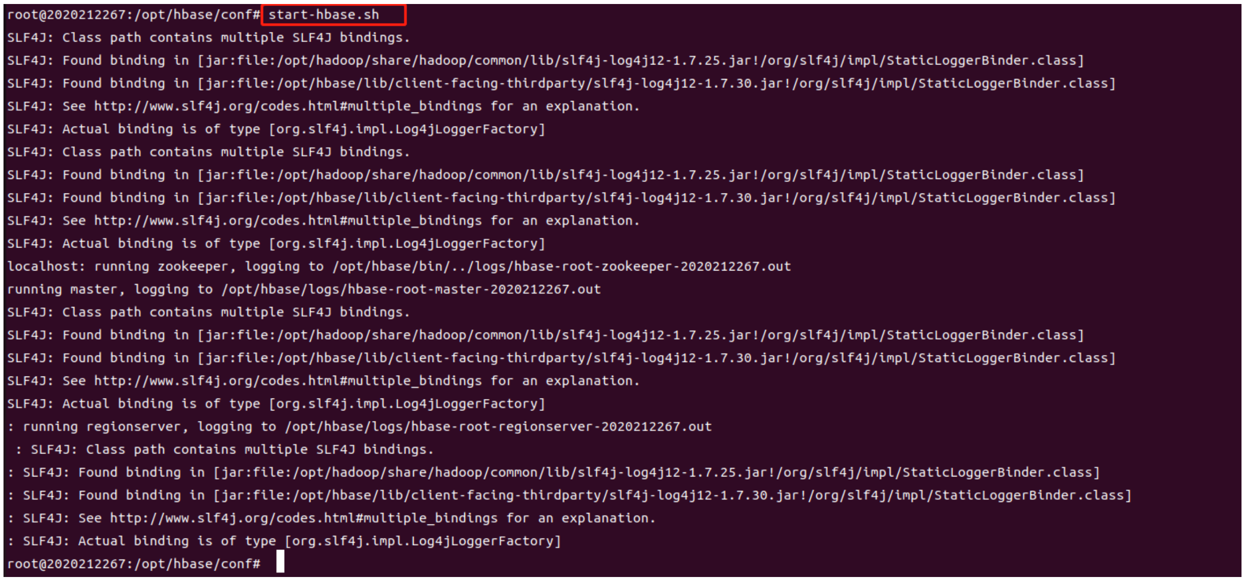
2.2 测试hbase，首先执行start-dfs.sh和`start-yarn.sh脚本运行hadoop，用于启动Hadoop分布式文件系统（HDFS）和YARN资源管理器



```
root@dawnzyt:~# start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /opt/hadoop/logs/hadoop-root-namenode-dawnzyt.out
localhost: starting datanode, logging to /opt/hadoop/logs/hadoop-root-datanode-dawnzyt.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-root-secondarynamenode-dawnzyt.out
root@dawnzyt:~# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /opt/hadoop/logs/yarn-root-resourcemanager-dawnzyt.out
localhost: starting nodemanager, logging to /opt/hadoop/logs/yarn-root-nodemanager-dawnzyt.out
root@dawnzyt:~# jps
12833 DataNode
13345 Jps
13298 NodeManager
13157 ResourceManager
13001 SecondaryNameNode
12701 NameNode
```

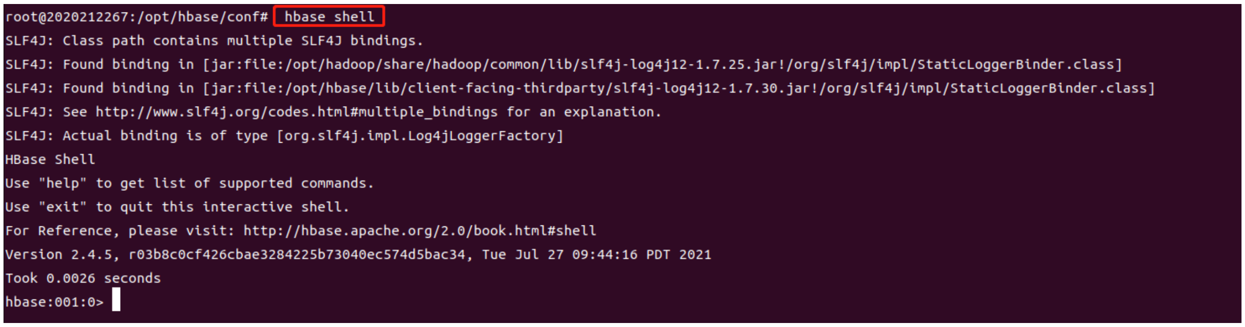
2.3 `start-hbase.sh`执行该脚本将启动HBase服务，并在终端中输出相应的日志信息。

查看`jps`

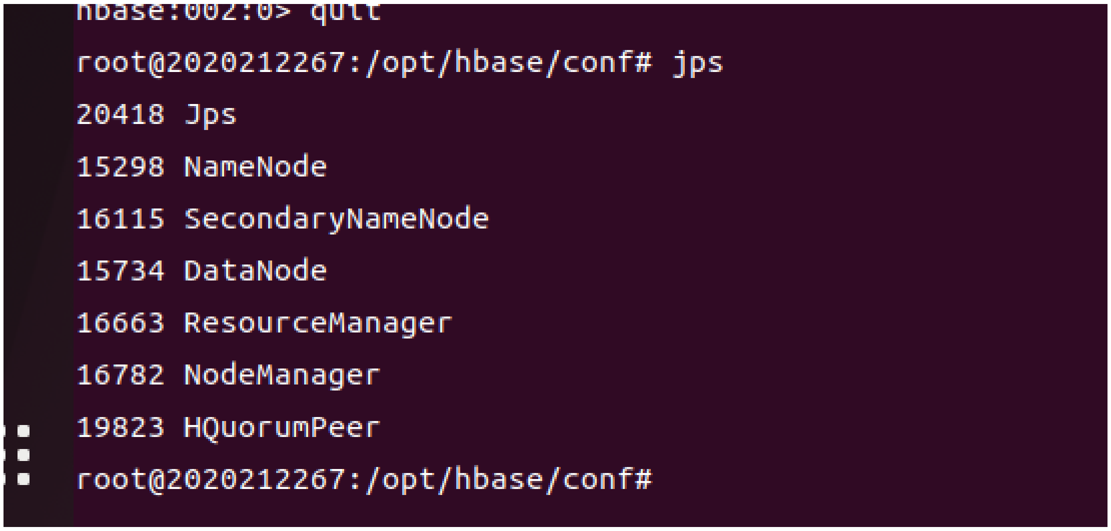


```
root@2020212267:/opt/hbase/conf# start-hbase.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
localhost: running zookeeper, logging to /opt/hbase/bin/../logs/hbase-root-zookeeper-2020212267.out
running master, logging to /opt/hbase/logs/hbase-root-master-2020212267.out
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
: running regionserver, logging to /opt/hbase/logs/hbase-root-regionserver-2020212267.out
 : SLF4J: Class path contains multiple SLF4J bindings.
: SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
: SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
root@2020212267:/opt/hbase/conf#
```

启动`shell`，从而执行hbase的各种操作，例如创建表、插入数据、查询数据等。



```
root@2020212267:/opt/hbase/conf# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.5, r03b8c0cf426cbae3284225b73040ec574d5bac34, Tue Jul 27 09:44:16 PDT 2021
Took 0.0026 seconds
hbase:001:0>
```



```
hbase:002:0> quit
root@2020212267:/opt/hbase/conf# jps
20418 Jps
15298 NameNode
16115 SecondaryNameNode
15734 DataNode
16663 ResourceManager
16782 NodeManager
19823 HQuorumPeer
root@2020212267:/opt/hbase/conf#
```

Hbase Docker 完全分布模拟

完全分布部分，由于时间问题，我们并没有在机房完成，而是在课下使用docker-compose模拟一个集群进行搭建。其核心部分包括单个节点的`dockerfile`配置和总体的`docker-compose.yml`配置

- **Dockerfile**
  - 主要进行，将Hadoop等文件复制进容器，设置ssh免密，暴露容器的端口
  - Docker File中，我将对配置文件的修改写为了config.sh

```
FROM ubuntu
COPY ./hosts /etc/hosts
COPY ./profile /etc/profile
COPY ./config.sh /app/config.sh
COPY ./ssh/* /root/.ssh/
COPY ./hadoop/ /opt/hadoop/
COPY ./java8/ /usr/java8/
COPY ./hbase/ /opt/hbase/

RUN chmod 600 /root/.ssh/id_rsa \
    && chmod +x /app/config.sh \
    && bash /app/config.sh \
    && echo 'root:123456' | chpasswd \
    && sed -i s@/archive.ubuntu.com/@/mirrors.aliyun.com/@g /etc/apt/sources.list \
    && apt-get clean \
    && apt-get update

RUN apt install openssh-server net-tools vim netcat sudo -y \
    && mkdir /var/run/sshd \
    && sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' \
      /etc/ssh/sshd_config \
    && sed -ri 's/^#?PubkeyAuthentication yes/PubkeyAuthentication yes/' \
      /etc/ssh/sshd_config \
    && sed -ri 's/^#?PasswordAuthentication yes/PasswordAuthentication no/' \
      /etc/ssh/sshd_config \
    && sed -ri 's/UsePAM yes/UsePAM no/g' /etc/ssh/sshd_config \
    && service ssh start
EXPOSE 22 8088 9000 19888 8042 16010 16000 \
  16030 16201 50010 50075 50475 50020 50070 \
  50470 8020 8485 8480 8019 8032 8030 8031 \
  8033 8088 8040 8042 8041 10020 19888 60000 \
  60010 60020 60030 2181 2888 3888 9083 10000 2181 2888 3888
ENTRYPOINT ["/bin/bash"]
# config.sh
echo 'export JAVA_HOME=/usr/java8' >> /etc/profile
echo 'export PATH=$PATH:$JAVA_HOME/bin' >> /etc/profile
echo 'export HADOOP_HOME=/opt/hadoop/' >> /etc/profile
echo 'export PATH=$PATH:$HADOOP_HOME/bin' >> /etc/profile
echo 'export PATH=$PATH:$HADOOP_HOME/sbin' >> /etc/profile
echo 'export HBASE_HOME=/opt/hbase' >> /etc/profile
echo 'export PATH=$PATH:$HBASE_HOME/bin' >> /etc/profile
echo 'source /etc/profile' >> /root/.bashrc
echo 'service ssh start' >> /root/.bashrc
echo "echo '172.16.238.10 Master' >> /etc/hosts" >> /root/.bashrc
echo "echo '172.16.238.11 Slave1' >> /etc/hosts" >> /root/.bashrc
echo "echo '172.16.238.12 Slave2' >> /etc/hosts" >> /root/.bashrc
source /etc/profile
```

- **Docker Compose**
  - 为各节点分配hostname
  - 为个节点分配固定的虚拟局域网地址
  - 开启tty和stdin_open, 保证后台运行

```
version: '3'
services:
  master:
    image: big_data:v1
    hostname: Master
    networks:
      app_net:
        ipv4_address: 172.16.238.10
    ports:
      - "50070:50070"
    stdin_open: true
    tty: true

  slave1:
    image: big_data:v1
    hostname: Slave1
    networks:
      app_net:
        ipv4_address: 172.16.238.11
    ports:
      - "50071:50070"
    stdin_open: true
    tty: true
  slave2:
    image: big_data:v1
    hostname: Slave2
    networks:
      app_net:
        ipv4_address: 172.16.238.12
    ports:
      - "50072:50070"
    stdin_open: true
    tty: true
networks:
  app_net:
    ipam:
      config:
        - subnet: 172.16.238.0/24
```

