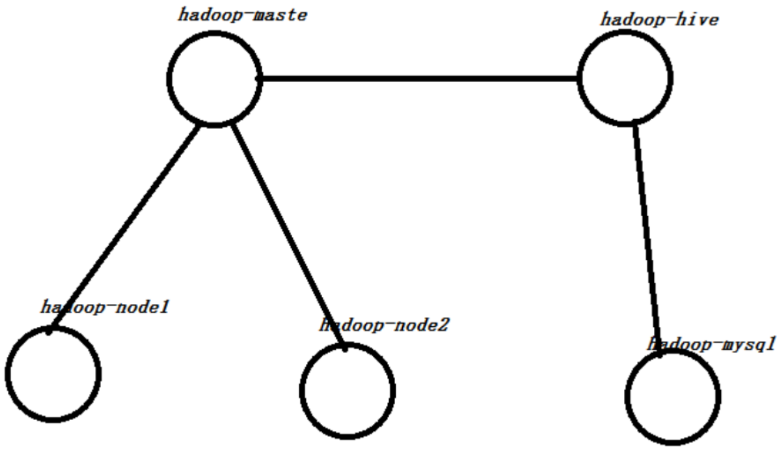
基于docker的spark环境搭建

1.镜像制作方案

我们要使用Docker来搭建hadoop,spark，hive，及mysql的集群，首先使用Dockerfile制作镜像，把相关的软件拷贝到约定好的目录下，把配置文件在外面先配置好，再拷贝移动到hadoop,spark,hive的配置目录，需要注意一点在spark中读取hive中的数据，需要把hive-site.xml拷贝到spark的conf目录；为了能使得mysql能从其它节点被访问到，要配置mysql的访问权限。

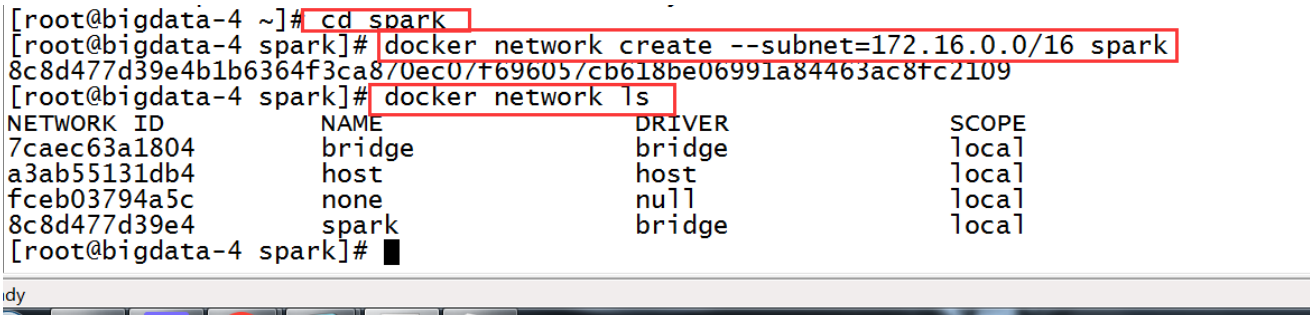
2.集群整体架构 一共5个节点，即启动5个容器。hadoop-maste,hadoop-node1,hadoop-node2这三个容器里面安装hadoop和spark集群，hadoop-hive这个容器安装hive，hadoop-mysql这个容器安装mysql数据库。



spark中可以在sparksession的builder中通过enableHiveSupport()方法，启用对hive数据仓库表操作的支持。mysql用于保存hive的元数据。当然也spark中的DataFrame也可以通过write方法将数据写入mysql中，具体的操作后面课程会详细讲解。

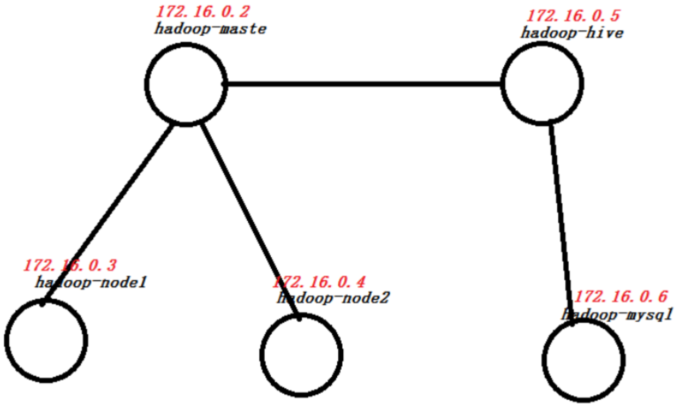
3.集群网络规划及子网配置

既然是做集群，网络的规划是少不了的,至于网络，可以通过Docker中的DockerNetworking的支持配置。首先设置网络，docker中设置子网可以通过docker network create 方法，这里我们通过命令设置如下的子网。--subnet指定子网络的网段，并为这个子网命名一个名字叫spark docker network create --subnet=172.16.0.0/16 spark 运行完成之后，通过docker network ls 查看创建的子网落列表。



```
[root@bigdata-4 ~]# cd spark
[root@bigdata-4 spark]# docker network create --subnet=172.16.0.0/16 spark
8c8d477d39e4b1b6364f3ca870ec07f69605/cb618be06991a84463ac8fc2109
[root@bigdata-4 spark]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
7caec63a1804        bridge              bridge              local
a3ab55131db4        host                host                local
fceb03794a5c        none                null                local
8c8d477d39e4        spark               bridge              local
[root@bigdata-4 spark]# █
```

接下来就在我们创建的子网落spark中规划集群中每个容器的ip地址。网络ip分配如下：



这个地方要注意，5个容器的hostname都是以hadoop-*开头，这个命名是有讲究的，因为我们要配置容器之间的SSH我密钥登录，在不生成id_rsa.pub公钥的条件下，我们可以通过配置SSH过滤规则来配置容器间的互通信！具体配置会在后面讲到。

4.软件版本： 网络规划好了，接下来我们看看需要用到哪些软件。首先Spark我们使用最新的2.3.0版本，Hadoop采用比较稳定的hadoop-2.7.3版本，Hive采用最新的稳定版本hive-2.3.2，scala采用scala-2.11.8，JDK采用jdk-8u101-linux-x64，mysql使用mysql-5.5.45-linux2.6-x86_64。当然hive和spark要连接mysql数据库少不了一个驱动程序，这个驱动程序我们使用的是mysql-connector-java-5.1.37-bin.jar。

5.SSH无密钥登录规则配置 注意这里不使用ssh-keygen -t rsa -P ''这种方式生成id_rsa.pub，然后集群节点互拷贝id_rsa.pub到authorized_keys文件这种方式，而是通过在.ssh目录下配置ssh_conf文件的方式，ssh_conf中可以配置SSH的通信规则，例如以正则表达式的方式指定hostname为XXX的机器之间实现互联互通，而不进行额外的密钥验证。为了编写这个正则表达式，我们5个节点的hostname都以hadoop-*的方式作为开头，这就是采用这种命名规则的原因。下面来看下ssh_conf配置的内容： Host localhost StrictHostKeyChecking no Host 0.0.0.0 StrictHostKeyChecking no

Host hadoop-* StrictHostKeyChecking no 注意上面的最后一行，Host hadoop-* 指定了它的严格的Host验证StrictHostKeyChecking 为no，这样既可以是这5个hostname以hadoop-*开头的容器之间实现互联互通，而不需要二外的验证。

6.Hadoop配置文件 hadoop的配置文件位于$HADOOP_HOME/etc/hadoop文件夹下，重要的配置文件有core-site.xml、hadoop-env.sh、hdfs-site.xml、mapred-env.sh、mapred-site.xml、yarn-env.sh、yarn-site.xml、master、slaves这九个配置文件。其中core-site.xml用于配置hadoop默认的文件系统的访问路径，访问文件系统的用户及用户组等相关的配置。core-site.xml配置如下：

```
<?xml version="1.0"?>
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://hadoop-maste:9000/</value>
    </property>
  <property>
         <name>hadoop.tmp.dir</name>
         <value>file:/usr/local/hadoop/tmp</value>
    </property>
  <property>
        <name>hadoop.proxyuser.root.hosts</name>
        <value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.root.groups</name>
        <value>*</value>
    </property>
  <property>
        <name>hadoop.proxyuser.oozie.hosts</name>
        <value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.oozie.groups</name>
        <value>*</value>
    </property>
</configuration>
```

hadoop-env.sh这个配置文件用来配置hadoop运行依赖的JDK环境，及一些JVM参数的配置，除了JDK路径的配置外，其他的我们不用管，内容如下：

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# Set Hadoop-specific environment variables here.
# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.
# The java implementation to use.
export JAVA_HOME=/usr/local/jdk1.8.0_101
# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}
# Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
```

```
done
# The maximum amount of heap to use, in MB. Default is 1000.
#export HADOOP_HEAPSIZE=
#export HADOOP_NAMENODE_INIT_HEAPSIZE=""
# Extra Java runtime options.  Empty by default.
export HADOOP_OPTS="$HADOOP_OPTS -Djava.net.preferIPv4Stack=true"
# Command specific options appended to HADOOP_OPTS when specified
export HADOOP_NAMENODE_OPTS="-Dhadoop.security.logger=${HADOOP_SECURITY_LOGGER:-INFO,RFAS} -Dhdfs.audit.logger=${HDFS_AUDIT_LOGGER:-INFO,NullAppender} $HADOOP_NAMENODE_OPTS"
export HADOOP_DATANODE_OPTS="-Dhadoop.security.logger=ERROR,RFAS $HADOOP_DATANODE_OPTS"
export HADOOP_SECONDARYNAMENODE_OPTS="-Dhadoop.security.logger=${HADOOP_SECURITY_LOGGER:-INFO,RFAS} -Dhdfs.audit.logger=${HDFS_AUDIT_LOGGER:-INFO,NullAppender} $HADOOP_SECONDARYNAMENODE_OPTS"
export HADOOP_NFS3_OPTS="$HADOOP_NFS3_OPTS"
export HADOOP_PORTMAP_OPTS="-Xmx512m $HADOOP_PORTMAP_OPTS"
# The following applies to multiple commands (fs, dfs, fsck, distcp etc)
export HADOOP_CLIENT_OPTS="-Xmx512m $HADOOP_CLIENT_OPTS"
#HADOOP_JAVA_PLATFORM_OPTS="-XX:-UsePerfData $HADOOP_JAVA_PLATFORM_OPTS"
# On secure datanodes, user to run the datanode as after dropping privileges.
# This **MUST** be uncommented to enable secure HDFS if using privileged ports
# to provide authentication of data transfer protocol.  This **MUST NOT** be
# defined if SASL is configured for authentication of data transfer protocol
# using non-privileged ports.
export HADOOP_SECURE_DN_USER=${HADOOP_SECURE_DN_USER}
# Where log files are stored.  $HADOOP_HOME/logs by default.
#export HADOOP_LOG_DIR=${HADOOP_LOG_DIR}/$USER
# Where log files are stored in the secure data environment.
export HADOOP_SECURE_DN_LOG_DIR=${HADOOP_LOG_DIR}/${HADOOP_HDFS_USER}
###
# HDFS Mover specific parameters
###
# Specify the JVM options to be used when starting the HDFS Mover.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HADOOP_MOVER_OPTS=""
###
# Advanced Users Only!
###
# The directory where pid files are stored. /tmp by default.
# NOTE: this should be set to a directory that can only be written to by
#       the user that will run the hadoop daemons.  Otherwise there is the
#       potential for a symlink attack.
export HADOOP_PID_DIR=${HADOOP_PID_DIR}
export HADOOP_SECURE_DN_PID_DIR=${HADOOP_PID_DIR}
# A string representing this instance of hadoop. $USER by default.
export HADOOP_IDENT_STRING=$USER
```

看似内容很多，但我们只需要更改一处配置即可，即上面标红的地方，导入JAVA_HOME

接下来的配置文件是hdfs-site.xml这个配置文件。它主要用来配置hdfs分布式文件系统的namenode即datanode数据的存储路径，及数据区块的冗余数。具体配置如下：

```
<?xml version="1.0"?>
<configuration>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/usr/local/hadoop2.7/dfs/name</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/usr/local/hadoop2.7/dfs/data</value>
    </property>
  <property>
        <name>dfs.webhdfs.enabled</name>
        <value>true</value>
    </property>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
    </property>
  <property>
      <name>dfs.permissions.enabled</name>
    <value>false</value>
  </property>
 </configuration>
```

配置也不复杂！不要怕~

mapred-env.sh和mapred-site.xml这两个配置文件是对hadoop中mapreduce计算框架的运行环境参数及网络的配置文件，因为我们不会用到hadoop中的mapreduce，因为它的计算性能不如spark快，spark官网称，spark运算速度是hadoop mapreduce速度的100倍！ 所以大胆放弃他们两个吧~ 为了消除一些强迫症患者心中的疑虑，这里为大家贴出mapred-site.xml的配置

```
<?xml version="1.0"?>
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.address</name>
        <!-- 配置实际的Master主机名和端口-->
        <value>hadoop-maste:10020</value>
    </property>
    <property>
        <name>mapreduce.map.memory.mb</name>
        <value>4096</value>
    </property>
    <property>
        <name>mapreduce.reduce.memory.mb</name>
        <value>8192</value>
    </property>
  <property>
      <name>yarn.app.mapreduce.am.staging-dir</name>
      <value>/stage</value>
    </property>
    <property>
      <name>mapreduce.jobhistory.done-dir</name>
      <value>/mr-history/done</value>
    </property>
    <property>
      <name>mapreduce.jobhistory.intermediate-done-dir</name>
      <value>/mr-history/tmp</value>
    </property>
</configuration>
```

接下来是yarn-env.sh 和 yarn-site.xml两个配置文件，yarn是hadoop中的任务调度系统，从配置文件的名字可以看出，他们分别用于yarn运行环境的配置及网络的配置。yarn-env.sh中会读取JAVA_HOME环境变量，还会设置一些默认的jdk参数，因此通常情况下我们都不用修改yarn-env.sh这个配置文件，对于yarn-site.xml配置文件，我们还是贴出配置.

```
<?xml version="1.0"?>
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>hadoop-maste</value>
    </property>
  <property>
        <name>yarn.resourcemanager.address</name>
        <value>hadoop-maste:8032</value>
    </property>
    <property>
        <name>yarn.resourcemanager.scheduler.address</name>
        <value>hadoop-maste:8030</value>
    </property>
    <property>
        <name>yarn.resourcemanager.resource-tracker.address</name>
        <value>hadoop-maste:8035</value>
    </property>
    <property>
        <name>yarn.resourcemanager.admin.address</name>
        <value>hadoop-maste:8033</value>
    </property>
    <property>
```

```
        <name>yarn.resourcemanager.webapp.address</name>
        <value>hadoop-maste:8088</value>
    </property>
    <property>
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
   </property>
    <property>
       <name>yarn.nodemanager.vmem-pmem-ratio</name>
       <value>5</value>
    </property>
<property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>22528</value>
    <discription>每个节点可用内存,单位MB</discription>
  </property>

  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>4096</value>
    <discription>单个任务可申请最少内存，默认1024MB</discription>
  </property>

  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>16384</value>
    <discription>单个任务可申请最大内存，默认8192MB</discription>
  </property>
</configuration>
```

它的配置也请简单，主要涉及到一些端口及资源的配置。 接下来看下master和slaves两个配置文件，hadoop是一个master-slave结构的分布式系统，指定哪个节点为master节点，哪些节点为slaves节点呢？hadoop的解决方法是通过master和slaves两个配置文件。下看下master配置文件中的内容：

```
hadoop-maste
```

很简单只有一句话，指定master主节点运行在我们上面网络规划的hadoop-maste这个hostname对应的容器中。再看下slaves配置文件：

```
hadoop-node1
hadoop-node2
```

它的内容也不复杂，指定slaves节点分别为hadoop-node1和hadoop-node2，在这两个容器中将会启动hdfs对应的DataNode进程及YARN资源管理系统启动的NodeManager进程。

7.Spark配置文件讲解

主要有masters、slaves、spark-defaults.conf、spark-env.sh。接下来依次讲解 spark也是一个master-slave结构的分布式计算引擎，它也是通过配置文件masters和slaves的方式来指定master节点和slave节点的。注意这里的文件名字是masters，和hadoop的master文件功能类似，但是却多了个's'，意思想必大家已经猜到，可以配置多个master的主机名，接下来看下masters内容：

```
hadoop-maste
```

简单一行，指定spark集群的master节点为hadoop-maste这个hostname对应的容器再来看下slaves文件，内容如下：

```
hadoop-node1
hadoop-node2
```

分别指定hadoop-node1和hadoop-node2两个节点，Spark集群启动后，会在hadoop-maste启动Master进程，在hadoop-node1和hadoop-node2启动Worker进程。

8.Hive配置文件讲解。

Hive是一个支持SQL语句的数据仓库，SparkSQL之前的版本曾经使用过Hive底层的SQL解释器及优化器，因此Spark自然也是支持读写Hive表格的，前提条件是在Spark中使用enableHiveSupport指定，还需要注意的是Hive的配置文件hive-site.xml需要放到$SPARK_HOME/conf目录下，这样Spark在操作Hive的时候才能找到相应的Hive的通信地址。Hive中重要的配置文件就两个。hive-site.xml和hive-env.sh两个配置文件。hive-site.xml文件内容如下：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
        <property>
                <name>hive.metastore.warehouse.dir</name>
                <value>/home/hive/warehouse</value>
        </property>
        <property>
                <name>hive.exec.scratchdir</name>
                <value>/tmp/hive</value>
        </property>
        <property>
                <name>hive.metastore.uris</name>
                <value>thrift://hadoop-hive:9083</value>
                <description>Thrift URI for the remote metastore. Used by metastore client to connect to remote metastore.
</description>
        </property>
        <property>
                <name>hive.server2.transport.mode</name>
                <value>http</value>
        </property>
        <property>
                <name>hive.server2.thrift.http.port</name>
                <value>10001</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionURL</name>
                <value>jdbc:mysql://hadoop-mysql:3306/hive?
createDatabaseIfNotExist=true</value>
hadoop•conf.zip
4.2KB
        </property>
        <property>
                <name>javax.jdo.option.ConnectionDriverName</name>
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionUserName</name>
                <value>root</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionPassword</name>
                <value>root</value>
        </property>
        <property>
                <name>hive.metastore.schema.verification</name>
                <value>false</value>
        </property>
        <property>
               <name>hive.server2.authentication</name>
               <value>NONE</value>
        </property>
</configuration>
```

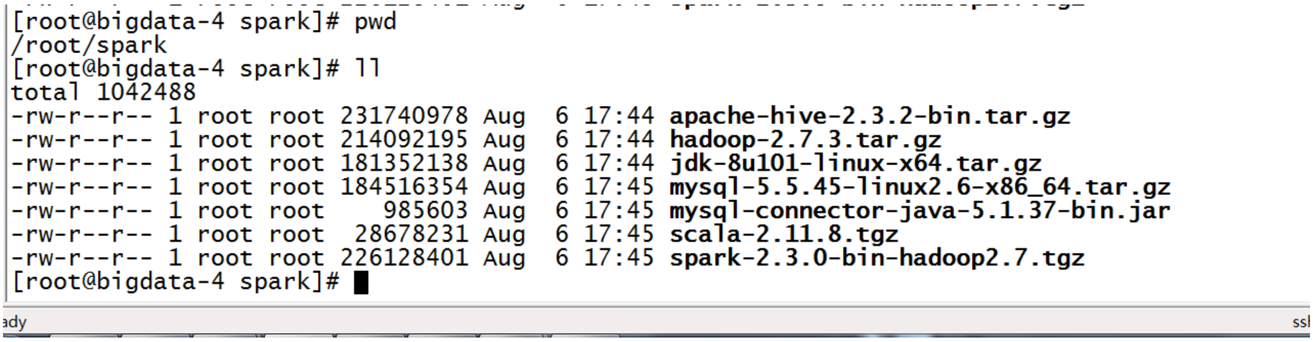
在配置文件中通过javax.jdo.option.ConnectionURL配置选项指定了Hive元数据存放的关系型数据库mysql的存储地址。通过javax.jdo.option.ConnectionDriverName指定驱动，通过hive.metastore.warehouse.dir指定数据仓库在HDFS中的存放位置。hive.metastore.uris指定Hive元数据访问的通信地址，使用的是thrift协议。javax.jdo.option.ConnectionUserName指定连接数据库的用户名，javax.jdo.option.ConnectionPassword指定连接数据库的密码。再来看hive-env.sh配置文件。因为Hive的数据要存储在HDFS中，那Hive怎么和Hadoop通信呢？Hive的解决方案是在hive-env.sh中加入Hadoop的路径，这样Hive就会从Hadoop的路径下去寻找配置文件，自然能找到和Hadoop中HDFS通信的信息，从而完成Hive和Hadoop的通信。其内容如下：

```
HADOOP_HOME=/usr/local/hadoop-2.7.3
```