- **regionservesr**

```
Master
Slave1
Slave2
```

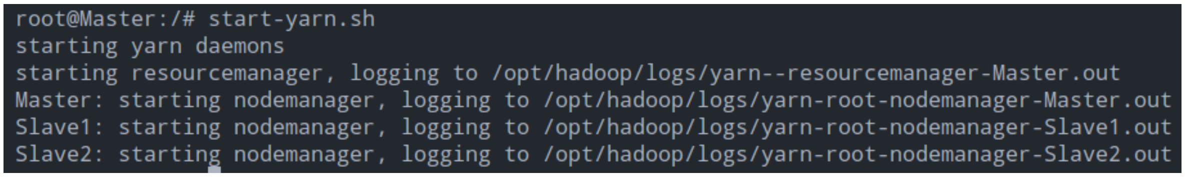
- **hbasesite**

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>hbase.tmp.dir</name>
    <value>./tmp</value>
  </property>
  <property>
    <name>hbase.unsafe.stream.capability.enforce</name>
    <value>false</value>
  </property>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://master:9000/hbase</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>Master,Slave1,Slave2</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
    <property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2181</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/opt/hbase/zookeeper</value>
  </property>
</configuration>
```

值得注意的是，在给出的指导文档中，并没有提及需要修改regionservers文件，以及hbase.zookeeper.quorum。但是，这两个配置对于配置完全分布式的Hbase至关重要。 其中，regionservers决定了HRegionServer能否在每个节点启动，而quorum决定了HQuorumPeer能否在每个节点启动。如果使用默认值localhost,则只能支持单机伪分布，只能在Master上启动，这两者都需要根据hostname进行设置！

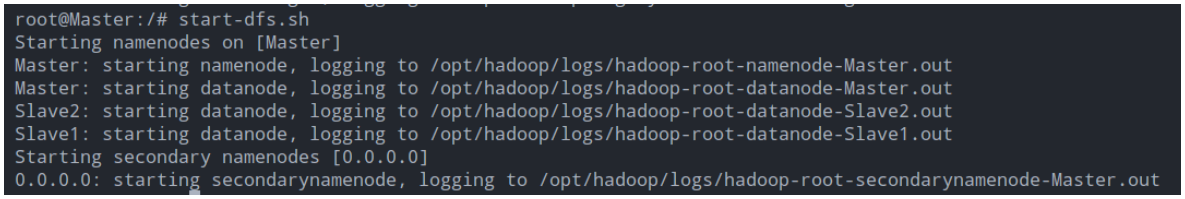
其具体启动过程与单机伪分布类似，依次执行

- hadoop namenode -format (如果没有执行过)
- start-yarn.sh



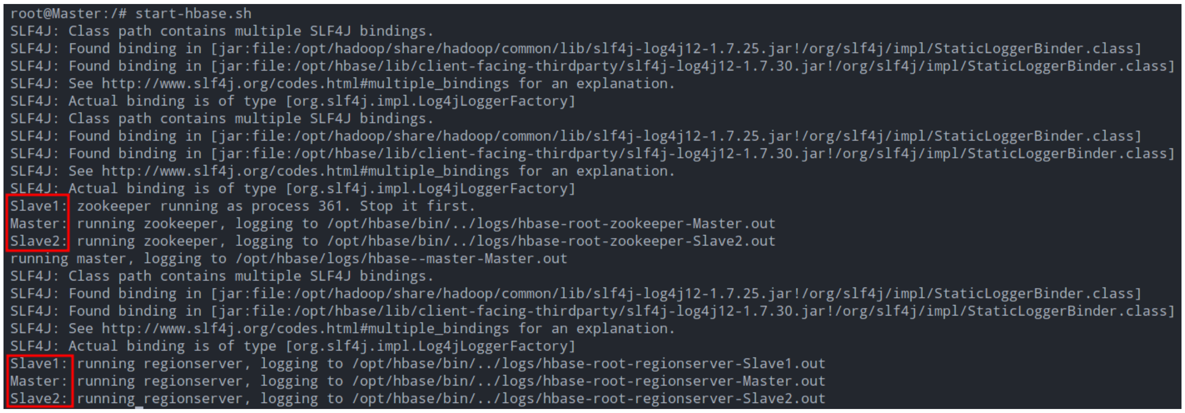
```
root@Master:/# start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /opt/hadoop/logs/yarn--resourcemanager-Master.out
Master: starting nodemanager, logging to /opt/hadoop/logs/yarn-root-nodemanager-Master.out
Slave1: starting nodemanager, logging to /opt/hadoop/logs/yarn-root-nodemanager-Slave1.out
Slave2: starting nodemanager, logging to /opt/hadoop/logs/yarn-root-nodemanager-Slave2.out
```

- start-dfs.sh



```
root@Master:/# start-dfs.sh
Starting namenodes on [Master]
Master: starting namenode, logging to /opt/hadoop/logs/hadoop-root-namenode-Master.out
Master: starting datanode, logging to /opt/hadoop/logs/hadoop-root-datanode-Master.out
Slave2: starting datanode, logging to /opt/hadoop/logs/hadoop-root-datanode-Slave2.out
Slave1: starting datanode, logging to /opt/hadoop/logs/hadoop-root-datanode-Slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /opt/hadoop/logs/hadoop-root-secondarynamenode-Master.out
```

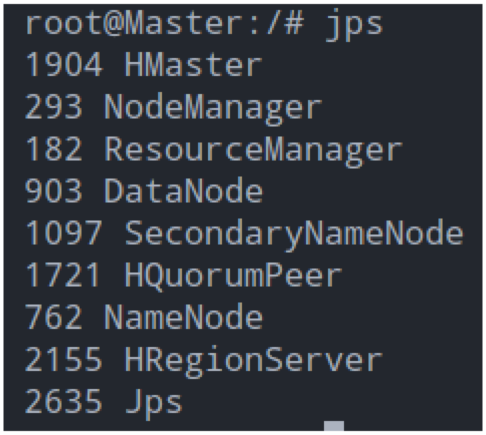
- start-hbase.sh



```
root@Master:/# start-hbase.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
Slave1: zookeeper running as process 361. Stop it first.
Master: running zookeeper, logging to /opt/hbase/bin/../logs/hbase-root-zookeeper-Master.out
Slave2: running zookeeper, logging to /opt/hbase/bin/../logs/hbase-root-zookeeper-Slave2.out
running master, logging to /opt/hbase/logs/hbase--master-Master.out
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
Slave1: running regionserver, logging to /opt/hbase/bin/../logs/hbase-root-regionserver-Slave1.out
Master: running regionserver, logging to /opt/hbase/bin/../logs/hbase-root-regionserver-Master.out
Slave2: running regionserver, logging to /opt/hbase/bin/../logs/hbase-root-regionserver-Slave2.out
```

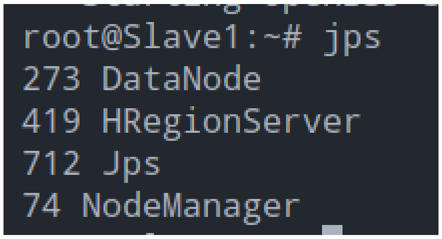
记录三个节点的jps输出

- Master



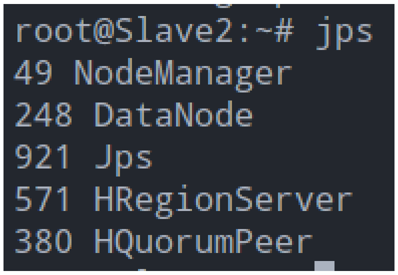
```
root@Master:/# jps
1904 HMaster
293 NodeManager
182 ResourceManager
903 DataNode
1097 SecondaryNameNode
1721 HQuorumPeer
762 NameNode
2155 HRegionServer
2635 Jps
```

- Slave1



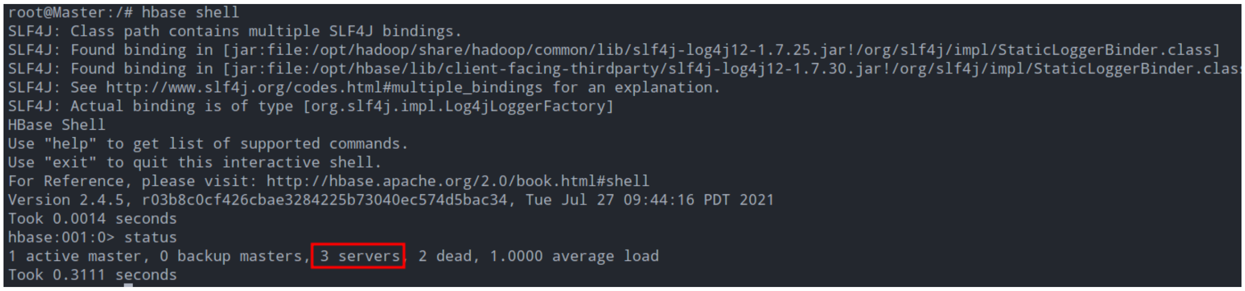
```
root@Slave1:~# jps
273 DataNode
419 HRegionServer
712 Jps
74 NodeManager
```

- Slave2



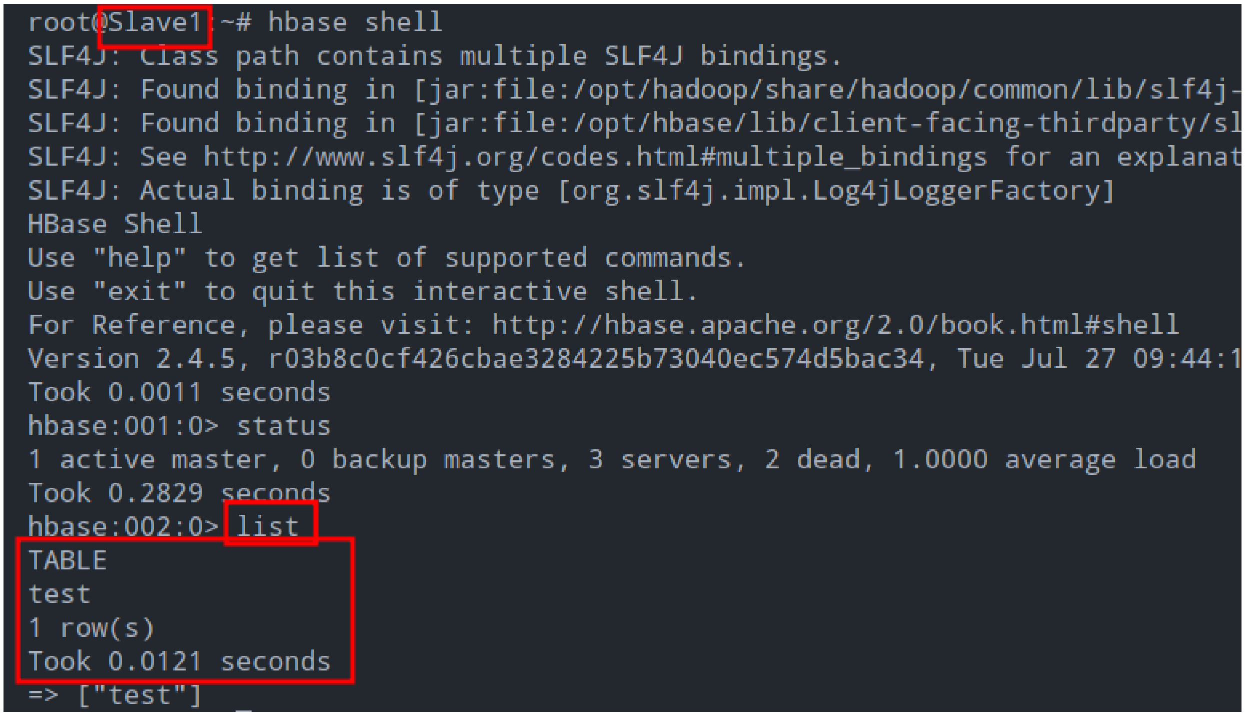
```
root@Slave2:~# jps
49 NodeManager
248 DataNode
921 Jps
571 HRegionServer
380 HQuorumPeer
```

观察Hbase Shell中的status信息，可以正确检测到三个节点