简简单单的一句话而已。

9.准备软件，编写Dockerfile脚本，制作镜像

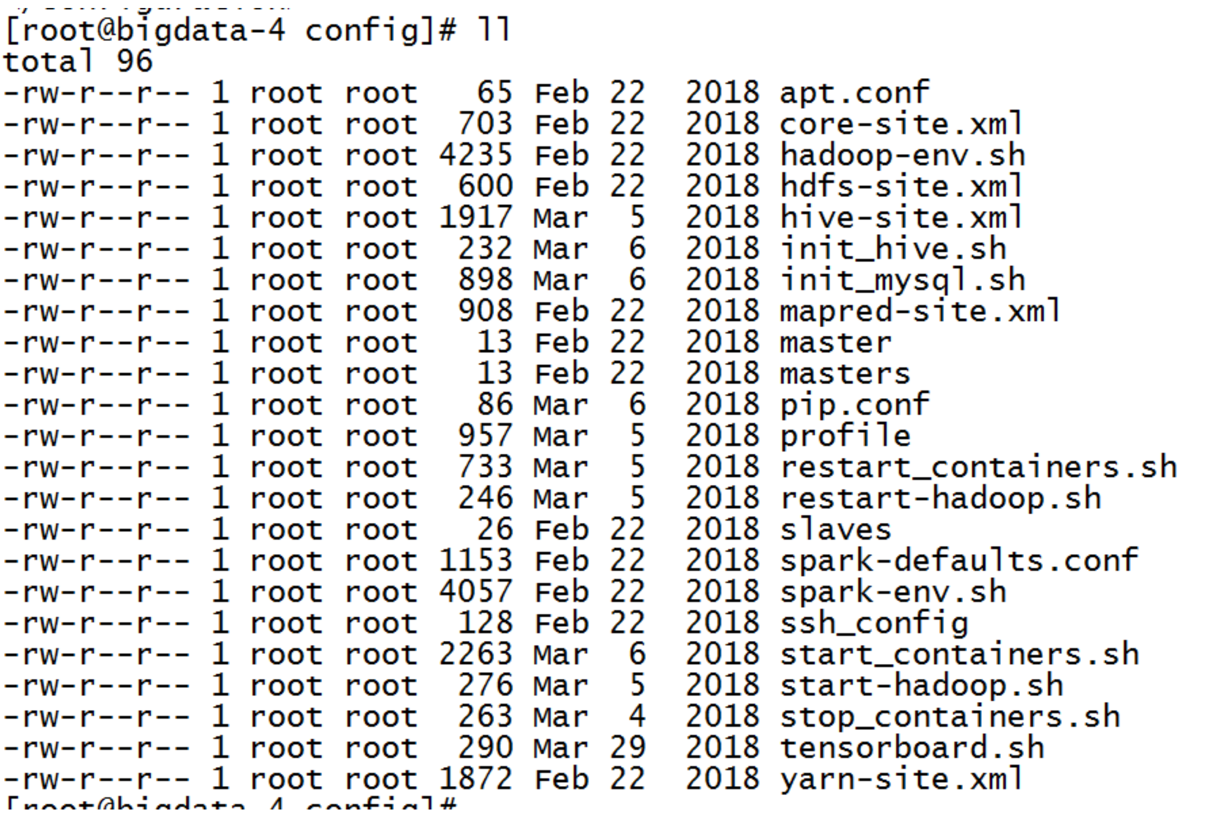
- 制作镜像所需要的软件:



```
[root@bigdata-4 spark]# pwd
/root/spark
[root@bigdata-4 spark]# ll
total 1042488
-rw-r--r-- 1 root root 231740978 Aug  6 17:44 apache-hive-2.3.2-bin.tar.gz
-rw-r--r-- 1 root root 214092195 Aug  6 17:44 hadoop-2.7.3.tar.gz
-rw-r--r-- 1 root root 181352138 Aug  6 17:44 jdk-8u101-linux-x64.tar.gz
-rw-r--r-- 1 root root 184516354 Aug  6 17:45 mysql-5.5.45-linux2.6-x86_64.tar.gz
-rw-r--r-- 1 root root    985603 Aug  6 17:45 mysql-connector-java-5.1.37-bin.jar
-rw-r--r-- 1 root root  28678231 Aug  6 17:45 scala-2.11.8.tgz
-rw-r--r-- 1 root root 226128401 Aug  6 17:45 spark-2.3.0-bin-hadoop2.7.tgz
[root@bigdata-4 spark]# 
```

- 接下来把hadoop,spark,hive的配置文件拷贝到当前目录下的config目录中。【先新建一个config目录】

包括SSH免密钥登录的ssh_config配置文件也拷贝在这里，待会儿使用COPY命令制作镜像。



```
[root@bigdata-4 config]# ll
total 96
-rw-r--r-- 1 root root   65 Feb 22  2018 apt.conf
-rw-r--r-- 1 root root  703 Feb 22  2018 core-site.xml
-rw-r--r-- 1 root root 4235 Feb 22  2018 hadoop-env.sh
-rw-r--r-- 1 root root  600 Feb 22  2018 hdfs-site.xml
-rw-r--r-- 1 root root 1917 Mar  5  2018 hive-site.xml
-rw-r--r-- 1 root root  232 Mar  6  2018 init_hive.sh
-rw-r--r-- 1 root root  898 Mar  6  2018 init_mysql.sh
-rw-r--r-- 1 root root  908 Feb 22  2018 mapred-site.xml
-rw-r--r-- 1 root root   13 Feb 22  2018 master
-rw-r--r-- 1 root root   13 Feb 22  2018 masters
-rw-r--r-- 1 root root   86 Mar  6  2018 pip.conf
-rw-r--r-- 1 root root  957 Mar  5  2018 profile
-rw-r--r-- 1 root root  733 Mar  5  2018 restart_containers.sh
-rw-r--r-- 1 root root  246 Mar  5  2018 restart-hadoop.sh
-rw-r--r-- 1 root root   26 Feb 22  2018 slaves
-rw-r--r-- 1 root root 1153 Feb 22  2018 spark-defaults.conf
-rw-r--r-- 1 root root 4057 Feb 22  2018 spark-env.sh
-rw-r--r-- 1 root root  128 Feb 22  2018 ssh_config
-rw-r--r-- 1 root root 2263 Mar  6  2018 start_containers.sh
-rw-r--r-- 1 root root  276 Mar  5  2018 start-hadoop.sh
-rw-r--r-- 1 root root  263 Mar  4  2018 stop_containers.sh
-rw-r--r-- 1 root root  290 Mar 29  2018 tensorboard.sh
-rw-r--r-- 1 root root 1872 Feb 22  2018 yarn-site.xml
[root@bigdata-4 config]#
```

- **重要！**pip.conf，pip默认会访问国外的源，速度非常慢，在制作镜像的时候会使用到pip下载python包，因此强烈推荐使用pip在国内的源，这个源由豆瓣管理。需要增加如下配置：

上面使用到了pip命令安装python中的一些包，pip默认的源有时候无法访问， 这时需要修改pip默认的源。 编辑配置文件：新建pip.conf文件 添加内容如下： [global] index-url = http://pypi.douban.com/simple trusted-host = pypi.douban.com 这个pip.conf文件是需要放到~/.pip/pip.conf文件中的，因此在制作镜像的时候首先新建~/.pip文件夹， 然后把config目录中已经配置好的pip.conf mv到~/.pip文件夹中，具体操作稍后见Dockerfile中的指令

- **profile** 这个配置文件用于配置系统环境变量的，profile配置文件位于/etc/profile，我们需要把hadoop，spark，hive,jdk,scala，mysql的环境变量配置在这里。其内容如下：

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
if [ "$PS1" ]; then
```

```
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
export JAVA_HOME=/usr/local/jdk1.8.0_101
export SCALA_HOME=/usr/local/scala-2.11.8
export HADOOP_HOME=/usr/local/hadoop-2.7.3
export SPARK_HOME=/usr/local/spark-2.3.0-bin-hadoop2.7
export HIVE_HOME=/usr/local/apache-hive-2.3.2-bin
export MYSQL_HOME=/usr/local/mysql
export PATH=$HIVE_HOME/bin:$MYSQL_HOME/bin:$JAVA_HOME/bin:$SCALA_HOME/bin:$HADOOP_HOME/bin:$SPARK_HOME/bin:$PATH
```

分别将JAVA_HOME、SCALA_HOME、HADOOP_HOME、SPARK_HOME、HIVE_HOME、MYSQL_HOME添加到PATH路径中，在制作镜像的时候，需要把profile文件COPY到/etc/profile。这里只做profile文件的目的是防止Dockerfile中通过ENV命令来设置环境变量不成功！我之前就遇到过使用ENV设置环境变量失败的例子。

- restart_containers.sh,start_containers.sh,stop_containers.sh

这几个脚本是用来启动、重启、停止容器的，也是一键启动、重启、关闭容器集群的脚本，后面会详细讲到的，大家别担心！

- **Dockerfile制作镜像的核心文件** 接下来就讲Dockerfile的编写，在前面Docker课程中，相信大家学到了Dockerfile的基本的命令的使用，那这一小节就来综合使用下吧。先把完整的Dockerfile贴出来：

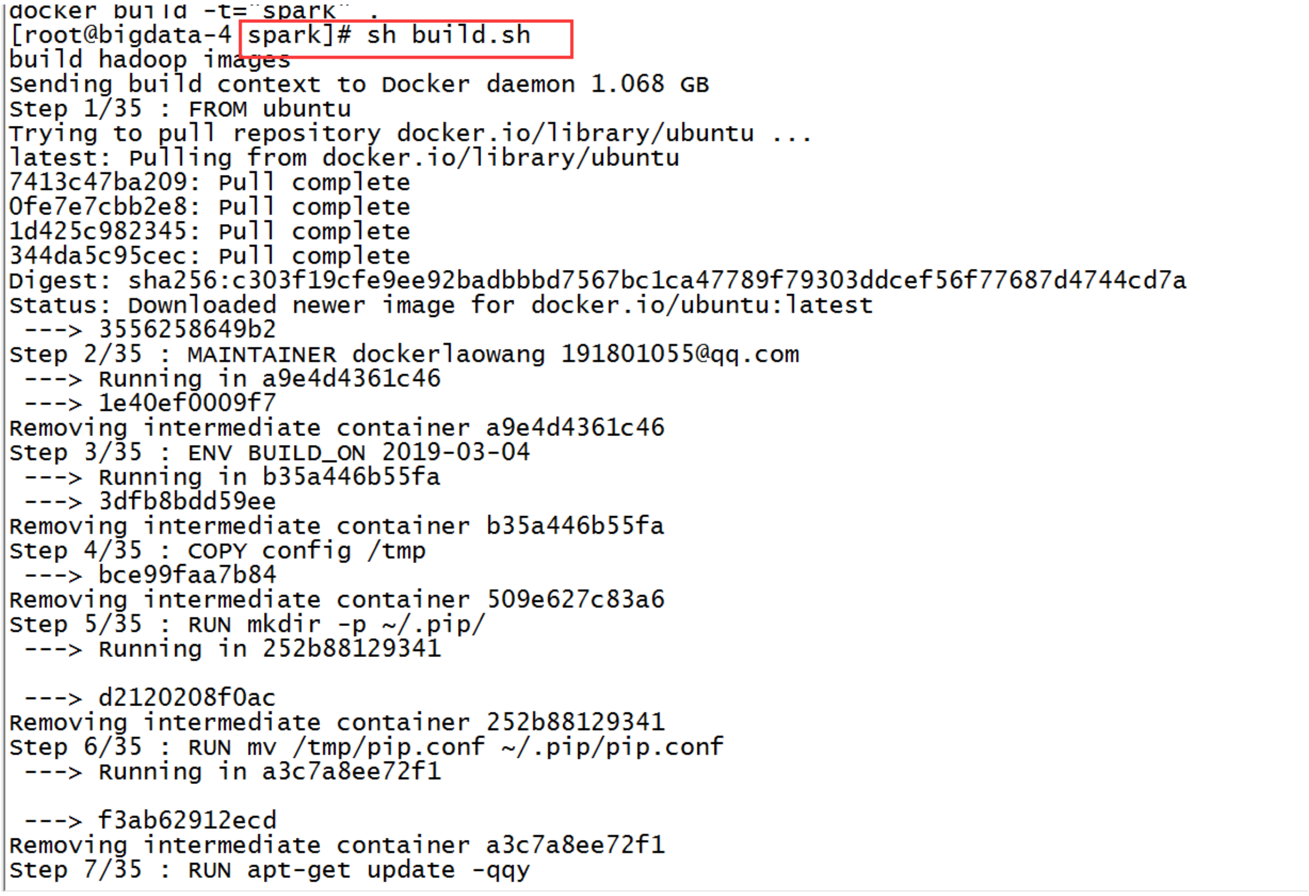
  ```
  FROM ubuntu
  MAINTAINER reganzm 626692024@qq.com
  ENV BUILD_ON 2018-03-04
  COPY config /tmp
  #这句若你的虚拟机能直接连上外网，需要注释掉！因为apt-get需要连上外网下载资源
  #RUN mv /tmp/apt.conf /etc/apt/
  #把pip的豆瓣镜像源配置文件pip.conf移动到~/.pip/pip.conf文件中
  RUN mkdir -p ~/.pip/
  ```

```
RUN mv /tmp/pip.conf ~/.pip/pip.conf
RUN apt-get update -qqy
RUN apt-get -qqy install vim wget net-tools  iputils-ping  openssh-server python3-pip libaio-dev apt-utils
#若你的虚拟机能够连上外网，红色的部分请去除掉！pip不必使用代理上网！
RUN pip3 install pandas  numpy  matplotlib  sklearn  seaborn  scipy tensorflow  gensim
#--proxy http://root:1qazxcde32@192.168.0.4:7890/
#添加JDK
ADD ./jdk-8u101-linux-x64.tar.gz /usr/local/
#添加hadoop
ADD ./hadoop-2.7.3.tar.gz  /usr/local
#添加scala
ADD ./scala-2.11.8.tgz /usr/local
#添加spark
ADD ./spark-2.3.0-bin-hadoop2.7.tgz /usr/local
#添加mysql
ADD ./mysql-5.5.45-linux2.6-x86_64.tar.gz /usr/local
RUN mv /usr/local/mysql-5.5.45-linux2.6-x86_64  /usr/local/mysql
ENV MYSQL_HOME /usr/local/mysql
#添加hive
ADD ./apache-hive-2.3.2-bin.tar.gz /usr/local
ENV HIVE_HOME /usr/local/apache-hive-2.3.2-bin
#写入hive-env.sh文件的内容
RUN echo "HADOOP_HOME=/usr/local/hadoop-2.7.3"  | cat >> /usr/local/apache-hive-2.3.2-bin/conf/hive-env.sh
#添加mysql-connector-java-5.1.37-bin.jar到hive的lib目录中
ADD ./mysql-connector-java-5.1.37-bin.jar /usr/local/apache-hive-2.3.2-bin/lib
#添加mysql-connector-java-5.1.37-bin.jar到spark的jars目录中
RUN cp /usr/local/apache-hive-2.3.2-bin/lib/mysql-connector-java-5.1.37-bin.jar /usr/local/spark-2.3.0-bin-hadoop2.7/jars
#增加JAVA_HOME环境变量
ENV JAVA_HOME /usr/local/jdk1.8.0_101
#hadoop环境变量
ENV HADOOP_HOME /usr/local/hadoop-2.7.3
#scala环境变量
ENV SCALA_HOME /usr/local/scala-2.11.8
#spark环境变量
ENV SPARK_HOME /usr/local/spark-2.3.0-bin-hadoop2.7
#将环境变量添加到系统变量中
ENV PATH $HIVE_HOME/bin:$MYSQL_HOME/bin:$SCALA_HOME/bin:$SPARK_HOME/bin:$HADOOP_HOME/bin:$JAVA_HOME/bin:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$PATH
#生成.ssh目录及id_rsa.pub 、authorized_keys文件。并把文件权限设置为600
RUN ssh-keygen -t rsa -f ~/.ssh/id_rsa -P '' && \
    cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys && \
    chmod 600 ~/.ssh/authorized_keys
#将配置移动到正确的位置
RUN mv /tmp/ssh_config    ~/.ssh/config && \
    echo StrictHostKeyChecking no >>/etc/ssh/ssh_config && \
    mv /tmp/profile /etc/profile && \
    mv /tmp/masters $SPARK_HOME/conf/masters && \
    cp /tmp/slaves $SPARK_HOME/conf/ && \
    mv /tmp/spark-defaults.conf $SPARK_HOME/conf/spark-defaults.conf && \
    mv /tmp/spark-env.sh $SPARK_HOME/conf/spark-env.sh && \
    cp /tmp/hive-site.xml $SPARK_HOME/conf/hive-site.xml && \
    mv /tmp/hive-site.xml $HIVE_HOME/conf/hive-site.xml && \
    mv /tmp/hadoop-env.sh $HADOOP_HOME/etc/hadoop/hadoop-env.sh && \
    mv /tmp/hdfs-site.xml $HADOOP_HOME/etc/hadoop/hdfs-site.xml && \
    mv /tmp/core-site.xml $HADOOP_HOME/etc/hadoop/core-site.xml && \
    mv /tmp/yarn-site.xml $HADOOP_HOME/etc/hadoop/yarn-site.xml && \
    mv /tmp/mapred-site.xml $HADOOP_HOME/etc/hadoop/mapred-site.xml && \
    mv /tmp/master $HADOOP_HOME/etc/hadoop/master && \
    mv /tmp/slaves $HADOOP_HOME/etc/hadoop/slaves && \
    mv /tmp/start-hadoop.sh ~/start-hadoop.sh && \
    mkdir -p /usr/local/hadoop2.7/dfs/data && \
    mkdir -p /usr/local/hadoop2.7/dfs/name && \
    mv /tmp/init_mysql.sh ~/init_mysql.sh && chmod 700 ~/init_mysql.sh && \
    mv /tmp/init_hive.sh ~/init_hive.sh && chmod 700 ~/init_hive.sh && \
    mv /tmp/restart-hadoop.sh ~/restart-hadoop.sh && chmod 700 ~/restart-hadoop.sh
RUN echo $JAVA_HOME
#设置工作目录
WORKDIR /root
#启动sshd服务
RUN /etc/init.d/ssh start
#修改start-hadoop.sh权限为700
RUN chmod 700 start-hadoop.sh
#修改root密码
RUN echo "root:111111" | chpasswd
CMD ["/bin/bash"]
```

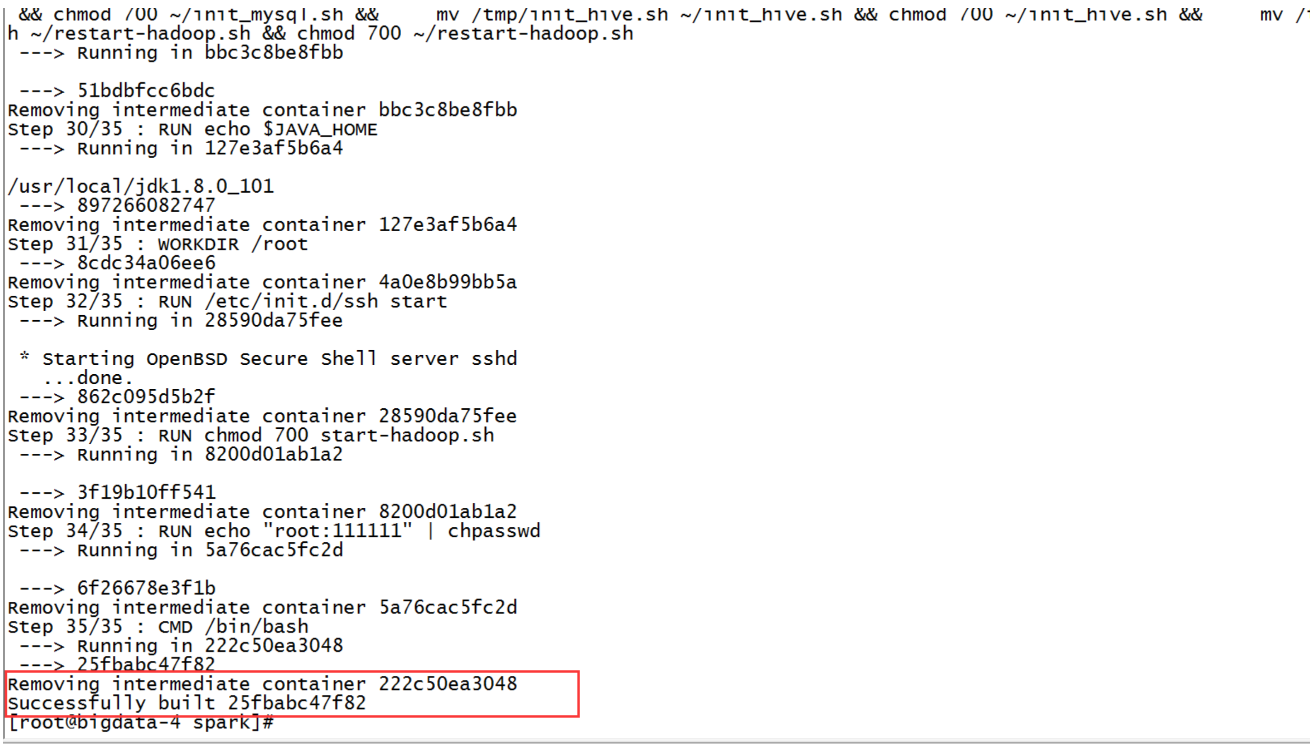
- Dockerfile编写完成，接下来写一个build.sh脚本，内容如下：