```
root@Master:/# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.clas
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.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.5, r03b8c0cf426cbae3284225b73040ec574d5bac34, Tue Jul 27 09:44:16 PDT 2021
Took 0.0014 seconds
hbase:001:0> status
1 active master, 0 backup masters, 3 servers, 2 dead, 1.0000 average load
Took 0.3111 seconds
```

且在从节点的hbase shell中可以正确读取到在Master节点上创建的表

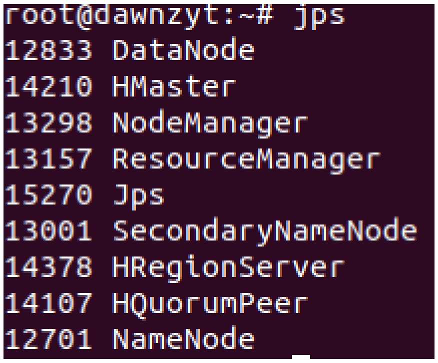


```
root@Slave1:~# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/sl
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanat
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.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.5, r03b8c0cf426cbae3284225b73040ec574d5bac34, Tue Jul 27 09:44:1
Took 0.0011 seconds
hbase:001:0> status
1 active master, 0 backup masters, 3 servers, 2 dead, 1.0000 average load
Took 0.2829 seconds
hbase:002:0> list
TABLE
test
1 row(s)
Took 0.0121 seconds
=> ["test"]
```

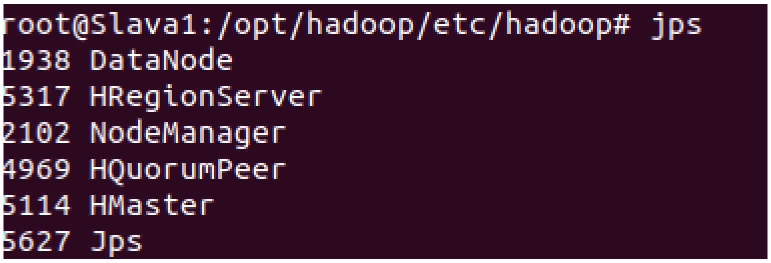
Hbase shell操作记录

- **启动相关服务**

从节点单机伪分布启动`hdfs`、`hbase`服务：



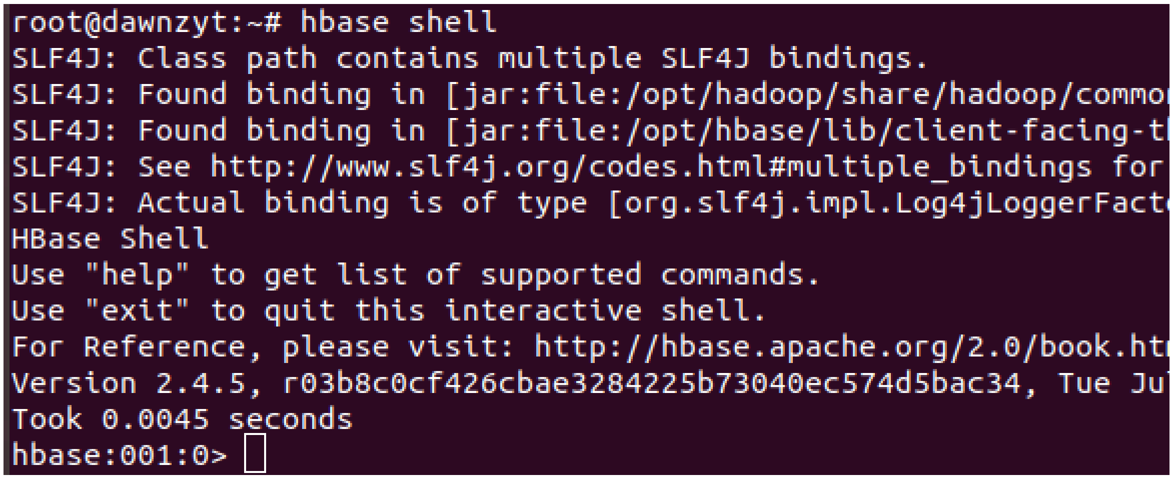
```
root@dawnzyt:~# jps
12833 DataNode
14210 HMaster
13298 NodeManager
13157 ResourceManager
15270 Jps
13001 SecondaryNameNode
14378 HRegionServer
14107 HQuorumPeer
12701 NameNode
```



```
root@Slava1:/opt/hadoop/etc/hadoop# jps
1938 DataNode
5317 HRegionServer
2102 NodeManager
4969 HQuorumPeer
5114 HMaster
5627 Jps
```

启动`hbase shell`：

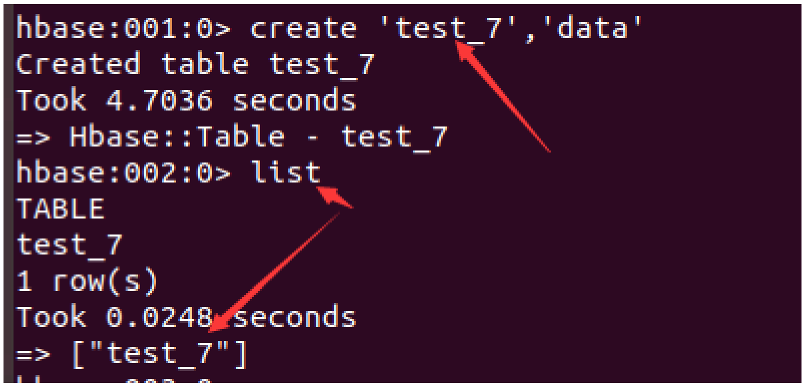
```
hbase shell
```



```
root@dawnzyt:~# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/commo
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-t
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFact
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.ht
Version 2.4.5, r03b8c0cf426cbae3284225b73040ec574d5bac34, Tue Ju
Took 0.0045 seconds
hbase:001:0> 
```

- 下面创建`test_7`表进行测试：

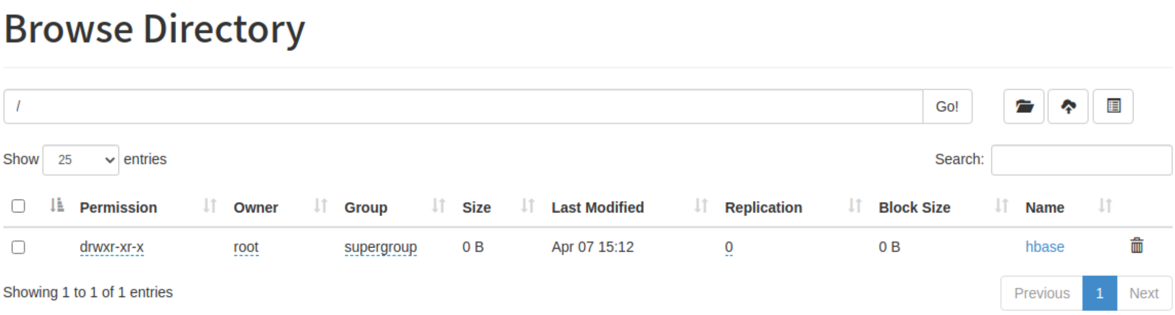
```
create 'test_7','data'
list
```



```
hbase:001:0> create 'test_7','data'
Created table test_7
Took 4.7036 seconds
=> Hbase::Table - test_7
hbase:002:0> list
TABLE
test_7
1 row(s)
Took 0.0248 seconds
=> ["test_7"]
```

- HBase在HDFS上的存储结构

在`hadoop ip:50070`web前端查看`hbase`，hbase在根目录：



Browse Directory

/ Go!

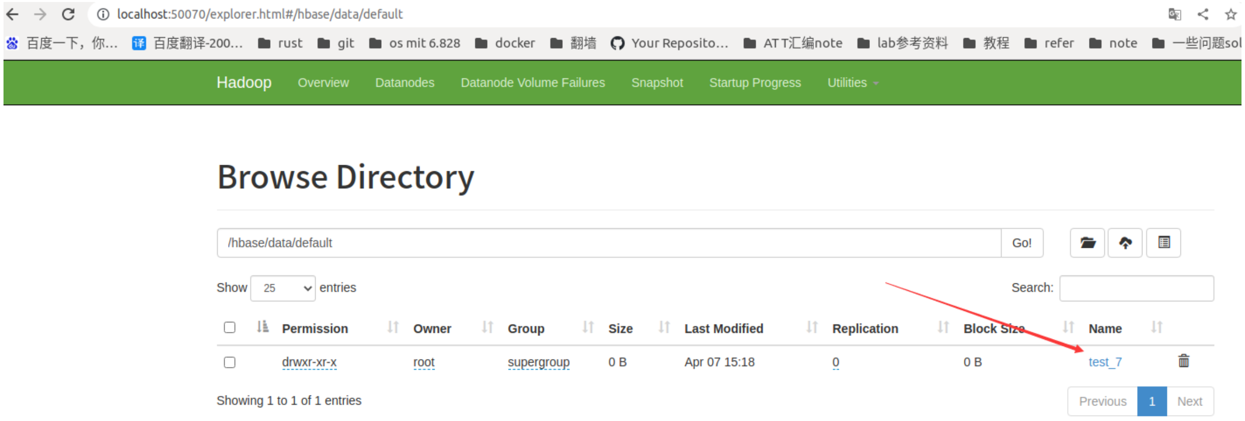
Show 25 entries Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	drwxr-xr-x	root	supergroup	0 B	Apr 07 15:12	0	0 B	hbase

Showing 1 to 1 of 1 entries

Previous 1 Next

继续查看`test_7`表，所有create的表都在目录`/hbase/data/default`下：



localhost:50070/explorer.html#/hbase/data/default
百度一下，你... 百度翻译-200... rust git os mit 6.828 docker 翻墙 Your Reposito... AT T汇编note lab参考资料 教程 refer note 一些问题sol
Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/hbase/data/default Go!

Show 25 entries Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	drwxr-xr-x	root	supergroup	0 B	Apr 07 15:18	0	0 B	test_7

Showing 1 to 1 of 1 entries

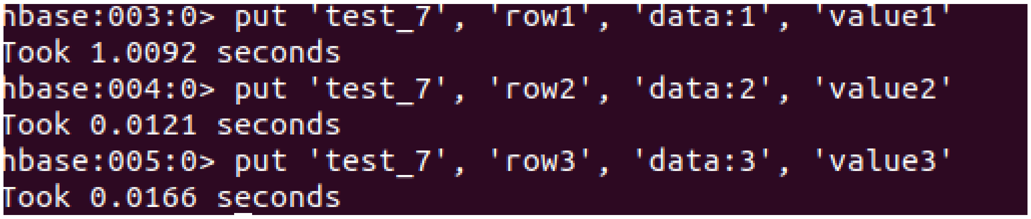
Previous 1 Next

- **test表插入、修改、删除测试**

向`test_7`表中插入3行数据：

```
put 'test_7', 'row1', 'data:1', 'value1'
put 'test_7', 'row2', 'data:2', 'value2'
put 'test_7', 'row3', 'data:3', 'value3'
```

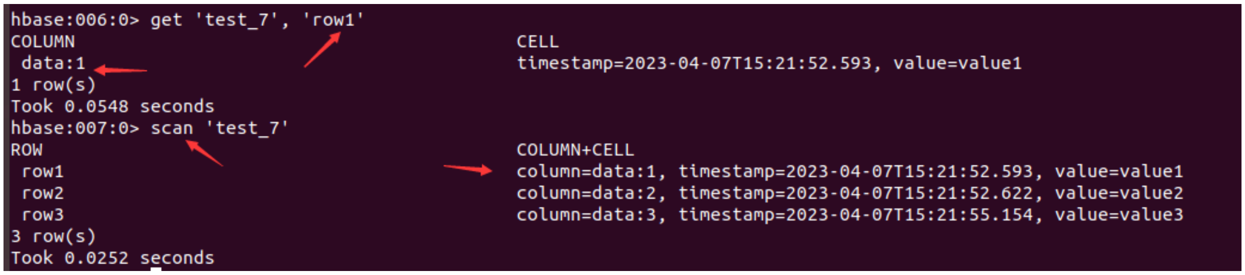
运行结果如下