```
echo build hadoop images
docker build -t="spark" .
```

表示构建一个名叫spark的镜像，.表示Dockerfile的路径，因为在当前路径下，所有用.,若在其他地方则用绝对路径指定Dockerfile的路径即可。运行sh build.sh，就会开始制作镜像了。

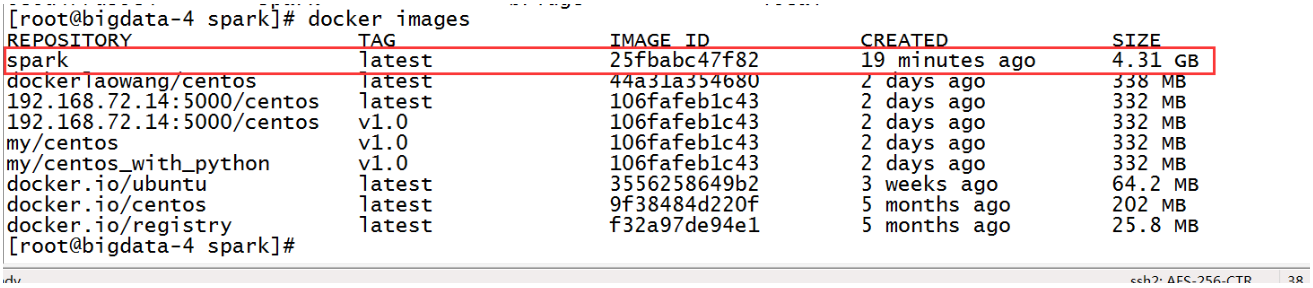


```
docker build -t="spark" .
[root@bigdata-4 spark]# sh build.sh
build hadoop images
Sending build context to Docker daemon 1.068 GB
Step 1/35 : FROM ubuntu
Trying to pull repository docker.io/library/ubuntu ...
latest: Pulling from docker.io/library/ubuntu
7413c47ba209: Pull complete
0fe7e7cbb2e8: Pull complete
1d425c982345: Pull complete
344da5c95cec: Pull complete
Digest: sha256:c303f19cfe9ee92badbbbd7567bc1ca47789f79303ddcef56f77687d4744cd7a
Status: Downloaded newer image for docker.io/ubuntu:latest
 ---> 3556258649b2
Step 2/35 : MAINTAINER dockerlaowang 191801055@qq.com
 ---> Running in a9e4d4361c46
 ---> 1e40ef0009f7
Removing intermediate container a9e4d4361c46
Step 3/35 : ENV BUILD_ON 2019-03-04
 ---> Running in b35a446b55fa
 ---> 3dfb8bdd59ee
Removing intermediate container b35a446b55fa
Step 4/35 : COPY config /tmp
 ---> bce99faa7b84
Removing intermediate container 509e627c83a6
Step 5/35 : RUN mkdir -p ~/.pip/
 ---> Running in 252b88129341

 ---> d2120208f0ac
Removing intermediate container 252b88129341
Step 6/35 : RUN mv /tmp/pip.conf ~/.pip/pip.conf
 ---> Running in a3c7a8ee72f1

 ---> f3ab62912ecd
Removing intermediate container a3c7a8ee72f1
Step 7/35 : RUN apt-get update -qqy
```



```
 && chmod 700 ~/init_mysql.sh &&      mv /tmp/init_hive.sh ~/init_hive.sh && chmod 700 ~/init_hive.sh &&      mv /
h ~/restart-hadoop.sh && chmod 700 ~/restart-hadoop.sh
 ---> Running in bbc3c8be8fbb

 ---> 51bdbfcc6bdc
Removing intermediate container bbc3c8be8fbb
Step 30/35 : RUN echo $JAVA_HOME
 ---> Running in 127e3af5b6a4

/usr/local/jdk1.8.0_101
 ---> 897266082747
Removing intermediate container 127e3af5b6a4
Step 31/35 : WORKDIR /root
 ---> 8cdc34a06ee6
Removing intermediate container 4a0e8b99bb5a
Step 32/35 : RUN /etc/init.d/ssh start
 ---> Running in 28590da75fee

 * Starting OpenBSD Secure Shell server sshd
   ...done.
 ---> 862c095d5b2f
Removing intermediate container 28590da75fee
Step 33/35 : RUN chmod 700 start-hadoop.sh
 ---> Running in 8200d01ab1a2

 ---> 3f19b10ff541
Removing intermediate container 8200d01ab1a2
Step 34/35 : RUN echo "root:111111" | chpasswd
 ---> Running in 5a76cac5fc2d

 ---> 6f26678e3f1b
Removing intermediate container 5a76cac5fc2d
Step 35/35 : CMD /bin/bash
 ---> Running in 222c50ea3048
 ---> 25fbabc47f82
Removing intermediate container 222c50ea3048
Successfully built 25fbabc47f82
[root@bigdata-4 spark]#
```

制作完成后，使用docker images可以看到制作的名为spark的镜像。



```
[root@bigdata-4 spark]# docker images
REPOSITORY                  TAG          IMAGE ID        CREATED           SIZE
spark                       latest       25fbabc47f82    19 minutes ago    4.31 GB
dockerlaowang/centos        latest       44a31a354680    2 days ago        338 MB
192.168.72.14:5000/centos   latest       106fafeb1c43    2 days ago        332 MB
192.168.72.14:5000/centos   v1.0         106fafeb1c43    2 days ago        332 MB
my/centos                   v1.0         106fafeb1c43    2 days ago        332 MB
my/centos_with_python       v1.0         106fafeb1c43    2 days ago        332 MB
docker.io/ubuntu            latest       3556258649b2    3 weeks ago       64.2 MB
docker.io/centos            latest       9f38484d220f    5 months ago      202 MB
docker.io/registry          latest       f32a97de94e1    5 months ago      25.8 MB
[root@bigdata-4 spark]#
```

10.启动容器 start_containers.sh

使用这个镜像可完成容器的启动，容器的启动需要注意什么呢？因为我们使用了基于DockerNetworking的网络机制，因此可以在启动容器的时候为容器在子网172.16.0.0/16 spark中分贝172.16.0.1 172.16.0.255以外的IP地址，容器内部容器的通信是基于hostname，因此需要指定hostname，为了方便容器的管理，需要为启动的每个容器指定一个名字。为了方便外网访问，需要通过-p命令指定容器到宿主机的端口映射。还要为每个容器增加host列表。接下来我们看看怎样启动容器，启动容器的脚本位于start_containers.sh脚本中，内容如下：

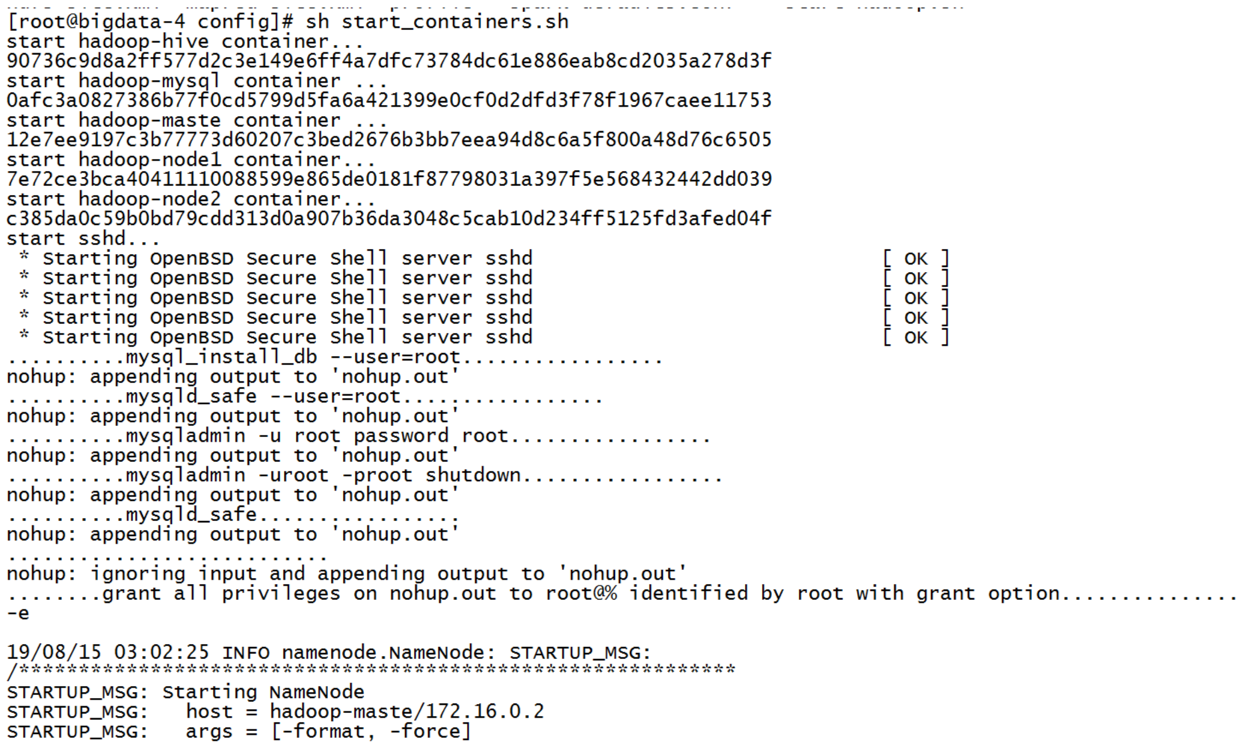
```
echo start hadoop-hive container...
docker run -itd --restart=always --net spark --ip 172.16.0.5 --privileged --name hive -
-hostname hadoop-hive --add-host hadoop-node1:172.16.0.3 \
--add-host hadoop-node2:172.16.0.4 --add-host hadoop-mysql:172.16.0.6 --add-host
hadoop-maste:172.16.0.2 spark /bin/bash
echo start hadoop-mysql container ...
docker run -itd --restart=always --net spark --ip 172.16.0.6 --privileged --name mysql
--hostname hadoop-mysql --add-host hadoop-node1:172.16.0.3 --add-host hadoop-
node2:172.16.0.4 --add-host hadoop-hive:172.16.0.5 --add-host hadoop-maste:172.16.0.2
spark /bin/bash
echo start hadoop-maste container ...
docker run -itd --restart=always \--net spark \--ip 172.16.0.2 \--privileged \-p
18032:8032 \-p 28080:18080 \-p 29888:19888 \-p 17077:7077 \-p 51070:50070 \-p
18888:8888 \-p 19000:9000 \-p 11100:11000 \-p 51030:50030 \-p 18050:8050 \-p 18081:8081
\-p 18900:8900 \--name hadoop-maste \
--hostname hadoop-maste  \--add-host hadoop-node1:172.16.0.3 \--add-host hadoop-
node2:172.16.0.4 --add-host hadoop-hive:172.16.0.5 --add-host hadoop-mysql:172.16.0.6
spark /bin/bash
echo "start hadoop-node1 container..."
docker run -itd --restart=always \--net spark  \--ip 172.16.0.3 \--privileged \-p
18042:8042 \-p 51010:50010 \-p 51020:50020 \--name hadoop-node1 \--hostname hadoop-
node1  --add-host hadoop-hive:172.16.0.5 --add-host hadoop-mysql:172.16.0.6 \--add-host
hadoop-maste:172.16.0.2 \--add-host hadoop-node2:172.16.0.4 spark  /bin/bash
echo "start hadoop-node2 container..."
docker run -itd --restart=always \--net spark  \--ip 172.16.0.4 \--privileged \-p
18043:8042 \-p 51011:50011 \-p 51021:50021 \--name hadoop-node2 \--hostname hadoop-
node2 --add-host hadoop-maste:172.16.0.2 \--add-host hadoop-node1:172.16.0.3 --add-host
hadoop-mysql:172.16.0.6 --add-host hadoop-hive:172.16.0.5 spark /bin/bash
echo start sshd...
##
docker exec -it hadoop-maste /etc/init.d/ssh start
docker exec -it hadoop-node1 /etc/init.d/ssh start
docker exec -it hadoop-node2 /etc/init.d/ssh start
docker exec -it hive  /etc/init.d/ssh start
docker exec -it mysql  /etc/init.d/ssh start
docker exec -it mysql sh ~/init_mysql.sh
docker exec -it hadoop-maste sh ~/start-hadoop.sh
docker exec -it hive  sh ~/init_hive.sh
##
echo finished
docker ps
#-------------------------------------------------------------------------------------
#使用docker run命令运行镜像  --net指定子网络  --ip指定从子网落中分配一个ip地址  -p指定端口映射 --
#name指定容器名称，--hostname指定容器的hostname，--add-host指定hostname和ip的映射，这个映射将被
添加到/etc/hosts文件中。最后要使用 docker exec 执行脚本启动每个容器中的ssh服务。就是上面注释的部
分。
```

start-hadoop.sh内容如下：用于启动hadoop,spark集群

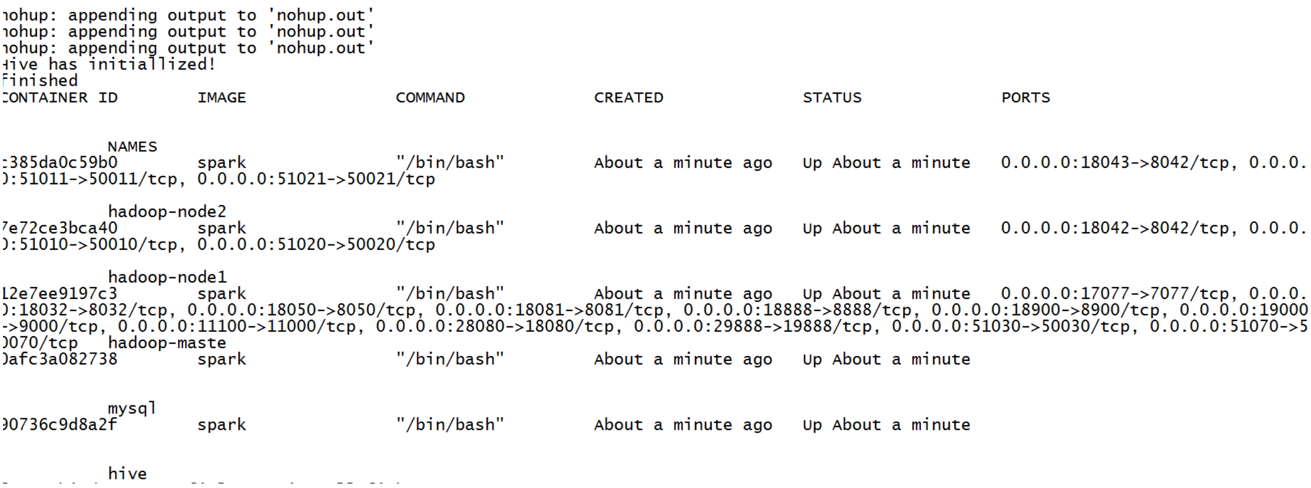
```
#!/bin/bash
echo -e "\n"
hdfs namenode -format
echo -e "\n"
$HADOOP_HOME/sbin/start-dfs.sh
echo -e "\n"
$HADOOP_HOME/sbin/start-yarn.sh
echo -e "\n"
$SPARK_HOME/sbin/start-all.sh
echo -e "\n"
hdfs dfs -mkdir /mr-history
hdfs dfs -mkdir /stage
echo -e "\n"
```

- 启动容器：sh start_containers.sh

  ```
  sh start_containers.sh
  ```



```
[root@bigdata-4 config]# sh start_containers.sh
start hadoop-hive container...
90736c9d8a2ff577d2c3e149e6ff4a7dfc73784dc61e886eab8cd2035a278d3f
start hadoop-mysql container ...
0afc3a0827386b77f0cd5799d5fa6a421399e0cf0d2dfd3f78f1967caee11753
start hadoop-maste container ...
12e7ee9197c3b77773d60207c3bed2676b3bb7eea94d8c6a5f800a48d76c6505
start hadoop-node1 container...
7e72ce3bca40411110088599e865de0181f87798031a397f5e568432442dd039
start hadoop-node2 container...
c385da0c59b0bd79cdd313d0a907b36da3048c5cab10d234ff5125fd3afed04f
start sshd...
 * Starting OpenBSD Secure Shell server sshd                            [ OK ]
 * Starting OpenBSD Secure Shell server sshd                            [ OK ]
 * Starting OpenBSD Secure Shell server sshd                            [ OK ]
 * Starting OpenBSD Secure Shell server sshd                            [ OK ]
 * Starting OpenBSD Secure Shell server sshd                            [ OK ]
..........mysql_install_db --user=root.................
nohup: appending output to 'nohup.out'
..........mysqld_safe --user=root.................
nohup: appending output to 'nohup.out'
..........mysqladmin -u root password root.................
nohup: appending output to 'nohup.out'
..........mysqladmin -uroot -proot shutdown.................
nohup: appending output to 'nohup.out'
..........mysqld_safe.................
nohup: appending output to 'nohup.out'
...........................
nohup: ignoring input and appending output to 'nohup.out'
........grant all privileges on nohup.out to root@% identified by root with grant option...............
-e

19/08/15 03:02:25 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = hadoop-maste/172.16.0.2
STARTUP_MSG:   args = [-format, -force]
```

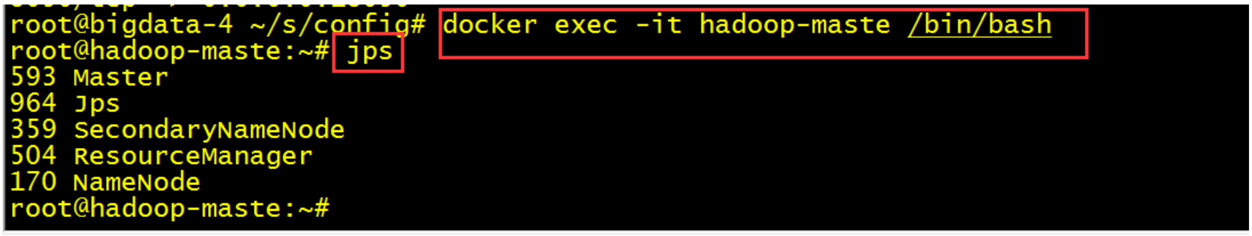


```
ıohup: appending output to 'nohup.out'
ıohup: appending output to 'nohup.out'
ıohup: appending output to 'nohup.out'
Hive has initiallized!
Finished
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS


         NAMES
:385da0c59b0        spark               "/bin/bash"         About a minute ago   Up About a minute   0.0.0.0:18043->8042/tcp, 0.0.0.
):51011->50011/tcp, 0.0.0.0:51021->50021/tcp

         hadoop-node2
7e72ce3bca40        spark               "/bin/bash"         About a minute ago   Up About a minute   0.0.0.0:18042->8042/tcp, 0.0.0.
):51010->50010/tcp, 0.0.0.0:51020->50020/tcp

         hadoop-node1
12e7ee9197c3        spark               "/bin/bash"         About a minute ago   Up About a minute   0.0.0.0:17077->7077/tcp, 0.0.0.
):18032->8032/tcp, 0.0.0.0:18050->8050/tcp, 0.0.0.0:18081->8081/tcp, 0.0.0.0:18888->8888/tcp, 0.0.0.0:18900->8900/tcp, 0.0.0.0:19000
->9000/tcp, 0.0.0.0:11100->11000/tcp, 0.0.0.0:28080->18080/tcp, 0.0.0.0:29888->19888/tcp, 0.0.0.0:51030->50030/tcp, 0.0.0.0:51070->5
)070/tcp    hadoop-maste
)afc3a082738        spark               "/bin/bash"         About a minute ago   Up About a minute


         mysql
90736c9d8a2f        spark               "/bin/bash"         About a minute ago   Up About a minute


         hive
```

- 进入hadoop-maste容器，运行start-hadoop.sh脚本，启动hadoop和spark集群

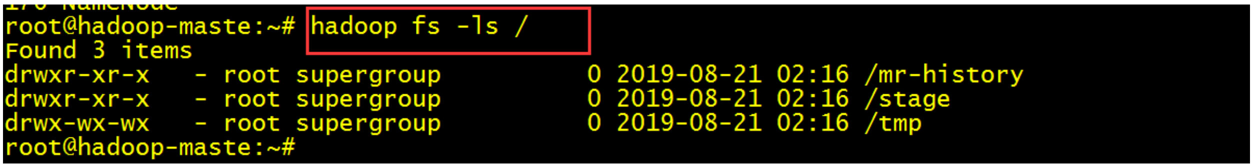
```
docker exec -it hadoop-maste /bin/bash
```

查看进程启动情况

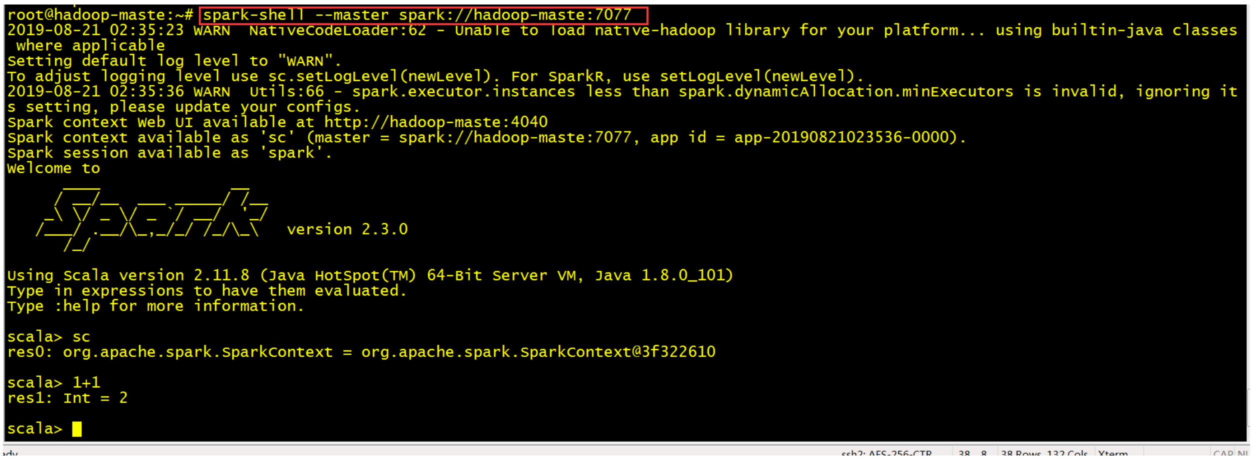