```
hbase:003:0> put 'test_7', 'row1', 'data:1', 'value1'
Took 1.0092 seconds
hbase:004:0> put 'test_7', 'row2', 'data:2', 'value2'
Took 0.0121 seconds
hbase:005:0> put 'test_7', 'row3', 'data:3', 'value3'
Took 0.0166 seconds
```

取出第一行`row1`的数据，并获取列表内容

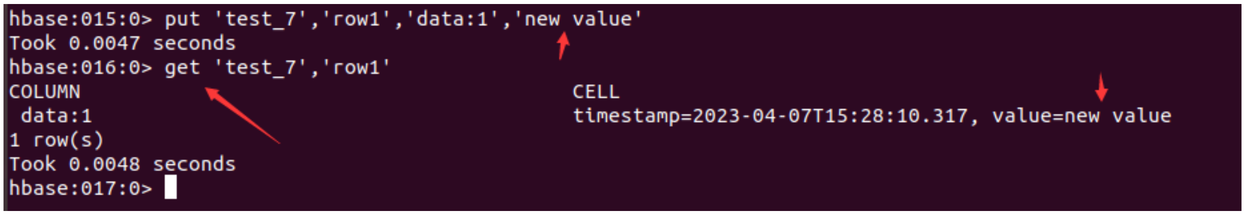
```
get 'test_7', 'row1'
scan 'test_7'
```



```
hbase:006:0> get 'test_7', 'row1'
COLUMN                                              CELL
 data:1                                             timestamp=2023-04-07T15:21:52.593, value=value1
1 row(s)
Took 0.0548 seconds
hbase:007:0> scan 'test_7'
ROW                                                 COLUMN+CELL
 row1                                               column=data:1, timestamp=2023-04-07T15:21:52.593, value=value1
 row2                                               column=data:2, timestamp=2023-04-07T15:21:52.622, value=value2
 row3                                               column=data:3, timestamp=2023-04-07T15:21:55.154, value=value3
3 row(s)
Took 0.0252 seconds
```

对`row1`的`data1`进行修改：

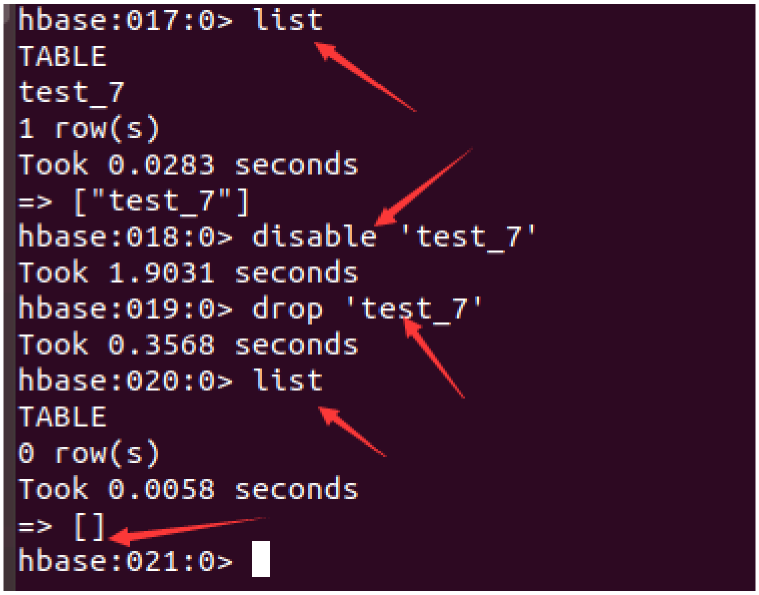
```
put 'test_7','row1','data:1','new value'
get 'test_7','row1'
```



```
hbase:015:0> put 'test_7','row1','data:1','new value'
Took 0.0047 seconds
hbase:016:0> get 'test_7','row1'
COLUMN                                              CELL
 data:1                                             timestamp=2023-04-07T15:28:10.317, value=new value
1 row(s)
Took 0.0048 seconds
hbase:017:0> 
```

删除表：

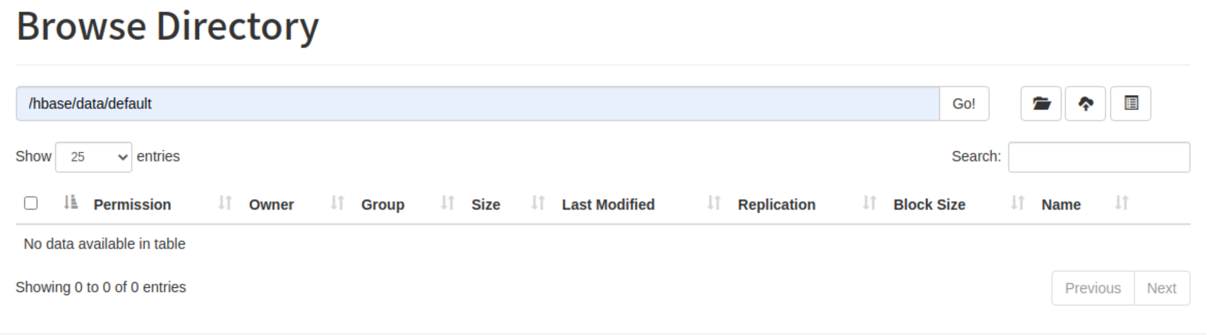
```
disable 'test_7'
drop 'test_7'
list
```



```
hbase:017:0> list
TABLE
test_7
1 row(s)
Took 0.0283 seconds
=> ["test_7"]
hbase:018:0> disable 'test_7'
Took 1.9031 seconds
hbase:019:0> drop 'test_7'
Took 0.3568 seconds
hbase:020:0> list
TABLE
0 row(s)
Took 0.0058 seconds
=> []
hbase:021:0>
```

- **web hdfs前端查看：**

如图`hbase`表文件已删除：



Browse Directory

/hbase/data/default

Show 25 entries

Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name

No data available in table

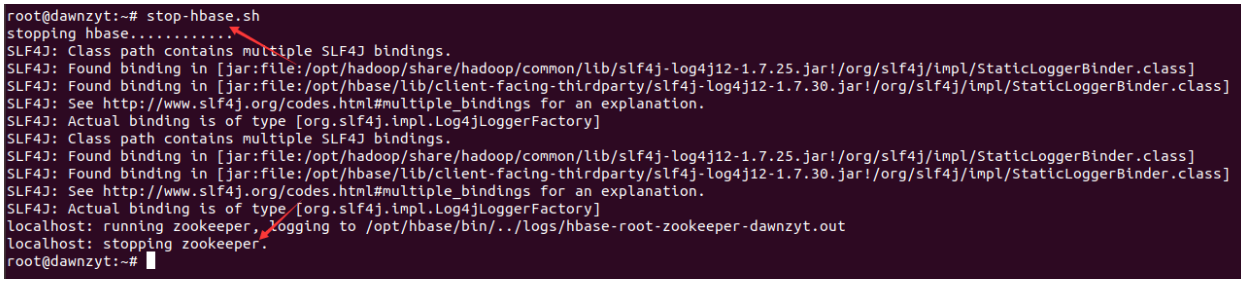
Showing 0 to 0 of 0 entries

- **exit**

最后，离开hbase后，停止hbase服务

```
stop-hbase.sh
```

如图：



```
root@dawnzyt:~# stop-hbase.sh
stopping hbase............
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.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]
localhost: running zookeeper, logging to /opt/hbase/bin/../logs/hbase-root-zookeeper-dawnzyt.out
localhost: stopping zookeeper.
root@dawnzyt:~#
```

- **namenode:**

　　这是Hadoop的NameNode服务，负责管理HDFS的元数据，如文件名、目录结构、副本数等。它使用bde2020/hadoop-namenode:2.0.0-hadoop2.7.4-java8镜像，并将/hadoop/dfs/name目录挂载到hadoop_namenode卷上，以持久化存储元数据。Namenode节点存储着整个HDFS文件系统的元数据，包括文件和目录的层次结构、文件的块列表以及块所在的Datanode节点信息等。当客户端请求读取或写入文件时，Namenode节点会提供相应的元数据信息，并告知客户端应该去哪些Datanode节点获取或存储数据块。

在Hadoop集群中，Namenode节点通常运行在一个比较高端的服务器上，因为它需要快速响应客户端的请求，并管理大量的元数据。为了保证数据的可靠性和可用性，Hadoop通常会在集群中多个Datanode之间复制数据块，从而提供容错和故障恢复的能力。当一个Datanode节点失效时，Namenode节点会检测到该节点失效，并启动复制数据块到其他节点的过程，以确保数据块的复制数量达到要求。

- **datanode:**

　　这是Hadoop的DataNode服务，负责存储HDFS的实际数据。Datanode节点通过与NameNode通信来获取关于HDFS中数据块的位置和副本状态的信息，并负责存储和管理数据块的复制。当客户端请求读取或写入文件时，Datanode节点会响应并提供相应的数据块。它使用bde2020/hadoop-datanode:2.0.0-hadoop2.7.4-java8镜像，并将/hadoop/dfs/data目录挂载到hadoop_datanode卷上，以持久化存储数据。它设置了环境变量SERVICE_PRECONDITION为"namenode:50070"，表示在namenode服务可用之前不启动。

- **resourcemanager:**

　　这是Hadoop的ResourceManager服务，负责管理YARN集群的资源分配和调度。它使用bde2020/hadoop-resourcemanager:2.0.0-hadoop2.7.4-java8镜像

ResourceManager节点维护着整个YARN集群中的资源信息，包括CPU、内存和磁盘等资源的使用情况，以及所有应用程序的资源请求和分配情况。当一个应用程序需要在YARN集群中启动时，它会向ResourceManager节点发送资源请求，并等待ResourceManager节点分配相应的资源。ResourceManager节点会根据集群中资源的使用情况和应用程序的需求，动态地调整资源分配，以确保集群资源的最优利用和所有应用程序的公平竞争。

在Hadoop集群中，ResourceManager节点通常运行在一个比较高端的服务器上，因为它需要处理大量的资源请求和分配。为了提高ResourceManager节点的可用性和容错能力，Hadoop通常会使用多个ResourceManager节点，以便在一个节点失效时，其他节点可以接替其工作。

- **nodemanager1:**

　　这是Hadoop的NodeManager服务，负责管理YARN集群中每个节点的资源使用和任务执行。它使用bde2020/hadoop-nodemanager:2.0.0-hadoop2.7.4-java8镜像.

它是YARN的从节点，负责管理在该节点上运行的应用程序和相应的资源分配。

NodeManager节点运行在集群中的每个节点上，它的主要功能是监视和管理该节点上的容器。一个容器是YARN中的一个资源分配单位，它包含了一个或多个应用程序进程以及它们所需的资源，例如内存、CPU和磁盘等。

NodeManager节点通过与ResourceManager节点通信来获取该节点上的资源信息和应用程序的运行状态，以便更好地管理和监视应用程序的运行。当一个应用程序需要启动时，ResourceManager节点会向可用的NodeManager节点发送相应的命令，以启动该应用程序进程并分配相应的资源。

- **historyserver:**

　　这是Hadoop的HistoryServer服务，负责存储和展示YARN集群中已完成的任务的历史信息。它使用bde2020/hadoop-historyserver:2.0.0-hadoop2.7.4-java8镜像，并将/hadoop/yarn/timeline目录挂载到hadoop_historyserver卷上，以持久化存储历史数据。

当一个应用程序完成后，它的运行信息，包括应用程序的日志、执行的命令、使用的资源等，都会被写入Hadoop集群中的一个历史记录文件中。HistoryServer节点负责读取和管理这些历史记录文件，并提供相应的查询和展示功能，以便用户可以方便地查询和分析应用程序的历史信息。

在Hadoop集群中，HistoryServer节点通常运行在一个较为中等的服务器上，因为它需要存储和管理大量的历史记录文件，并提供相应的查询和展示功能。为了提高HistoryServer节点的可用性和容错能力，Hadoop通常会使用多个HistoryServer节点，以便在一个节点失效时，其他节点可以接替其工作。它也从./hadoop.env文件中读取其他环境变量。它将8188端口映射到宿主机上，以提供Web界面。

- **zoo:**

　　这是ZooKeeper服务，负责提供分布式协调和配置管理功能。它使用zookeeper:3.4.10镜像，并设置了环境变量ZOO_MY_ID为1和ZOO_SERVERS为server.1=0.0.0.0:2888:388

ZooKeeper主要由三个组件组成：Leader节点、Follower节点和Observer节点。

在一个ZooKeeper集群中，Leader节点是主节点，负责协调和管理集群中的其他节点。Follower节点是从节点，它们负责接收来自Leader节点的更新信息，并将其同步到自己的数据存储中。Observer节点是另一种从节点，它们类似于Follower节点，但不参与Leader选举和投票。

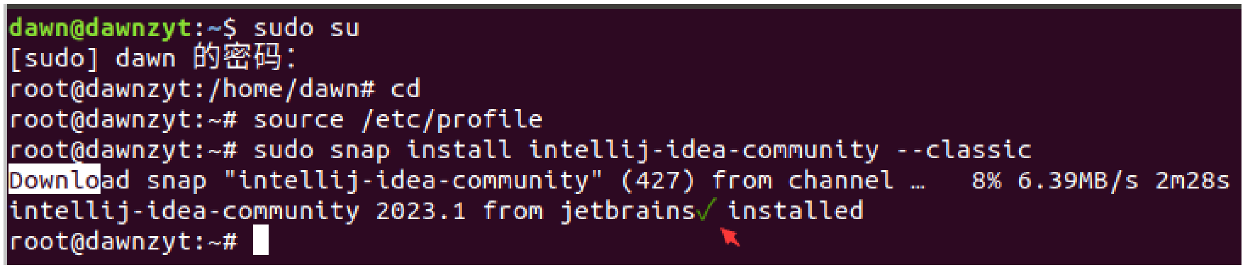
ZooKeeper的主要功能包括协调和同步、配置管理、命名服务和分布式锁等。例如，一个分布式应用程序可以使用ZooKeeper来实现集群中节点之间的同步和协调，以确保它们的状态和行为一致。另外，ZooKeeper还可以用于实现分布式锁机制，以确保在多个节点同时访问共享资源时的互斥性和同步性。

JAVA操作Hbase

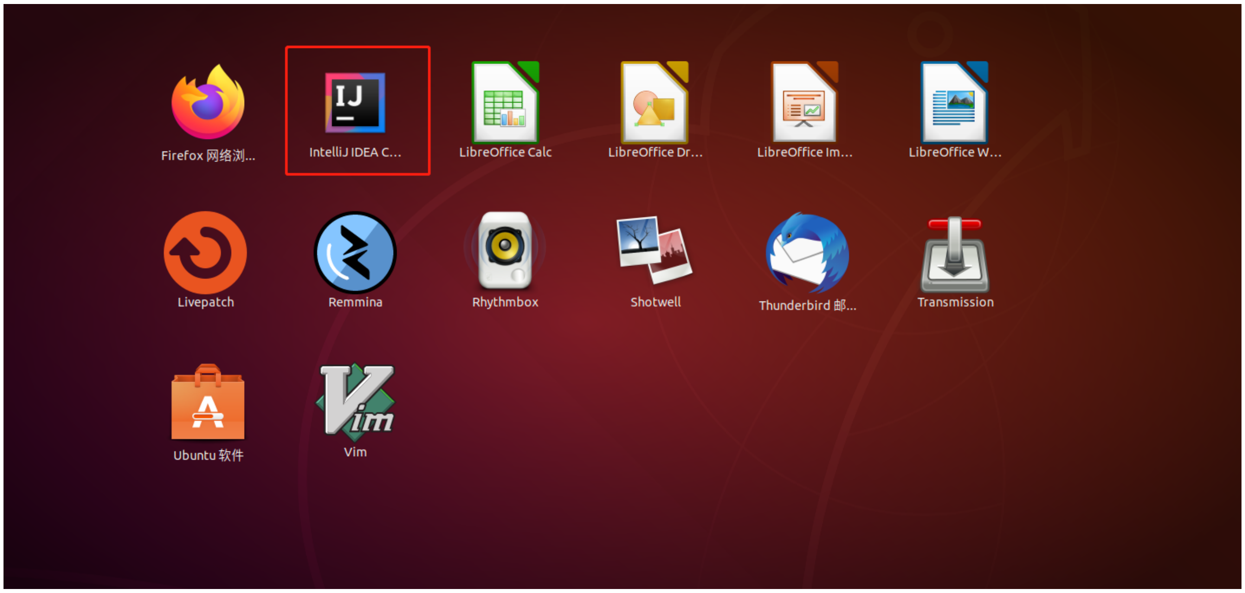
环境配置

1. 安装`intellij` ide

```
sudo snap install intellij-idea-community --classic
```

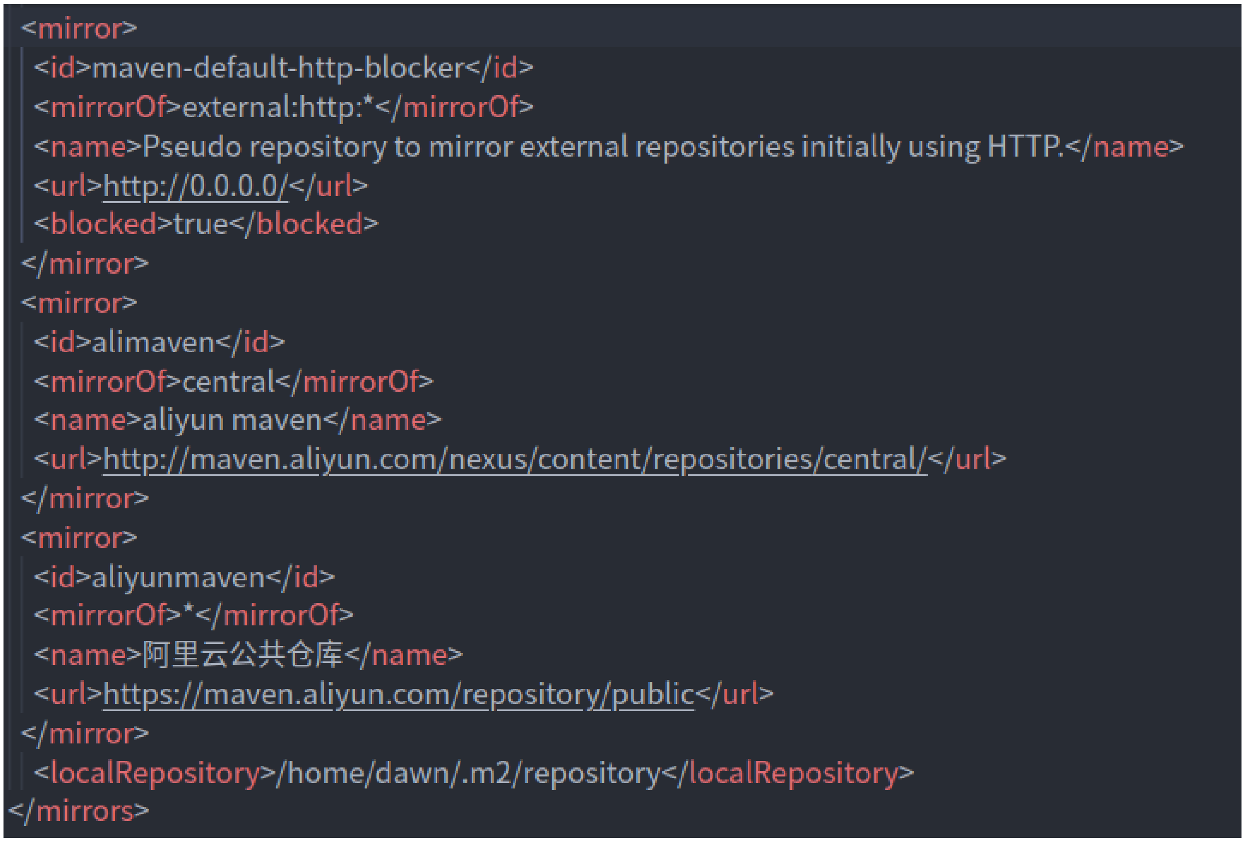


```
dawn@dawnzyt:~$ sudo su
[sudo] dawn 的密码：
root@dawnzyt:/home/dawn# cd
root@dawnzyt:~# source /etc/profile
root@dawnzyt:~# sudo snap install intellij-idea-community --classic
Download snap "intellij-idea-community" (427) from channel …   8% 6.39MB/s 2m28s
intellij-idea-community 2023.1 from jetbrains✓ installed
root@dawnzyt:~# 
```

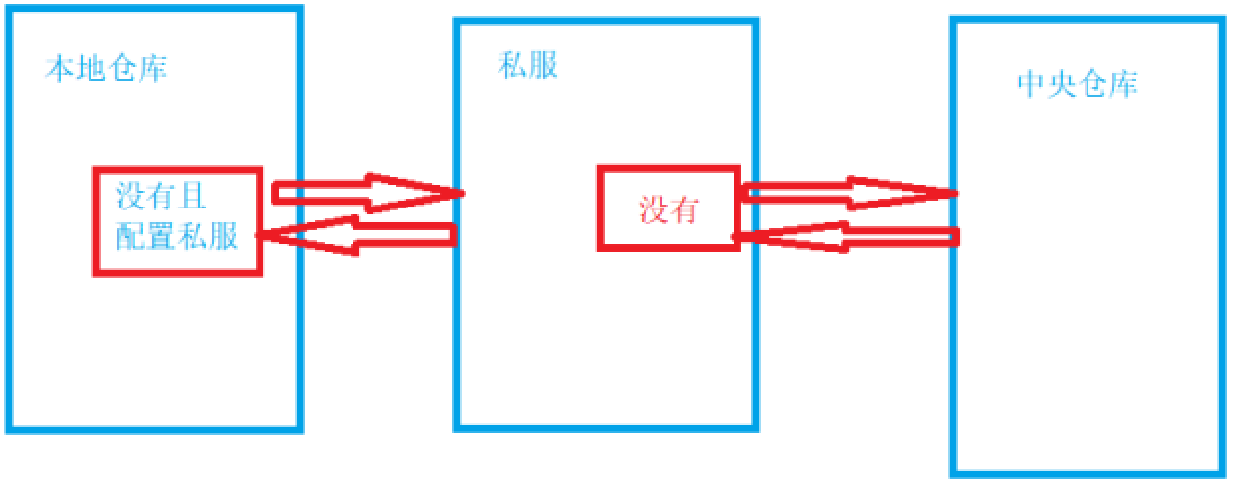


1. 安装`maven`3.8.8并配置远程镜像和本地镜像。

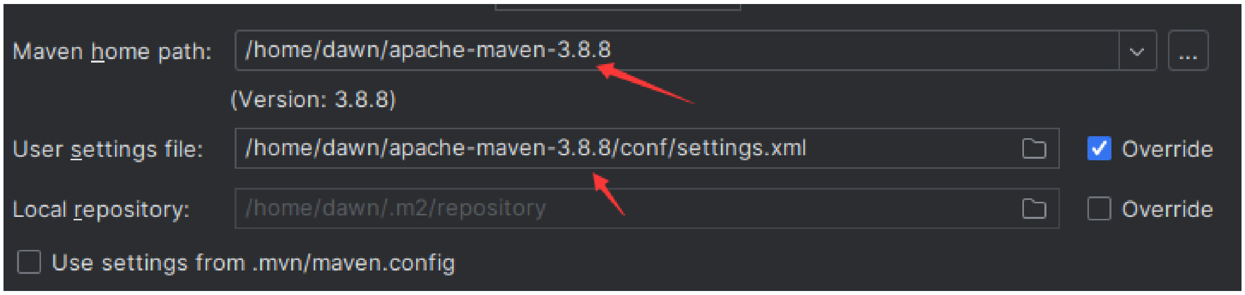
```
wget https://dlcdn.apache.org/maven/maven-3/3.8.8/binaries/apache-maven-3.8.8-bin.tar.gz
tar -zxvf ~/apache-maven-3.8.8-bin.tar.gz
```



```
  <mirror>
    <id>maven-default-http-blocker</id>
    <mirrorOf>external:http:*</mirrorOf>
    <name>Pseudo repository to mirror external repositories initially using HTTP.</name>
    <url>http://0.0.0.0/</url>
    <blocked>true</blocked>
  </mirror>
  <mirror>
    <id>alimaven</id>
    <mirrorOf>central</mirrorOf>
    <name>aliyun maven</name>
    <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
  </mirror>
  <mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>阿里云公共仓库</name>
    <url>https://maven.aliyun.com/repository/public</url>
  </mirror>
    <localRepository>/home/dawn/.m2/repository</localRepository>
</mirrors>
```



1. 新建项目时配置`maven3.8.8`管理项目的jar包环境依赖


Maven home path: /home/dawn/apache-maven-3.8.8

(Version: 3.8.8)

User settings file: /home/dawn/apache-maven-3.8.8/conf/settings.xml ☑ Override

Local repository: /home/dawn/.m2/repository □ Override

□ Use settings from .mvn/maven.config

- 拉取远程阿里云远程仓库的jar包时报错`Could not transfer artifact org.apache.maven.plugins*`

```
http\://maven.aliyun.com/nexus/content/repositories/central/.error=Could not
transfer
 artifact org.apache.hbase\:hbase\:pom\:2.4.5 from/to alimaven
(http\://maven.aliyun.com/
 nexus/content/repositories/central/)\: java.lang.RuntimeException\: Unexpected
error\:
 java.security.InvalidAlgorithmParameterException\: the trustAnchors parameter must
be
 non-empty ...
```

原因： SSL配置和信任库出现问题

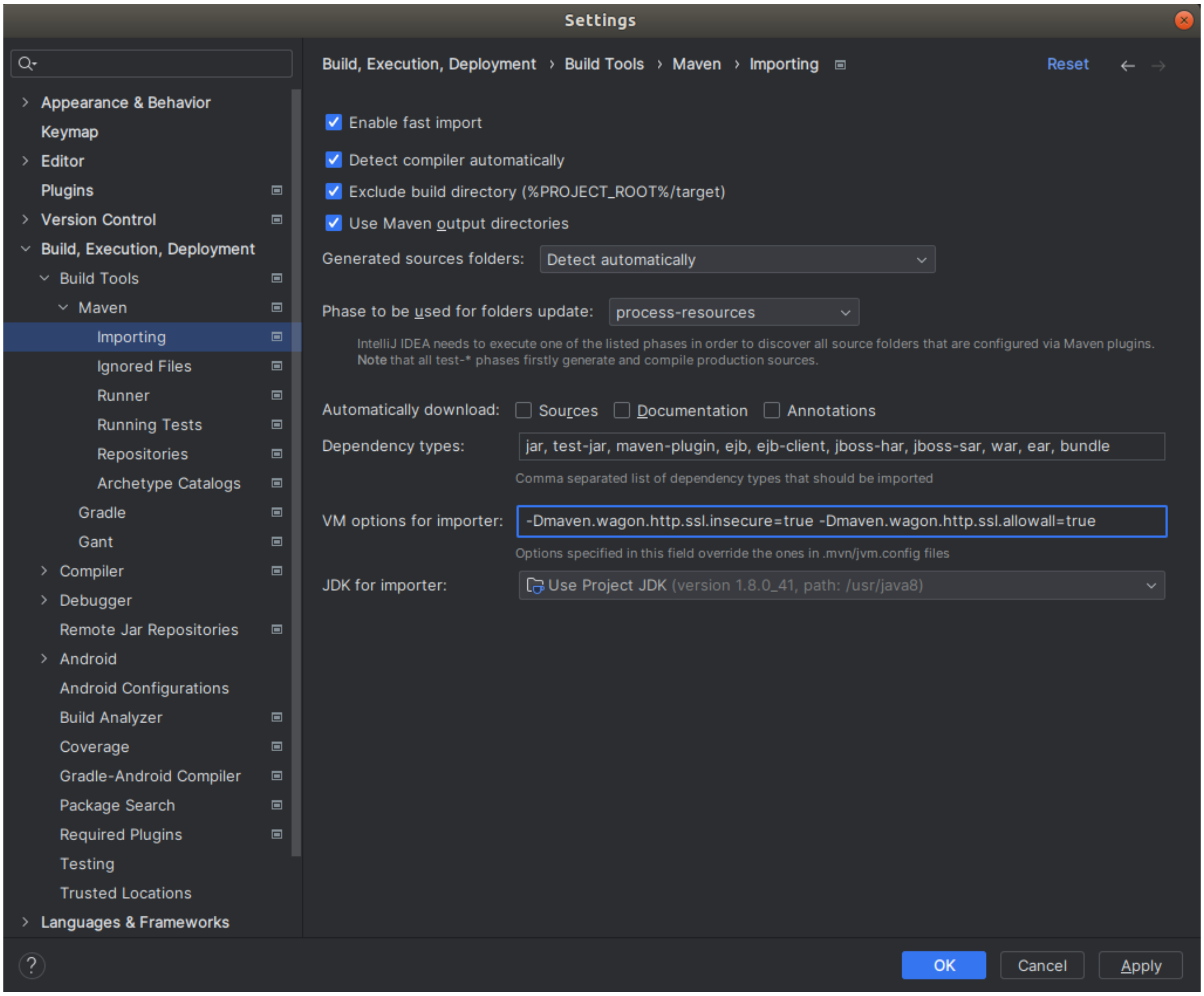
解决办法： 在IDEA中设置`maven`编译时忽略HTTPS的SSL证书验证，加上参数：

- `Maven/Importing`

```
-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
```

- `Maven/Runner`

```
-Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true
-Dmaven.wagon.http.ssl.ignore.validity.dates=true
```



Build, Execution, Deployment › Build Tools › Maven › Importing

☑ Enable fast import

☑ Detect compiler automatically

☑ Exclude build directory (%PROJECT_ROOT%/target)

☑ Use Maven output directories

Generated sources folders: Detect automatically

Phase to be used for folders update: process-resources

IntelliJ IDEA needs to execute one of the listed phases in order to discover all source folders that are configured via Maven plugins.
Note that all test-* phases firstly generate and compile production sources.

Automatically download: □ Sources □ Documentation □ Annotations

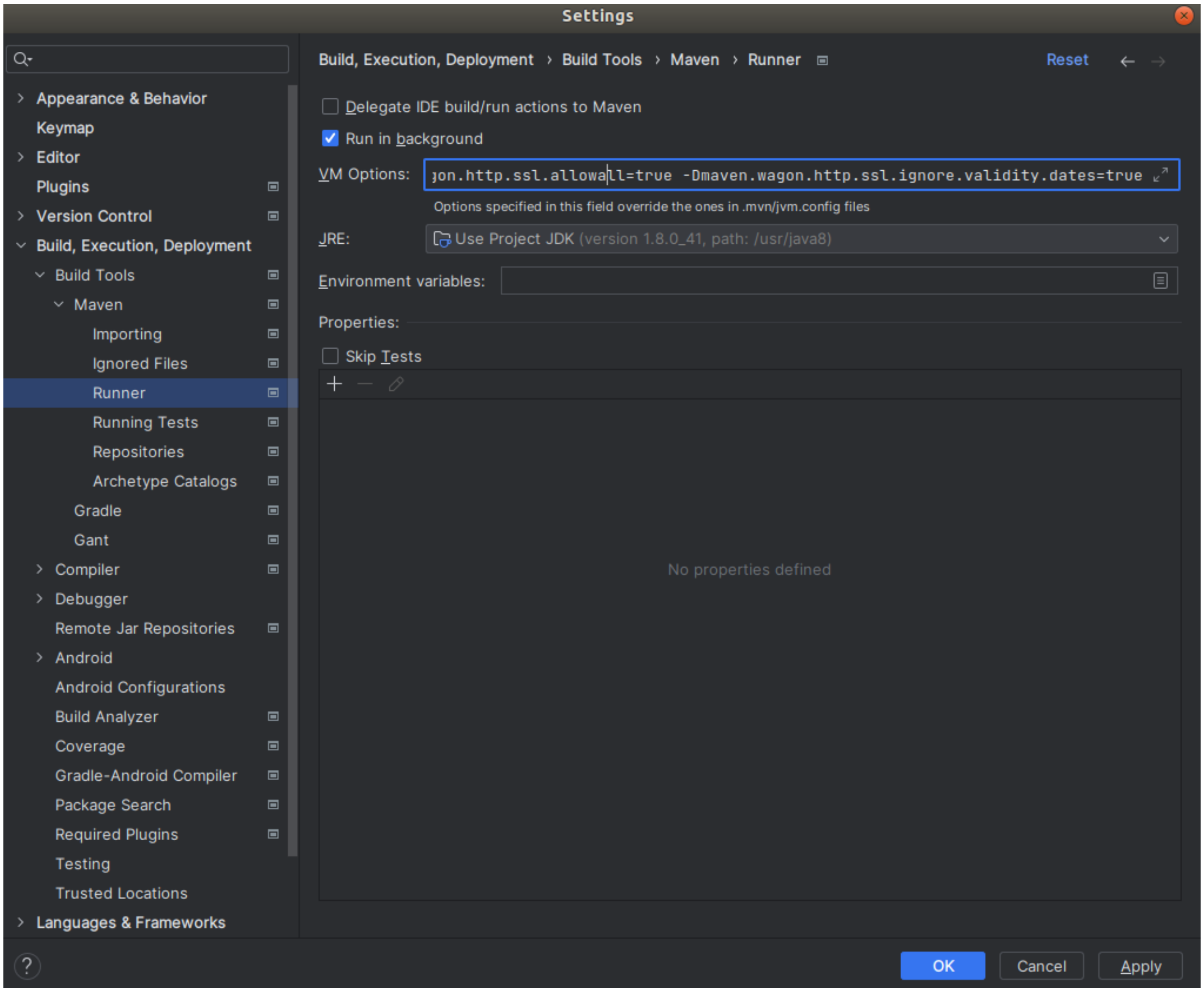
Dependency types: jar, test-jar, maven-plugin, ejb, ejb-client, jboss-har, jboss-sar, war, ear, bundle

Comma separated list of dependency types that should be imported

VM options for importer: -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true

Options specified in this field override the ones in .mvn/jvm.config files

JDK for importer: Use Project JDK (version 1.8.0_41, path: /usr/java8)



Build, Execution, Deployment › Build Tools › Maven › Runner

□ Delegate IDE build/run actions to Maven

☑ Run in background

VM Options: `gon.http.ssl.allowall=true -Dmaven.wagon.http.ssl.ignore.validity.dates=true`

Options specified in this field override the ones in .mvn/jvm.config files

JRE: Use Project JDK (version 1.8.0_41, path: /usr/java8)

Environment variables:

Properties:

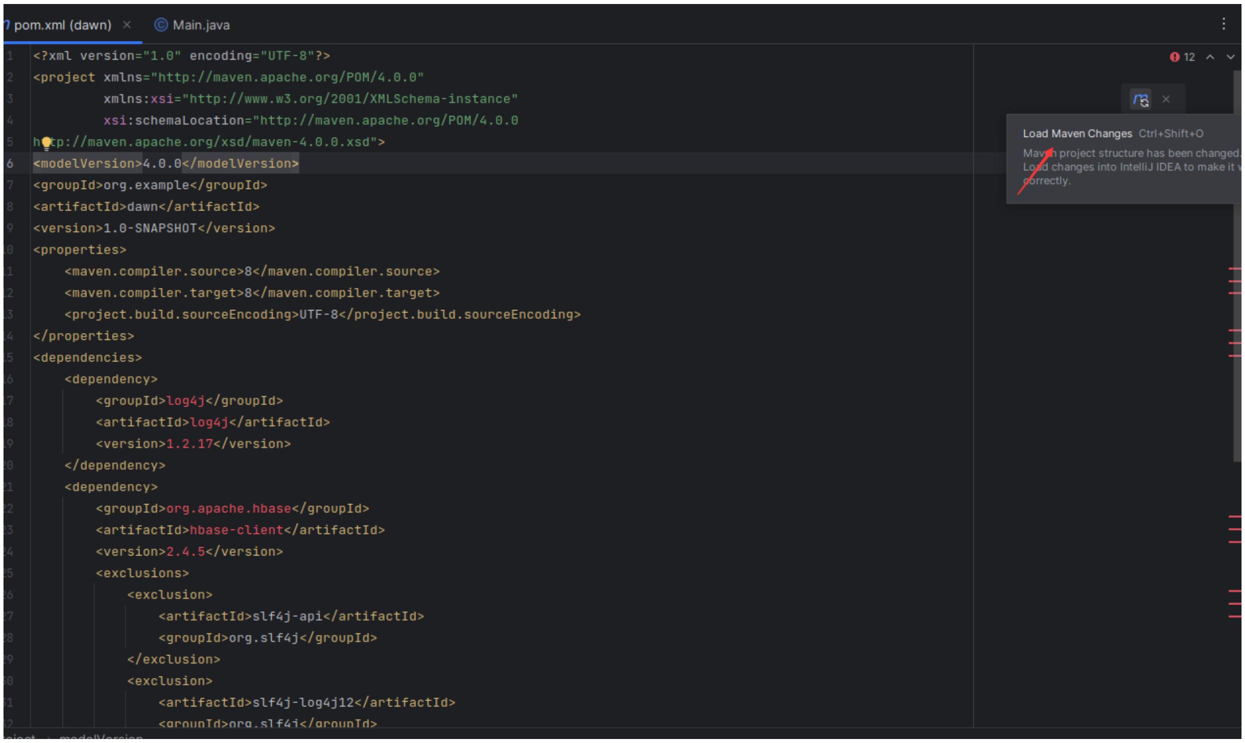
□ Skip Tests

- 本地没有`plugins`的报错

```
No plugin found for prefix 'compile' in the current project and in the plugin group
...
```

确保远程仓库无误后重新pull安装必需的plugins执行:`mvn clean install`安装plugins到本地库

1. 配置`maven`项目的`pom.xml`配置文件引入hbase相关`jar`包

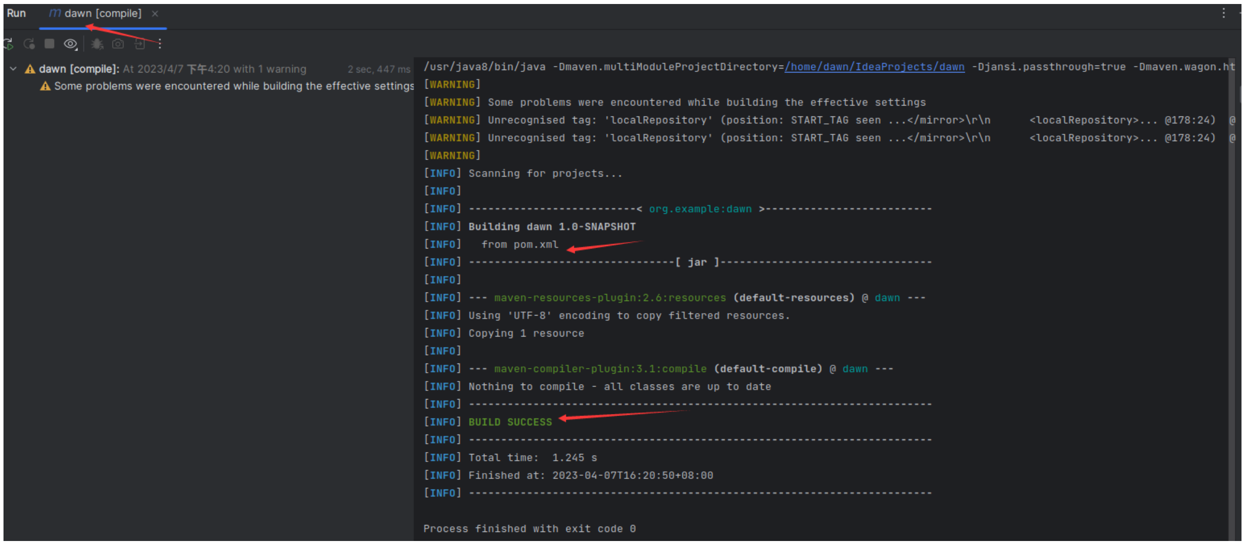


```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</groupId>
<artifactId>dawn</artifactId>
<version>1.0-SNAPSHOT</version>
<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>1.2.17</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hbase</groupId>
        <artifactId>hbase-client</artifactId>
        <version>2.4.5</version>
        <exclusions>
            <exclusion>
                <artifactId>slf4j-api</artifactId>
                <groupId>org.slf4j</groupId>
            </exclusion>
            <exclusion>
                <artifactId>slf4j-log4j12</artifactId>
                <groupId>org.slf4j</groupId>
```

Load Maven Changes Ctrl+Shift+O

Maven project structure has been changed.
Load changes into IntelliJ IDEA to make it w
correctly.

重新编译`maven`，从远程仓库aliyun下载`pom.xml`所有依赖的`jar`包：



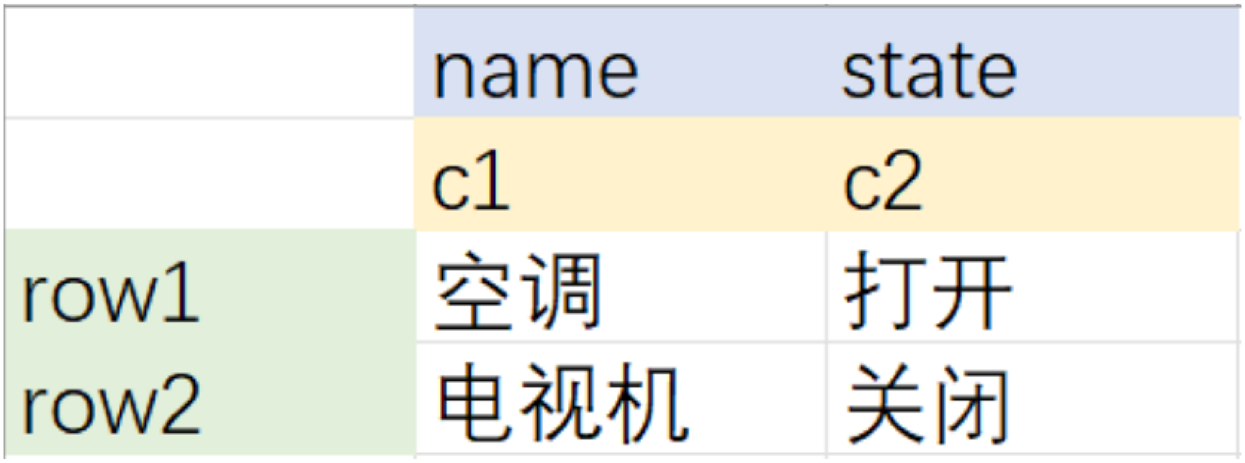
dawn [compile]: At 2023/4/7 下午4:20 with 1 warning 2 sec, 447 ms
Some problems were encountered while building the effective settings

```
/usr/java8/bin/java -Dmaven.multiModuleProjectDirectory=/home/dawn/IdeaProjects/dawn -Djansi.passthrough=true -Dmaven.wagon.ht
[WARNING]
[WARNING] Some problems were encountered while building the effective settings
[WARNING] Unrecognised tag: 'localRepository' (position: START_TAG seen ...</mirror>\r\n      <localRepository>... @178:24)  @
[WARNING] Unrecognised tag: 'localRepository' (position: START_TAG seen ...</mirror>\r\n      <localRepository>... @178:24)  @
[WARNING]
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< org.example:dawn >-------------------------
[INFO] Building dawn 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ dawn ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ dawn ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.245 s
[INFO] Finished at: 2023-04-07T16:20:50+08:00
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

1. 配置`log4j`

测试java控制hbase代码

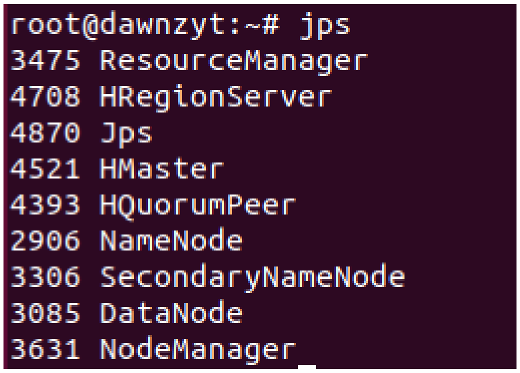
实验中给出的java代码的效果是创建一个`DeviceState`表，添加`name`，`state`两个列族，并添加数据，最终应得到的表：



	name	state
	c1	c2
row1	空调	打开
row2	电视机	关闭

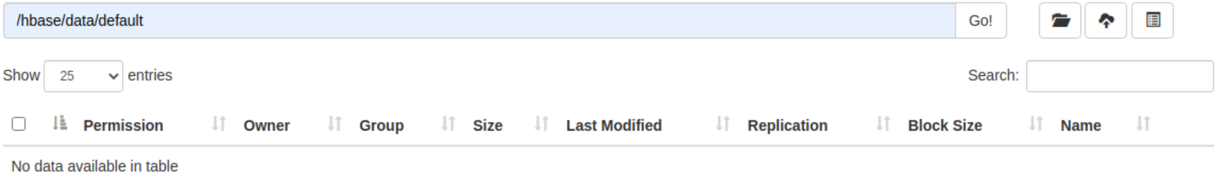
1. 先启动hdfs和hbase等服务

```
start-dfs.sh
start-yarn.sh
start-hbase.sh
```



```
root@dawnzyt:~# jps
3475 ResourceManager
4708 HRegionServer
4870 Jps
4521 HMaster
4393 HQuorumPeer
2906 NameNode
3306 SecondaryNameNode
3085 DataNode
3631 NodeManager
```

在hdfs web上先确认`hbase`中的表为空：



/hbase/data/default

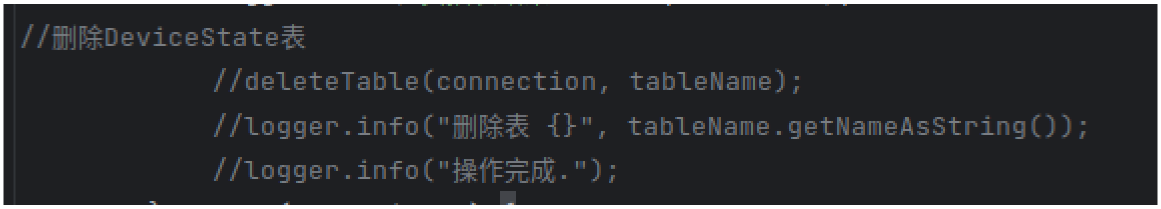
Show 25 entries

Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
No data available in table								

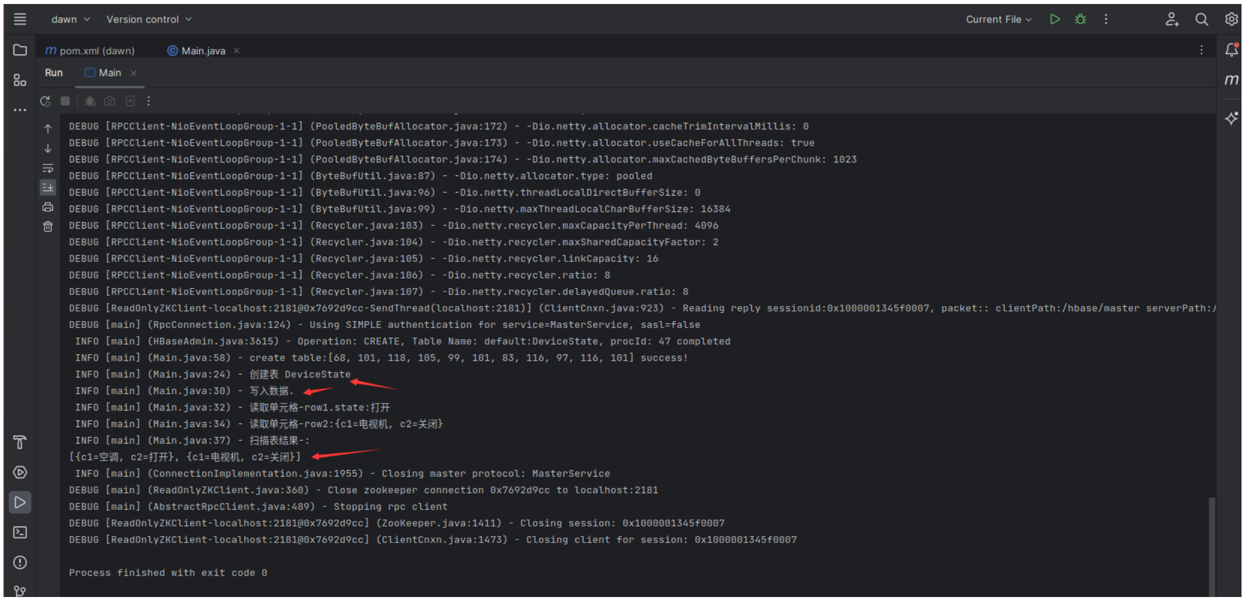
1. 运行java代码操作`hbase`创建`DeviceState`表

先注释掉源码的删除表操作:



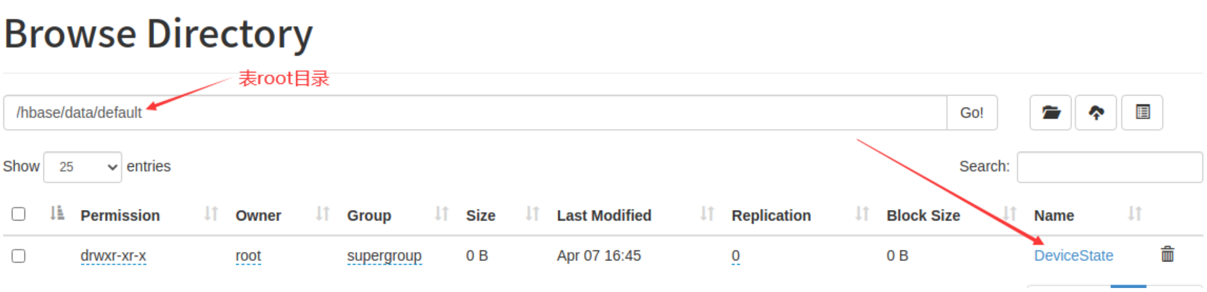
```
//删除DeviceState表
            //deleteTable(connection, tableName);
            //logger.info("删除表 {}", tableName.getNameAsString());
            //logger.info("操作完成.");
```

运行java代码写入表：



```
DEBUG [RPCClient-NioEventLoopGroup-1-1] (PooledByteBufAllocator.java:172) - -Dio.netty.allocator.cacheTrimIntervalMillis: 0
DEBUG [RPCClient-NioEventLoopGroup-1-1] (PooledByteBufAllocator.java:173) - -Dio.netty.allocator.useCacheForAllThreads: true
DEBUG [RPCClient-NioEventLoopGroup-1-1] (PooledByteBufAllocator.java:174) - -Dio.netty.allocator.maxCachedByteBuffersPerChunk: 1023
DEBUG [RPCClient-NioEventLoopGroup-1-1] (ByteBufUtil.java:87) - -Dio.netty.allocator.type: pooled
DEBUG [RPCClient-NioEventLoopGroup-1-1] (ByteBufUtil.java:96) - -Dio.netty.threadLocalDirectBufferSize: 0
DEBUG [RPCClient-NioEventLoopGroup-1-1] (ByteBufUtil.java:99) - -Dio.netty.maxThreadLocalCharBufferSize: 16384
DEBUG [RPCClient-NioEventLoopGroup-1-1] (Recycler.java:103) - -Dio.netty.recycler.maxCapacityPerThread: 4096
DEBUG [RPCClient-NioEventLoopGroup-1-1] (Recycler.java:104) - -Dio.netty.recycler.maxSharedCapacityFactor: 2
DEBUG [RPCClient-NioEventLoopGroup-1-1] (Recycler.java:105) - -Dio.netty.recycler.linkCapacity: 16
DEBUG [RPCClient-NioEventLoopGroup-1-1] (Recycler.java:106) - -Dio.netty.recycler.ratio: 8
DEBUG [RPCClient-NioEventLoopGroup-1-1] (Recycler.java:107) - -Dio.netty.recycler.delayedQueue.ratio: 8
DEBUG [ReadOnlyZKClient-localhost:2181@0x7692d9cc-SendThread(localhost:2181)] (ClientCnxn.java:923) - Reading reply sessionid:0x1000001345f0007, packet:: clientPath:/hbase/master serverPath:/
DEBUG [main] (RpcConnection.java:124) - Using SIMPLE authentication for service=MasterService, sasl=false
 INFO [main] (HBaseAdmin.java:3615) - Operation: CREATE, Table Name: default:DeviceState, procId: 47 completed
 INFO [main] (Main.java:58) - create table:[68, 101, 118, 105, 99, 101, 83, 116, 97, 116, 101] success!
 INFO [main] (Main.java:24) - 创建表 DeviceState
 INFO [main] (Main.java:30) - 写入数据.
 INFO [main] (Main.java:32) - 读取单元格-row1.state:打开
 INFO [main] (Main.java:34) - 读取单元格-row2:{c1=电视机, c2=关闭}
 INFO [main] (Main.java:37) - 扫描表结果-:
[{c1=空调, c2=打开}, {c1=电视机, c2=关闭}]
 INFO [main] (ConnectionImplementation.java:1955) - Closing master protocol: MasterService
DEBUG [main] (ReadOnlyZKClient.java:360) - Close zookeeper connection 0x7692d9cc to localhost:2181
DEBUG [main] (AbstractRpcClient.java:489) - Stopping rpc client
DEBUG [ReadOnlyZKClient-localhost:2181@0x7692d9cc] (ZooKeeper.java:1411) - Closing session: 0x1000001345f0007
DEBUG [ReadOnlyZKClient-localhost:2181@0x7692d9cc] (ClientCnxn.java:1473) - Closing client for session: 0x1000001345f0007

Process finished with exit code 0
```

在hdfs的web上查看java创建的表`DeviceState`：



Browse Directory

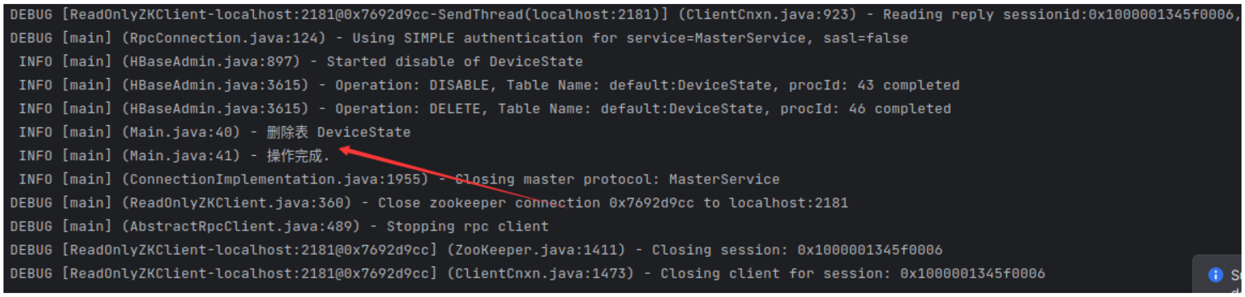
表root目录

/hbase/data/default Go!

Show 25 entries   Search:

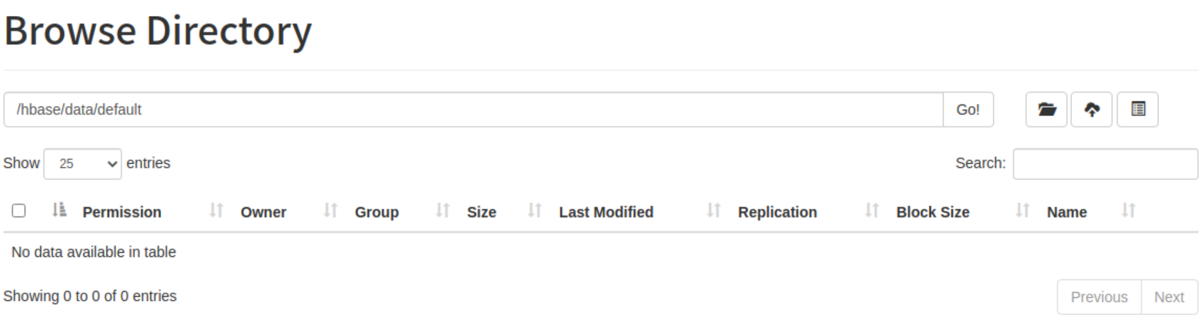
□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
□	drwxr-xr-x	root	supergroup	0 B	Apr 07 16:45	0	0 B	DeviceState

1. 用java操作hbase删除表`DeviceState`



```
DEBUG [ReadOnlyZKClient-localhost:2181@0x7692d9cc-SendThread(localhost:2181)] (ClientCnxn.java:923) - Reading reply sessionid:0x1000001345f0006,
DEBUG [main] (RpcConnection.java:124) - Using SIMPLE authentication for service=MasterService, sasl=false
 INFO [main] (HBaseAdmin.java:897) - Started disable of DeviceState
 INFO [main] (HBaseAdmin.java:3615) - Operation: DISABLE, Table Name: default:DeviceState, procId: 43 completed
 INFO [main] (HBaseAdmin.java:3615) - Operation: DELETE, Table Name: default:DeviceState, procId: 46 completed
 INFO [main] (Main.java:40) - 删除表 DeviceState
 INFO [main] (Main.java:41) - 操作完成.
 INFO [main] (ConnectionImplementation.java:1955) - Closing master protocol: MasterService
DEBUG [main] (ReadOnlyZKClient.java:360) - Close zookeeper connection 0x7692d9cc to localhost:2181
DEBUG [main] (AbstractRpcClient.java:489) - Stopping rpc client
DEBUG [ReadOnlyZKClient-localhost:2181@0x7692d9cc] (ZooKeeper.java:1411) - Closing session: 0x1000001345f0006
DEBUG [ReadOnlyZKClient-localhost:2181@0x7692d9cc] (ClientCnxn.java:1473) - Closing client for session: 0x1000001345f0006
```

hdfs前端确认表已删除：



Browse Directory

/hbase/data/default Go!

Show 25 entries   Search:

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
No data available in table								

Showing 0 to 0 of 0 entries

总结

HBase 使用 HDFS 作为其底层的存储系统，将数据以块（Block）的形式存储在 HDFS 上。与传统的关系型数据库相比，HBase 具有更好的水平扩展性和高并发性能。

对于单机伪分布式，我们完成了 Ubuntu 单机伪分布式 HBase 的配置，并初步使用 HBase shell 操作存储在 HDFS 上的 HBase 数据库。我们还使用 Maven 集成配置 HBase 相关的 Java 环境，实现了使用高级语言 Java 操纵 HBase。

对于完全分布式，我们使用docker-compose模拟了Hbase完全分布式部署。我们在配置Hbase的过程中，需要注意设置regionservesr和hbase.zookeeper.quorum以及hbase数据存储路径等参数。通过脚本化的操作方式，我们提高了配置效率，同时降低了出错的可能性。

总的来说，通过配置 HDFS、HBase 以及探索 HBase 的数据模型和集群架构，我们深入了解了 HBase 的特点、优势和应用场景，为后续的大数据处理和分析工作打下了坚实的基础。