```
root@bigdata-4 ~/s/config# docker exec -it hadoop-maste /bin/bash
root@hadoop-maste:~# jps
593 Master
964 Jps
359 SecondaryNameNode
504 ResourceManager
170 NameNode
root@hadoop-maste:~#
```

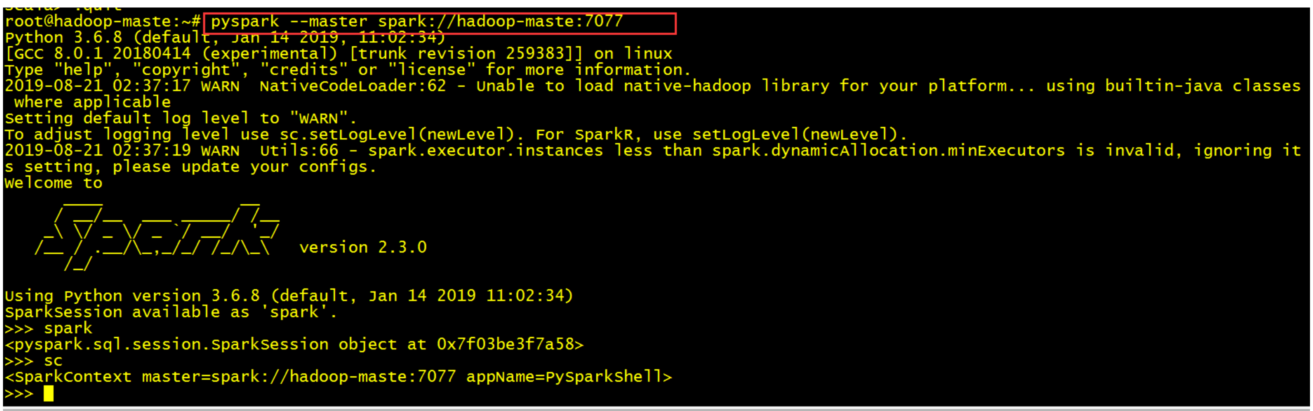
查看hadoop shell



```
root@hadoop-maste:~# hadoop fs -ls /
Found 3 items
drwxr-xr-x   - root supergroup          0 2019-08-21 02:16 /mr-history
drwxr-xr-x   - root supergroup          0 2019-08-21 02:16 /stage
drwx-wx-wx   - root supergroup          0 2019-08-21 02:16 /tmp
root@hadoop-maste:~#
```



```
root@hadoop-maste:~# spark-shell --master spark://hadoop-maste:7077
2019-08-21 02:35:23 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
 where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2019-08-21 02:35:36 WARN  Utils:66 - spark.executor.instances less than spark.dynamicAllocation.minExecutors is invalid, ignoring it
s setting, please update your configs.
Spark context Web UI available at http://hadoop-maste:4040
Spark context available as 'sc' (master = spark://hadoop-maste:7077, app id = app-20190821023536-0000).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_101)
Type in expressions to have them evaluated.
Type :help for more information.

scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@3f322610

scala> 1+1
res1: Int = 2

scala> 
```



```
scala> :quit
root@hadoop-maste:~# pyspark --master spark://hadoop-maste:7077
Python 3.6.8 (default, Jan 14 2019, 11:02:34)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux
Type "help", "copyright", "credits" or "license" for more information.
2019-08-21 02:37:17 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
 where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
2019-08-21 02:37:19 WARN  Utils:66 - spark.executor.instances less than spark.dynamicAllocation.minExecutors is invalid, ignoring it
s setting, please update your configs.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 3.6.8 (default, Jan 14 2019 11:02:34)
SparkSession available as 'spark'.
>>> spark
<pyspark.sql.session.SparkSession object at 0x7f03be3f7a58>
>>> sc
<SparkContext master=spark://hadoop-maste:7077 appName=PySparkShell>
>>> 
```

- 关闭容器

```
#sh stop_containers.sh
#内容-------------------------------------------
docker stop hadoop-maste
docker stop hadoop-node1
docker stop hadoop-node2
docker stop hive
docker stop mysql
echo stop containers
docker rm hadoop-maste
docker rm hadoop-node1
docker rm hadoop-node2
docker rm hive
docker rm mysql
echo rm containers
docker ps
```

- 重启容器

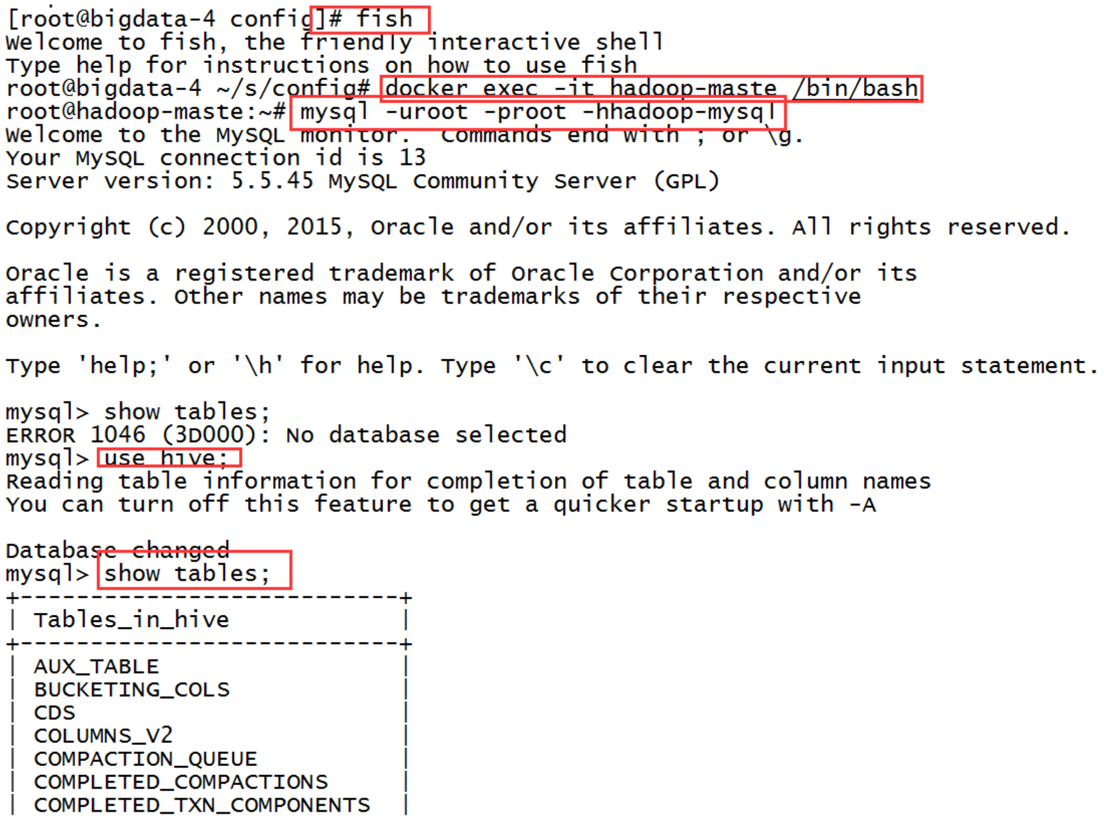
```
#sh restart_containers.sh
#内容-------------------------------------------
echo stop containers
docker stop hadoop-maste
docker stop hadoop-node1
docker stop hadoop-node2
docker stop hadoop-mysql
docker stop hadoop-hive
echo restart containers
docker start hadoop-maste
docker start hadoop-node1
docker start hadoop-node2
docker start hadoop-mysql
docker start hadoop-hive
echo start sshd
docker exec -it hadoop-maste /etc/init.d/ssh start
docker exec -it hadoop-node1 /etc/init.d/ssh start
docker exec -it hadoop-node2 /etc/init.d/ssh start
docker exec -it hadoop-mysql /etc/init.d/ssh start
docker exec -it hadoop-hive /etc/init.d/ssh start
echo start mysql 、 hadoop、spark、hive
docker exec -it mysql sh ~/init_mysql.sh
docker exec -it hadoop-maste ~/restart-hadoop.sh
docker exec -it hive  sh ~/init_hive.sh
echo  containers started
docker ps

#重启要记得重启ssh服务！
#start-hadoop.sh内容如下：用于启动hadoop,spark集群
#!/bin/bash
echo -e "\n"
hdfs namenode -format
echo -e "\n"
$HADOOP_HOME/sbin/start-dfs.sh
echo -e "\n"
$HADOOP_HOME/sbin/start-yarn.sh
echo -e "\n"
$SPARK_HOME/sbin/start-all.sh
echo -e "\n"
hdfs dfs -mkdir /mr-history
hdfs dfs -mkdir /stage
echo -e "\n"
```

- 测试mysql

```
#init_mysql.sh配置文件内容如下：
#!/bin/bash
cd /usr/local/mysql/
echo ..........mysql_install_db --user=root.................
nohup ./scripts/mysql_install_db --user=root &
sleep 3
echo ..........mysqld_safe --user=root.................
nohup ./bin/mysqld_safe --user=root &
sleep 3
echo ..........mysqladmin -u root password 'root'.................
nohup ./bin/mysqladmin -u root password 'root' &
sleep 3
echo ..........mysqladmin -uroot -proot shutdown.................
nohup ./bin/mysqladmin -uroot -proot shutdown &
```

```
sleep 3
echo ..........mysqld_safe.................
nohup ./bin/mysqld_safe --user=root &
sleep 3
echo ...........................
nohup ./bin/mysql -uroot -proot -
e "grant all privileges on *.* to root@'%' identified by 'root' with grant option;"
sleep 3
echo ........grant all privileges on *.* to root@'%' identified by 'root' with gran
t option...............
```

这个文件被上传到了~/init_mysql.sh文件，并且在启动容器的时候使用命令 docker exec -it mysql sh ~/init_mysql.sh运行了这个文件，因此容器启动后mysql服务就被启动和初始化了。 启动容器后，我们登录到hadoop-maste，尝试连接下mysql数据库 docker exec -it hadoop-maste /bin/bash 命令连接到hadoop-maste容器 进入容器后运行mysql -uroot -proot -hhadoop-mysql连接到mysql



```
[root@bigdata-4 config]# fish
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
root@bigdata-4 ~/s/config# docker exec -it hadoop-maste /bin/bash
root@hadoop-maste:~# mysql -uroot -proot -hhadoop-mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.5.45 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> use hive;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_hive            |
+---------------------------+
| AUX_TABLE                 |
| BUCKETING_COLS            |
| CDS                       |
| COLUMNS_V2                |
| COMPACTION_QUEUE          |
| COMPLETED_COMPACTIONS     |
| COMPLETED_TXN_COMPONENTS  |
```

- 配置hive

  初始化Hive，需要把Hive的元数据保存到mysql数据库中，还要启动元数据服务器及Thriftserver服务。这一系列的启动操作及初始化操作，我们也把它保存到init_hive.sh文件中，在启动容器的时候通过docker exec -it hive sh ~/.init_hive.sh执行这个shell文件把初始化hive的操作放到init_hive.sh的外部配置文件中。脚本内容如下：

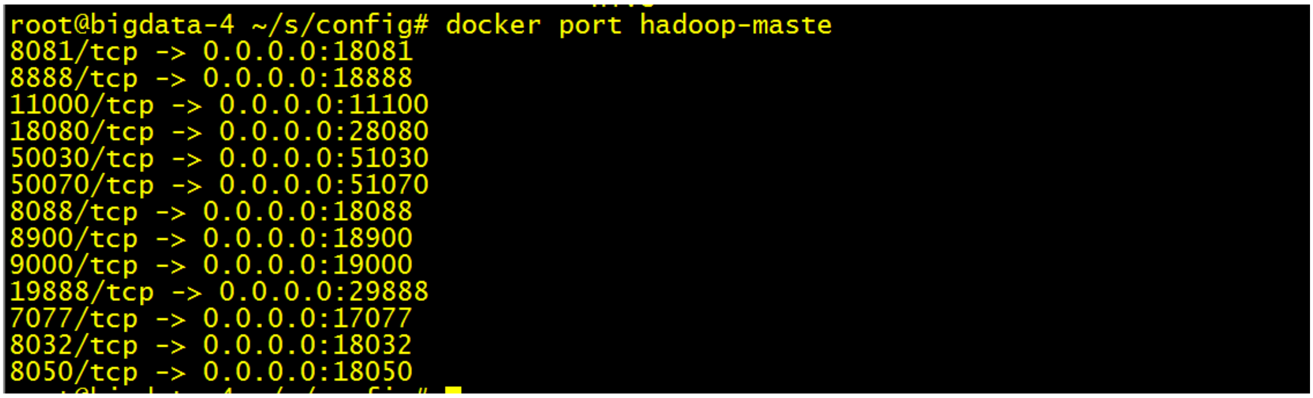
```
#!/bin/bash
cd /usr/local/apache-hive-2.3.2-bin/bin
sleep 3
nohup ./schematool -initSchema -dbType mysql &
sleep 3
nohup ./hive --service metastore  &
sleep 3
nohup ./hive --service hiveserver2 &
sleep 5
echo Hive has initiallized!
```

运行start_container.sh之后，进入到hadoop-maste节点，使用beeline工具连接到hive进行检验。

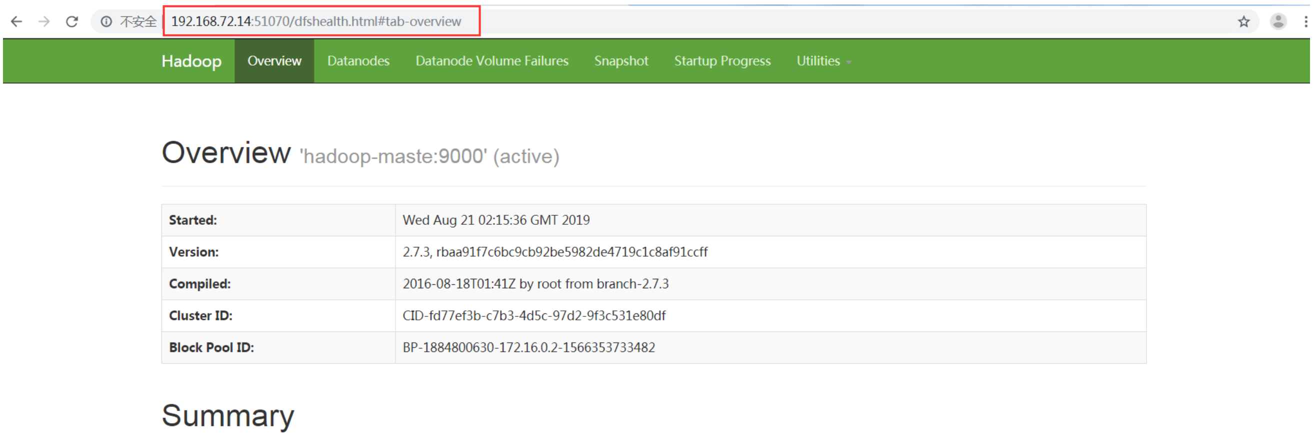
```
  beeline -u "jdbc:hive2://hadoop-hive:10001/default;transportMode=http;httpPath=cliservice"  --color=true -n root
```

正确的连接到了hive中，执行sql语句没有问题，hive配置成功！ 参数说明： beeline -u "jdbc:hive2://hadoop-hive:10001/default;transportMode=http;httpPath=cliservice" --color=true -n root -e"yoursql;" -u指定连接的参数及模式 --color=true指定有颜色显示 -n指定用户名称 -e引号中指定要执行的sql语句，注意语句后面要加分好结尾，sql才能执行！

```
#查看容器和虚拟机端口映射
docker port hadoop-maste
```



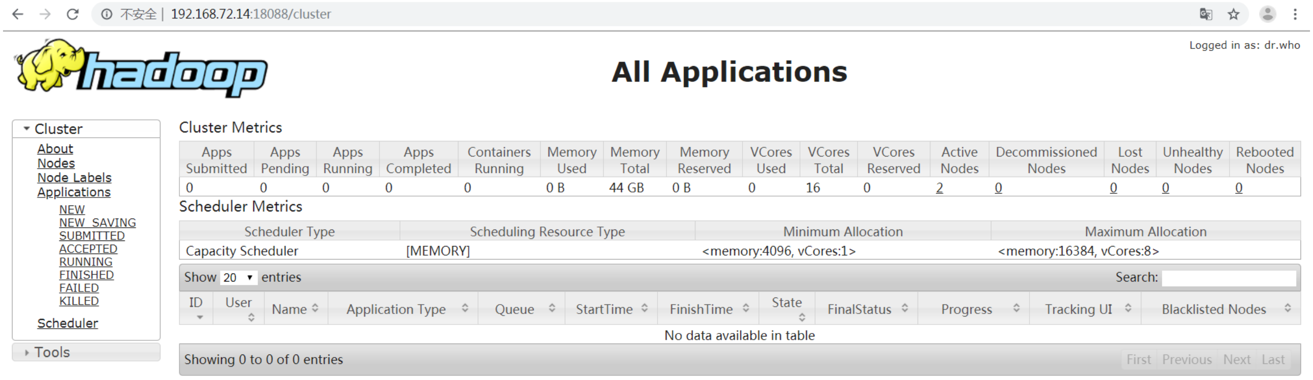
```
root@bigdata-4 ~/s/config# docker port hadoop-maste
8081/tcp -> 0.0.0.0:18081
8888/tcp -> 0.0.0.0:18888
11000/tcp -> 0.0.0.0:11100
18080/tcp -> 0.0.0.0:28080
50030/tcp -> 0.0.0.0:51030
50070/tcp -> 0.0.0.0:51070
8088/tcp -> 0.0.0.0:18088
8900/tcp -> 0.0.0.0:18900
9000/tcp -> 0.0.0.0:19000
19888/tcp -> 0.0.0.0:29888
7077/tcp -> 0.0.0.0:17077
8032/tcp -> 0.0.0.0:18032
8050/tcp -> 0.0.0.0:18050
```



Overview 'hadoop-maste:9000' (active)

Started:	Wed Aug 21 02:15:36 GMT 2019
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-fd77ef3b-c7b3-4d5c-97d2-9f3c531e80df
Block Pool ID:	BP-1884800630-172.16.0.2-1566353733482

Summary



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	44 GB	0 B	0	16	0	2	0	0	0	0

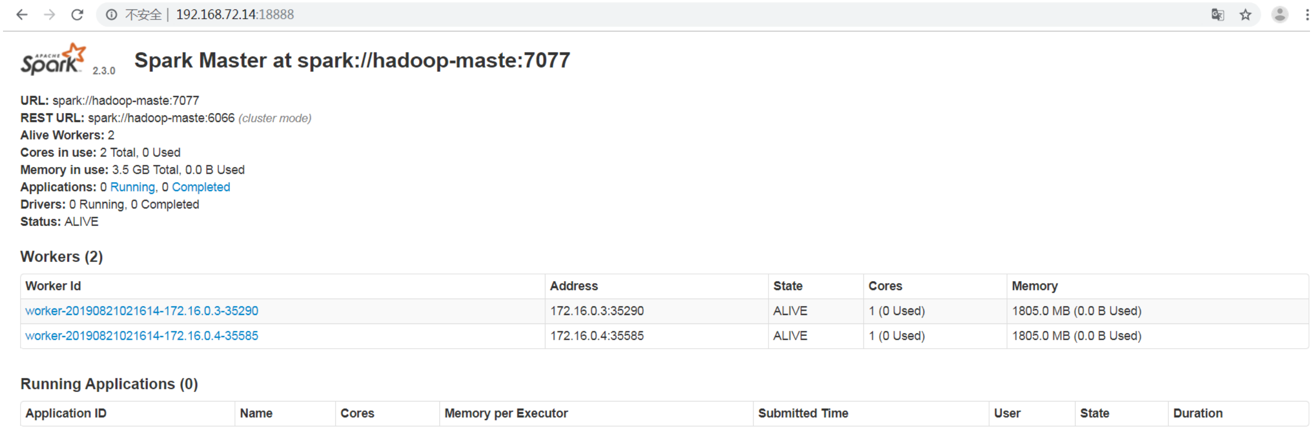
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:4096, vCores:1>	<memory:16384, vCores:8>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
No data available in table											

Showing 0 to 0 of 0 entries



Spark Master at spark://hadoop-maste:7077

URL: spark://hadoop-maste:7077
REST URL: spark://hadoop-maste:6066 *(cluster mode)*
Alive Workers: 2
Cores in use: 2 Total, 0 Used
Memory in use: 3.5 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20190821021614-172.16.0.3-35290	172.16.0.3:35290	ALIVE	1 (0 Used)	1805.0 MB (0.0 B Used)
worker-20190821021614-172.16.0.4-35585	172.16.0.4:35585	ALIVE	1 (0 Used)	1805.0 MB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration