Titan数据运营系统部署流程手册

**二○二○ 年 七 月**

文档修改历史记录

|  |  |  |
| --- | --- | --- |
| 版本状态 | 修改日期 | 备注 |
| v 1.0 | 2020/07/25 | Titan运营系统骨架和基础设施搭建 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[第1章 部署环境 3](#_Toc46584222)

[1.1 系统配置 3](#_Toc46584223)

[1.2 系统依赖配置 4](#_Toc46584224)

[1.2.1 JDK配置 4](#_Toc46584225)

[1.2.2 防火墙配置 5](#_Toc46584226)

[1.2.3 主机名配置 5](#_Toc46584227)

[1.2.4 免密ssh配置 6](#_Toc46584228)

[1.3 依赖组件配置 6](#_Toc46584229)

[1.3.1 Hadoop配置 6](#_Toc46584230)

[1.3.2 Mysql配置 18](#_Toc46584231)

[1.3.3 Hive配置 20](#_Toc46584232)

[1.3.4 Zookeeper配置 27](#_Toc46584233)

[1.3.5 HBase配置 30](#_Toc46584234)

[1.3.6 Kafka配置 34](#_Toc46584235)

[1.3.7 Flulme配置 36](#_Toc46584236)

[1.3.8 Spark配置 37](#_Toc46584237)

[1.3.9 Atlas配置 38](#_Toc46584238)

[1.3.10 Azkaban配置 41](#_Toc46584239)

[第2章 模块安装与配置 49](#_Toc46584240)

[2.1 总体说明 49](#_Toc46584241)

[2.2 数据库数据初始化 49](#_Toc46584242)

[2.3 系统安装部署 49](#_Toc46584243)

[2.3.1 数据提取、存储子系统 49](#_Toc46584244)

[2.3.2分析结果可视化展示子系统 53](#_Toc46584245)

[2.3.3分析、迁移定时任务子系统 54](#_Toc46584246)

[2.4 模块使用 56](#_Toc46584247)

[第3章 其他事项 57](#_Toc46584248)

[3.1 故障排查 57](#_Toc46584249)

[3.1.1 hive metastroe 的问题 57](#_Toc46584250)

[3.1.2hiveserver2 的启动问题 58](#_Toc46584251)

[3.2 Q&A 59](#_Toc46584252)

# 

# 第1章 部署环境

## 1.1 系统配置

操作系统的版本：CentOS Linux release 7.5.1804 (Core)

系统内核：3.10.0-862.el7.x86\_64

内存：32GB

CPU个数：6

CPU核数：1

CPU型号：Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz

## 1.2 系统依赖配置

### 1.2.1 JDK配置

* 安装步骤：
  + 如虚拟机上有自带的OpenJDk必须卸载后安装（通过yum –y remove [包名]的方式卸载 rpm –qa |grep jdk 查询)
  + 本次安装使用的jdk版本为jdk-8u151-linux-x64.tar.gz版本,新建一个jdk的安装目录， /usr/local/下新建了Java目录，命令:mkdir /usr/local/java ，
  + 把 jdk-8u151-linux-x64.tar.gz压缩包通过Xftp放入/usr/local/java文件夹内并通过tar命令解压：tar –vxf {java安装包名}
  + 配置环境变量

1. 在"/etc/profile"文件的尾部添加以下内容：

# set java environment

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_151

export CLASSPATH=.:$JAVA\_HOME/jre/lib/rt.jar:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

export PATH=$PATH:$JAVA\_HOME/bin

1. 通过source /etc/profile 使配置文件立即生效

* 检验安装成功：
  + 输入java –version 判断是否安装成功

成功标志：

(base) [root@master ~]# java -version

java version "1.8.0\_151"

Java(TM) SE Runtime Environment (build 1.8.0\_151-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)

### 1.2.2 防火墙配置

* 关闭防火墙：

systemctl stop firewalld

systemctl disable firewalld

* 成功标志：

输入：firewall-cmd --state

(base) [root@master ~]# firewall-cmd --state

not running

### 1.2.3 主机名配置

* 修改host文件
  + 修改机器的/etc/hosts文件，将每个机器的hostname和ip对应：

(base) [root@master ~]# vim /etc/hosts

172.16.29.121 slave2

172.16.29.117 slave1

172.16.29.109 master

* + 将修改后/etc/hosts文件复制到集群中的每台机器，操作如下：

scp /etc/hosts 172.16.29.117:/etc/

### 1.2.4 免密ssh配置

* 输入命令，ssh-keygen -t rsa，生成key，都不输入密码，一直回车，/root就会生成.ssh文件夹，每台服务器都要设置，
* 合并公钥到authorized\_keys文件，在Master服务器，进入/root/.ssh目录，通过SSH命令合并

cat id\_rsa.pub>> authorized\_keys

ssh 192.168.88.91 cat ~/.ssh/id\_rsa.pub>> authorized\_keys

scp ~/.ssh/authorized\_keys 192.168.88.90:/root

cd /

cat authorized\_keys >> ~/.ssh/authorized\_keys

* 把Master服务器的authorized\_keys、known\_hosts复制到Slave服务器的/root/.ssh目录
* 成功标志：ssh root@192.16.29.117、ssh root@192.16.29.121不需要输入密码

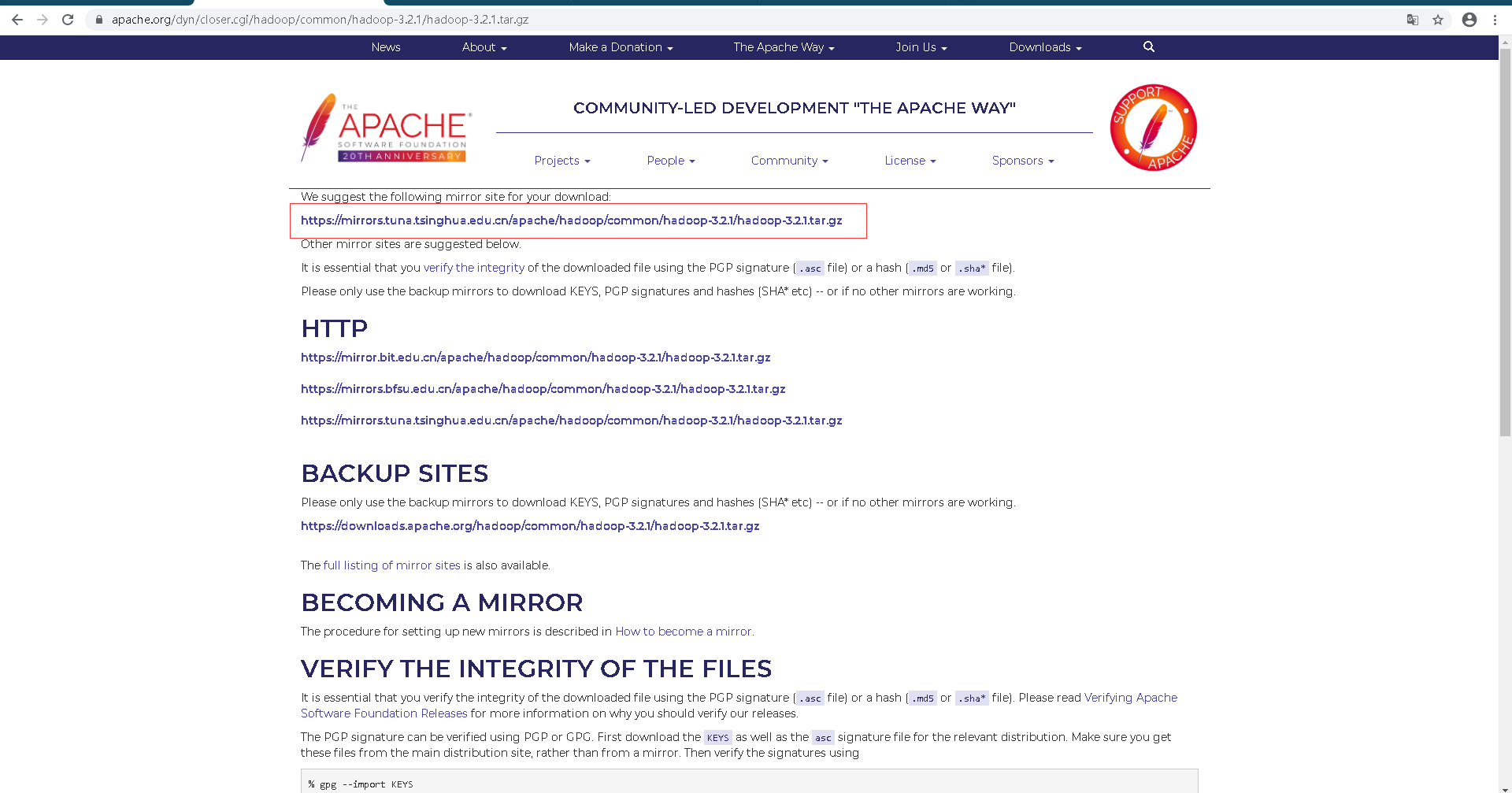
## 1.3 依赖组件配置

### 1.3.1 Hadoop配置

* 安装步骤：

1. 下载安装包：

地址：[**https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz**](https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz)

****

1. 解压安装包：

命令：tar -zxvf [hadoop-3.](user_cancel)2.1.tar.gz

1. 创建文件夹:

在hadoop根目录创建相关存放数据的文件夹：

mkdir tmp、hdfs、hdfs/data、hdfs/name

1. 修改配置文件：
   1. hadoop\_env.sh:

export HADOOP\_PID\_DIR=/root/BigDataBigData/hadoop-3.2.1/pids

export HADOOP\_SECURE\_DN\_PID\_DIR=${HADOOP\_PID\_DIR}

export HADOOP\_IDENT\_STRING=$USER

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_151

* 1. core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://172.16.29.109:9000</value>

</property>

<property>

<name>io.file.buffer.size</name>

<value>131072</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>file:/BigData/hadoop-3.2.1/tmp</value>

<description>Abase for other temporary directories.</description>

</property>

<property>

<name>hadoop.proxyuser.root.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.root.groups</name>

<value>\*</value>

</property>

<property>

<name>fs.trash.interval</name>

<value>1440</value>

</property>

<property>

<name>fs.trash.checkpoint.interval</name>

<value>5</value>

</property>

<property>

<name>hadoop.proxyuser.sqoop2.hosts</name>

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.sqoop2.groups</name>

<value>\*</value>

</property>

<property>

<name>io.compression.codecs</name>

<value>

org.apache.hadoop.io.compress.DefaultCodec,

com.hadoop.compression.lzo.LzoCodec,

com.hadoop.compression.lzo.LzopCodec,

org.apache.hadoop.io.compress.GzipCodec,

org.apache.hadoop.io.compress.BZip2Codec

</value>

</property>

<property>

<name>io.compression.codec.lzo.class</name>

<value>com.hadoop.compression.lzo.LzoCodec</value>

</property>

</configuration>

* 1. hdfs-site.xml

<configuration>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/BigData/hadoop-3.2.1/hdfs/name</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/BigData/hadoop-3.2.1/hdfs/data</value>

</property>

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>172.16.29.117:9001</value>

</property>

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.namenode.acls.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.datanode.max.xcievers</name>

<value>4096</value>

</property>

</configuration>

* 1. yarn-site.xml

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>172.16.29.109:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>172.16.29.109:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>172.16.29.109:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>172.16.29.109:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>172.16.29.109:8088</value>

</property>

<property>

<name>yarn.nodemanager.pmem-check-enabled</name>

<value>false</value>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

<description>Whether virtual memory limits will be enforced for containers</description>

</property>

<property>

<name>yarn.nodemanager.vmem-pmem-ratio</name>

<value>4</value>

<description>Ratio between virtual memory to physical memory when setting memory limits for containers</description>

</property>

<property>

<name>yarn.application.classpath</name>

<value>/BigData/hadoop-3.2.1/etc/hadoop:/BigData/hadoop-3.2.1/share/hadoop/common/lib/\*:/BigData/hadoop-3.2.1/share/hadoop/common/\*:/BigData/hadoop-3.2.1/share/hadoop/hdfs:/BigData/hadoop-3.2.1/share/hadoop/hdfs/lib/\*:/BigData/hadoop-3.2.1/share/hadoop/hdfs/\*:/BigData/hadoop-3.2.1/share/hadoop/mapreduce/lib/\*:/BigData/hadoop-3.2.1/share/hadoop/mapreduce/\*:/BigData/hadoop-3.2.1/share/hadoop/yarn:/BigData/hadoop-3.2.1/share/hadoop/yarn/lib/\*:/BigData/hadoop-3.2.1/share/hadoop/yarn/\*</value>

</property>

</configuration>

* 1. maped-site.xml

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<!-- map输出是否压缩 -->

<!-- 默认值:false -->

<property>

<name>mapreduce.map.output.compress</name>

<value>true</value>

<description>

Should the outputs of the maps be compressed before being

sent across the network. Uses SequenceFile compression.

</description>

</property>

<!-- 设置map输出压缩所使用的对应压缩算法的编解码器,此处设置为LzoCodec,生成的文件后缀为.lzo\_deflate -->

<!-- 默认值:org.apache.hadoop.io.compress.DefaultCodec -->

<property>

<name>mapreduce.map.output.compress.codec</name>

<value>com.hadoop.compression.lzo.LzoCodec</value>

<description>

If the map outputs are compressed, how should they be compressed?

</description>

</property>

<!-- 设置MR job最终输出文件是否压缩 -->

<!-- 默认值:false -->

<property>

<name>mapreduce.output.fileoutputformat.compress</name>

<value>true</value>

<description>Should the job outputs be compressed?

</description>

</property>

<!-- 设置MR job最终输出文件所使用的压缩算法对应的编解码器,此处设置为LzoCodec,生成的文件后缀为.lzo\_deflate -->

<!-- 默认值:org.apache.hadoop.io.compress.DefaultCodec -->

<property>

<name>mapreduce.output.fileoutputformat.compress.codec</name>

<value>com.hadoop.compression.lzo.LzoCodec</value>

<description>If the job outputs are compressed, how should they be compressed?

</description>

</property>

<!-- 设置序列文件的压缩格式 -->

<!-- 默认值:RECORD -->

<property>

<name>mapreduce.output.fileoutputformat.compress.type</name>

<value>BLOCK</value>

<description>If the job outputs are to compressed as SequenceFiles, how should

they be compressed? Should be one of NONE, RECORD or BLOCK.

</description>

</property>

</configuration>

1. 远程复制配置好的hadoop到其它服务器

使用 scp -r 命令

1. 格式化NameNode

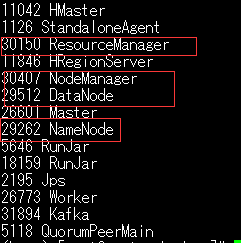
hdfs namenode -format

* 启动步骤：

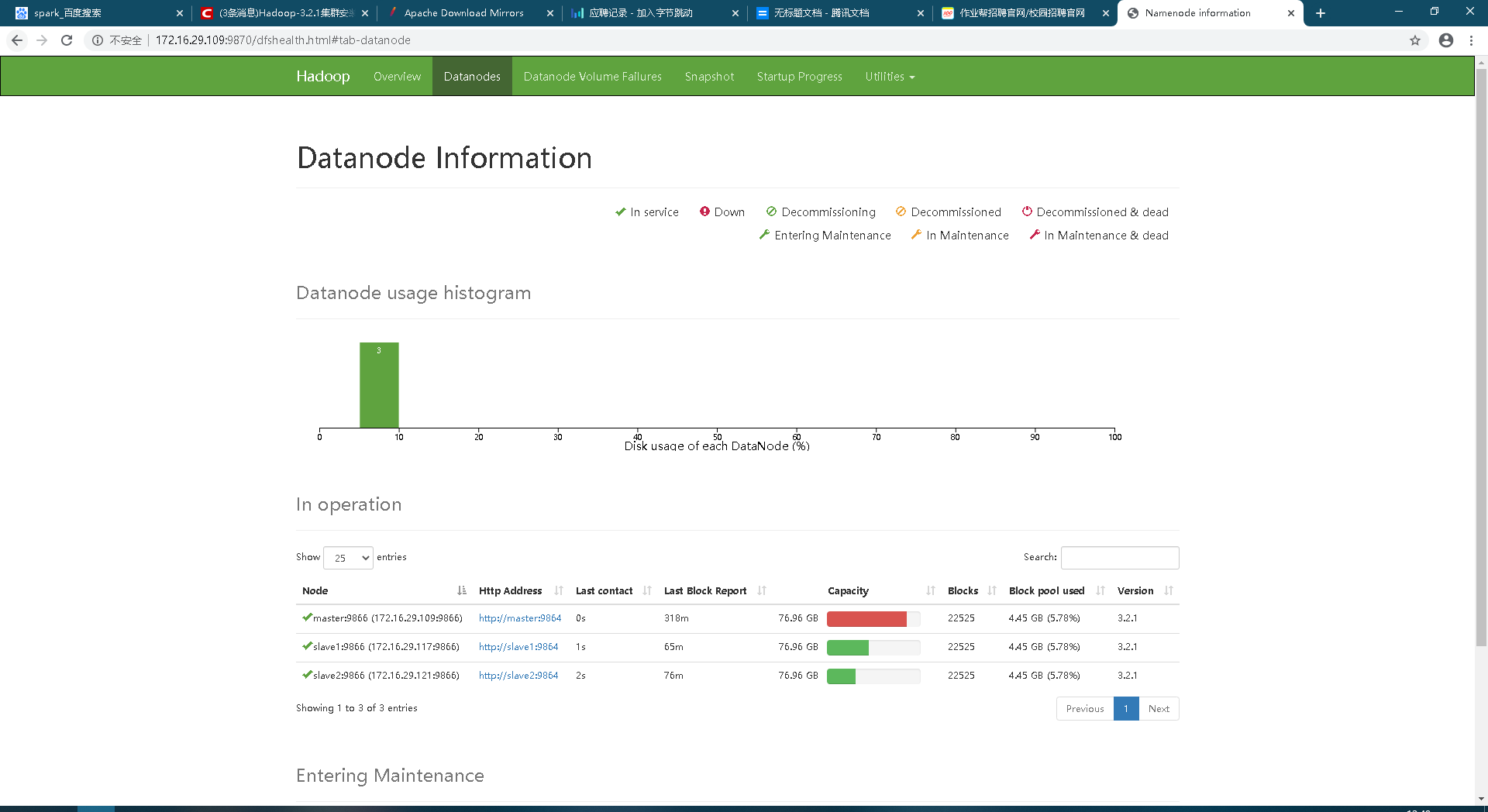
[root@master hadoop]# /BigData/hadoop-3.2.1/sbin/start-all.sh

* 测试安装成功：

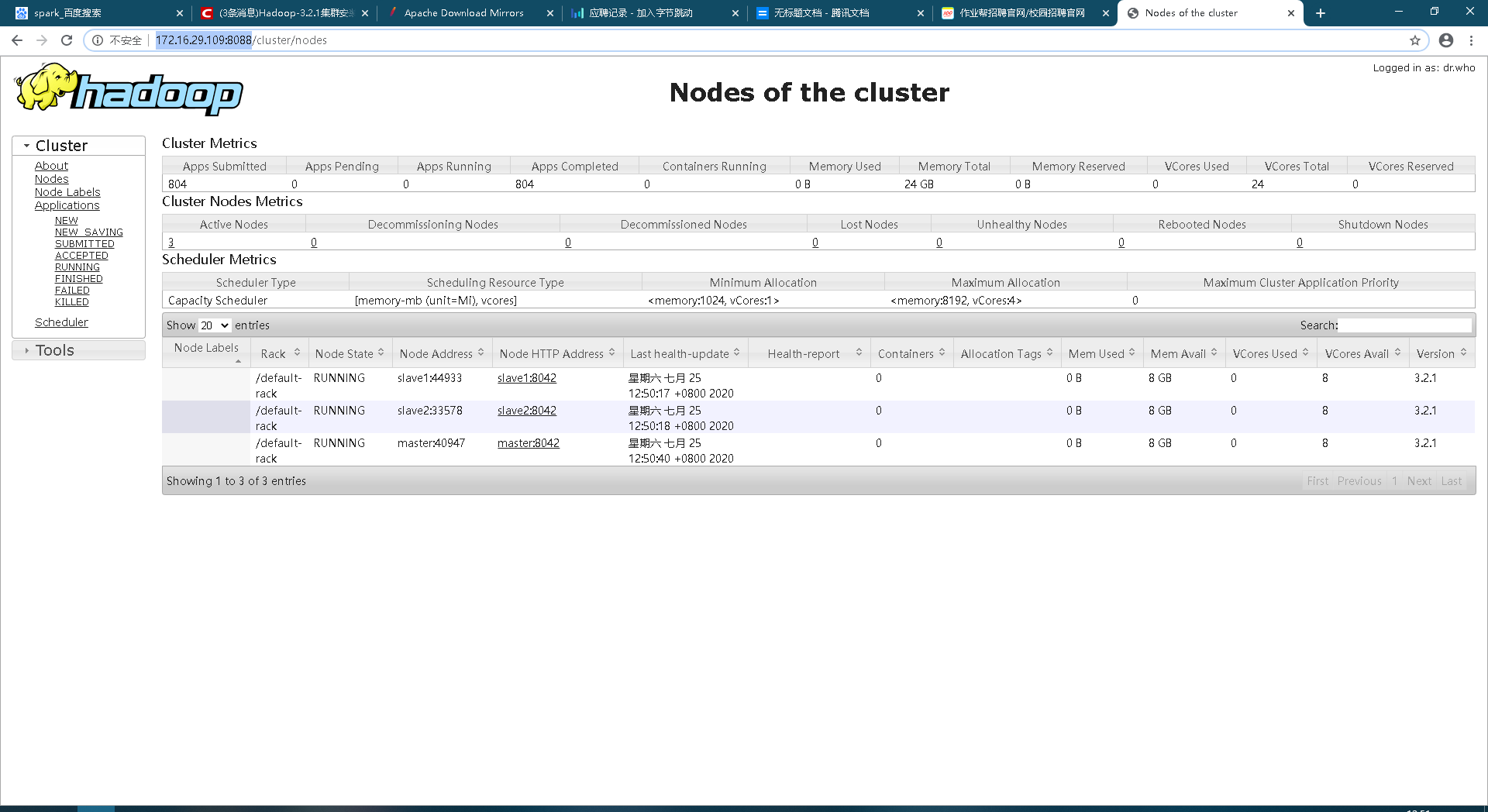
1. 运行jps



1. http://172.16.29.109:9870/



1. http://172.16.29.109:8088/

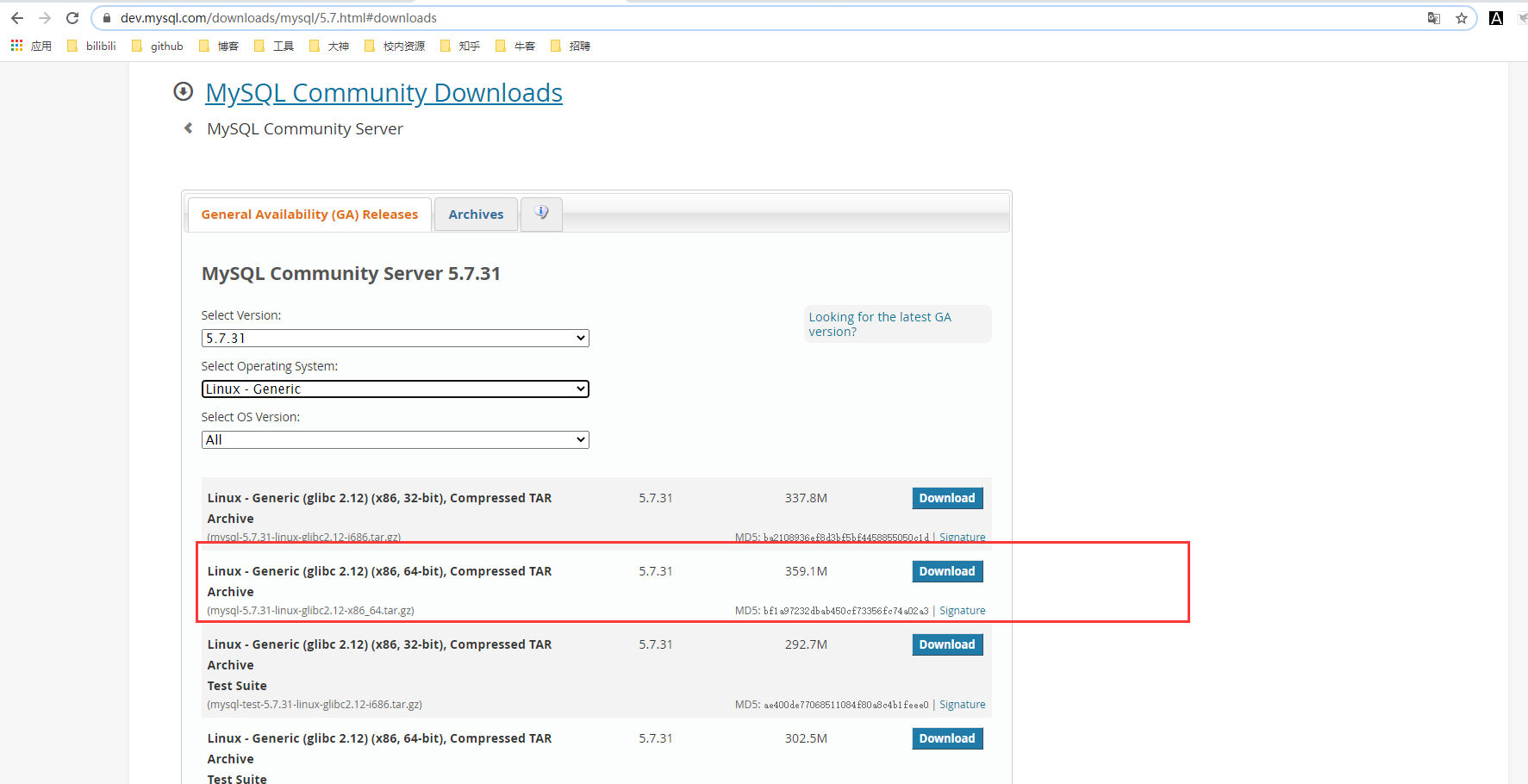
****

### 1.3.2 Mysql配置

* 安装步骤

1. 下载安装包

<https://dev.mysql.com/downloads/mysql/5.7.html#downloads>



1. 解压安装包
2. 修改/etc/my.cnf

[mysqld]

bind-address=0.0.0.0

port=3306

user=mysql

basedir=/usr/local/mysql

datadir=/data/mysql

socket=/tmp/mysql.sock

log-error=/data/mysql/mysql.err

pid-file=/data/mysql/mysql.pid

#character config

character\_set\_server=utf8mb4

symbolic-links=0

explicit\_defaults\_for\_timestamp=true

1. 进入mysql的bin目录

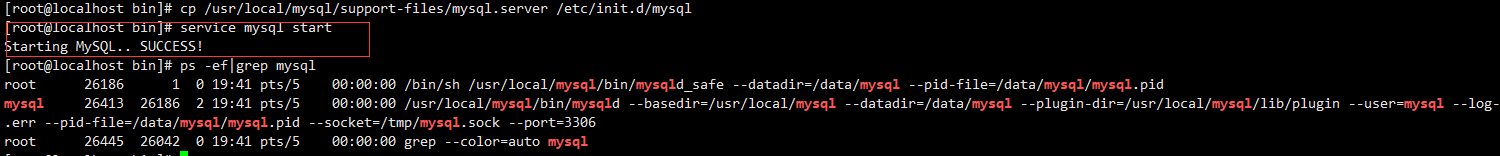
cd /usr/local/mysql/bin/

1. 初始化

./mysqld --defaults-file=/etc/my.cnf --basedir=/usr/local/mysql/ --datadir=/data/mysql/ --user=mysql --initialize

1. 查看密码

cat /data/mysql/mysql.err

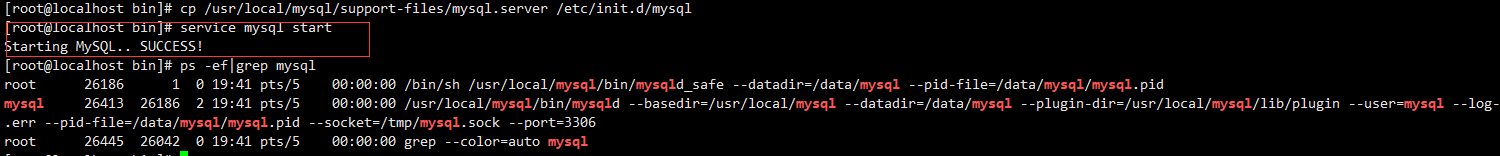


1. 启动mysql，并更改root密码

cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql

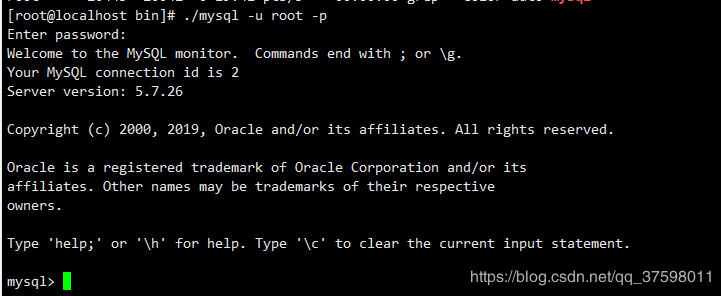
service mysql start

ps -ef|grep mysql



./mysql -u root -p

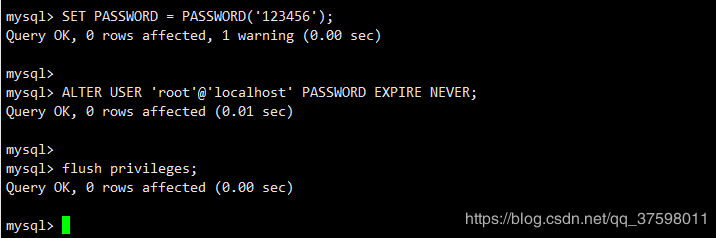
输入上面的密码：



SET PASSWORD = PASSWORD('123456');

ALTER USER 'root'@'localhost' PASSWORD EXPIRE NEVER;

FLUSH PRIVILEGES;

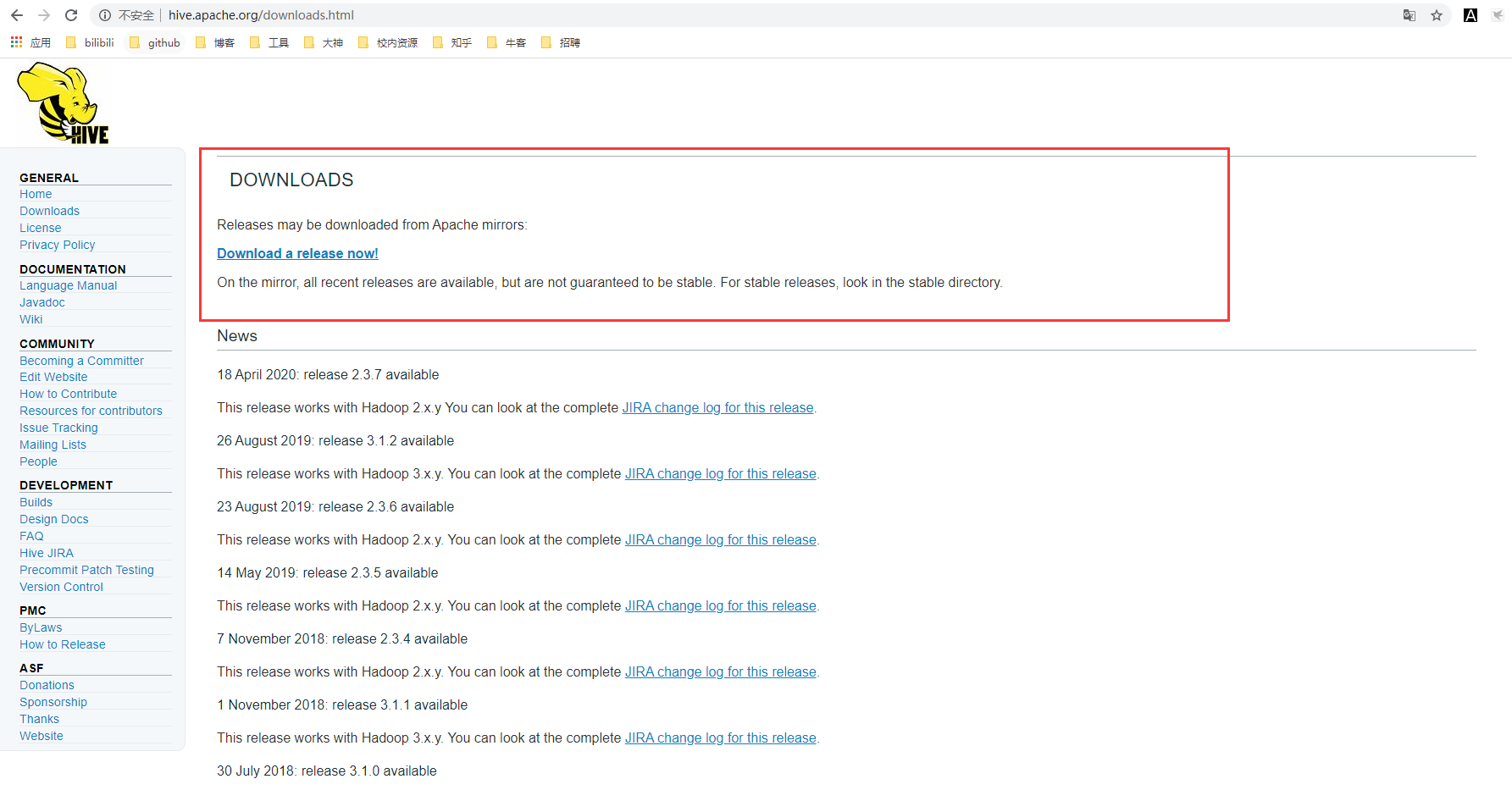


### 1.3.3 Hive配置

* **安装步骤**

1）下载安装包

<http://hive.apache.org/downloads.html>

****

2）解压安装包

3) 将mysql驱动jar包传至/BigData/hive-3.1.2/lib目录下

4) 配置环境变量

export HIVE\_HOME=/BigData/hive-3.1.2

export HIVE\_CONF\_DIR=$HIVE\_HOME/conf

5）修改hive-env.sh

export HADOOP\_HOME=/BigData/hadoop-3.2.1

export HIVE\_CONF\_DIR=/BigData/hive-3.1.2/conf

6) 创建临时目录

[root@slave1 hive-3.1.2]# mkdir -p tmp/resources/

7） 创建hdfs目录

hadoop fs -mkdir -p /tmp/hive

hadoop fs -mkdir -p /apps/hive/warehouse

8） 新建用户组并将hdfs下的/user/hive/目录的权限、所属主与所属组分别赋权

groupadd hadoop

usermod -G hadoop hive

hadoop fs -chown -R hive:hadoop /apps/hive/warehouse

hadoop fs -chown -R hive:hadoop /tmp/hive/

hadoop fs -chmod 755 /apps/hive/warehouse

hadoop fs -chmod 777 /tmp/hive/

9）修改hive-site.xml

<configuration>

<!-- WARNING!!! This file is auto generated for documentation purposes ONLY! -->

<!-- WARNING!!! Any changes you make to this file will be ignored by Hive. -->

<!-- WARNING!!! You must make your changes in hive-site.xml instead. -->

<!-- Hive Execution Parameters -->

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://slave2:3306/hive?createDatabaseIfNotExit=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hive</value>

<description>Username to use against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>hive</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.exec.local.scratchdir</name>

<value>/BigData/hive-3.1.2/tmp</value>

<description>Local scratch space for Hive jobs</description>

</property>

<property>

<name>hive.downloaded.resources.dir</name>

<value>/BigData/hive-3.1.2/tmp/resources</value>

<description>Temporary local directory for added resources in the remote file system.</description>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/apps/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.exec.scratchdir</name>

<value>/tmp/hive</value>

<description>HDFS root scratch dir for Hive jobs which gets created with write all (733) permission.</description>

</property>

<property>

<name>hive.hbase.snapshot.restoredir</name>

<value>/tmp</value>

<description>The directory in which to restore the HBase table snapshot.</description>

</property>

<property>

<name>hive.scratch.dir.permission</name>

<value>700</value>

<description>The permission for the user specific scratch directories that get created.</description>

</property>

<property>

<name>datanucleus.schema.autoCreateAll</name>

<value>true</value>

</property>

<property>

<name>hive.metastore.uris</name>

<value>thrift://master:9083</value>

</property>

<property>

<name>hive.exec.post.hooks</name>

<value>org.apache.atlas.hive.hook.HiveHook</value>

</property>

<!-- 设置hive语句执行输出文件是否开启压缩,具体的压缩算法和压缩格式取决于hadoop中

设置的相关参数 -->

<!-- 默认值:false -->

<property>

<name>hive.exec.compress.output</name>

<value>true</value>

<description>

This controls whether the final outputs of a query (to a local/HDFS file or a Hive table)

is compressed.

The compression codec and other options are determined from Hadoop config variables

mapred.output.compress\*

</description>

</property>

<!-- 控制多个MR Job的中间结果文件是否启用压缩,具体的压缩算法和压缩格式取决于hadoop中

设置的相关参数 -->

<!-- 默认值:false -->

<property>

<name>hive.exec.compress.intermediate</name>

<value>true</value>

<description>

This controls whether intermediate files produced by Hive between multiple map-reduce jobs are compressed.

The compression codec and other options are determined from Hadoop config variables mapred.output.compress\*

</description>

</property>

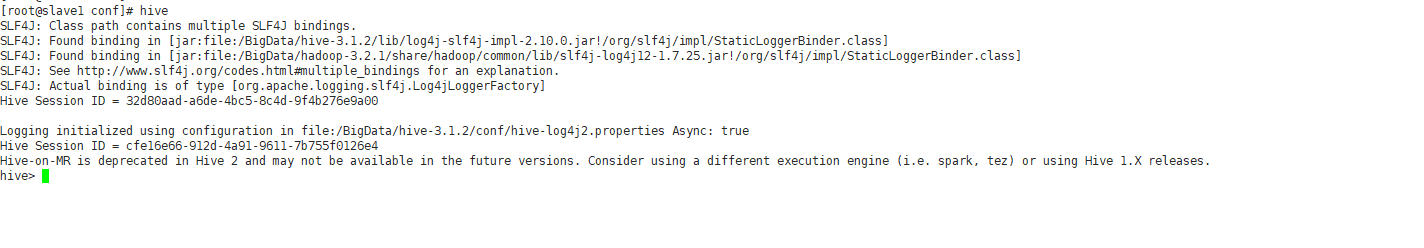
</configuration>

10）初始化元数据库

schematool -dbType mysql -initSchema

* 检测安装成功

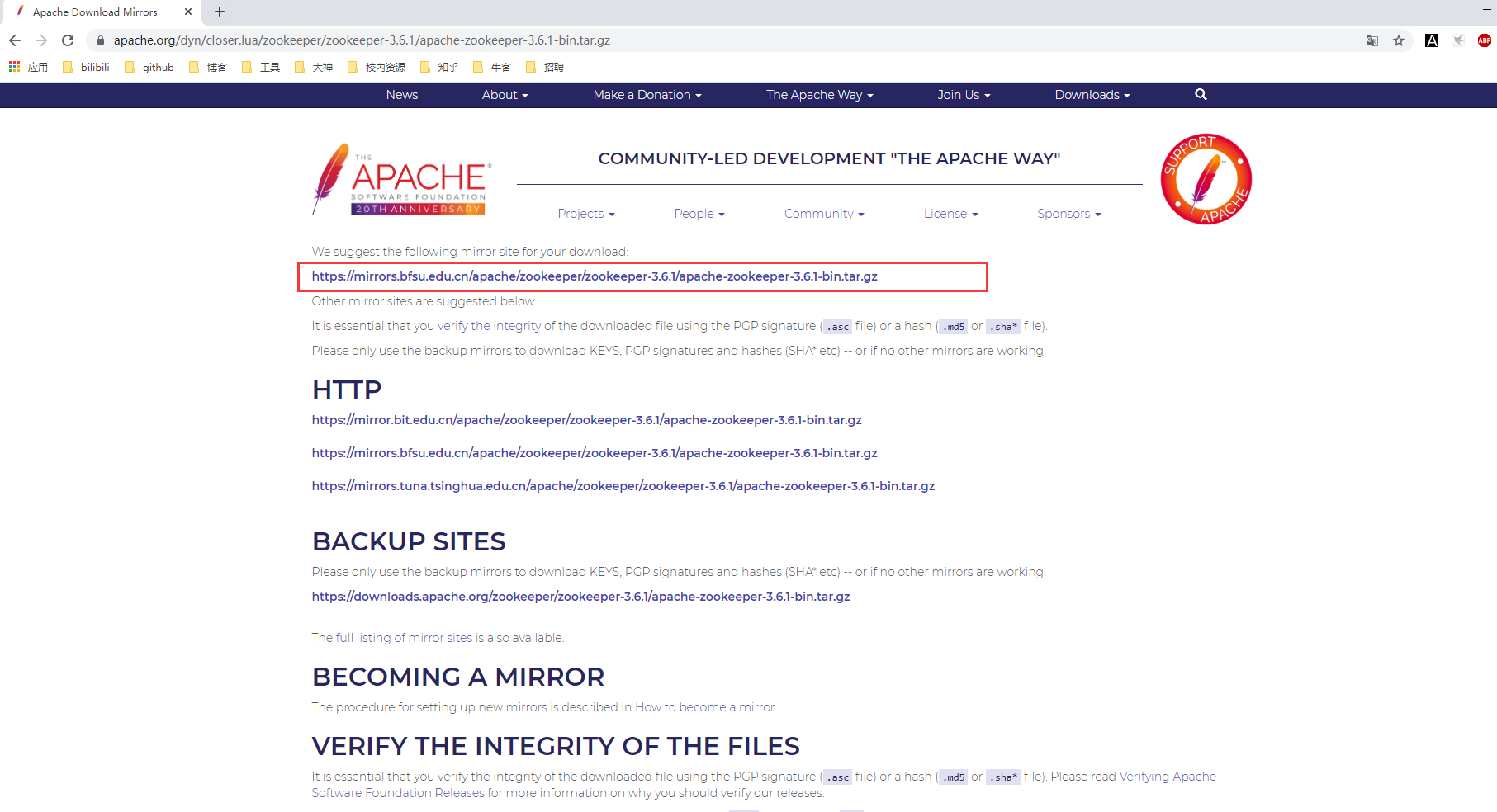
输入hive



### 1.3.4 Zookeeper配置

1. 下载安装包

地址：<https://www.apache.org/dyn/closer.lua/zookeeper/zookeeper-3.6.1/apache-zookeeper-3.6.1-bin.tar.gz>



1. 解压安装包

tar–C /hadoop/–vxf apache-zookeeper-3.5.6-bin.tar

1. 修改配置文件
   1. [root@slave1 conf]# vim zoo.cfg

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/BigData/zookeeper-3.6.1/data

# the port at which the clients will connect

clientPort=2181

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

initLimit=5

syncLimit=2

server.1=master:2888:3888

server.2=slave1:2888:3888

server.3=slave2:2888:3888

1. 创建myid文件

在dataDir目录下创建一个myid文件，然后分别在myid文件中按照zoo.cfg文件的server.A中A的数值，在不同机器上的该文件中填写相应的值

1. 将整个zookeeper-3.5.6scp到其他机器上

* 启动zookeeper

在每台机器上运行bin/zkServer.shstart

查看运行状态:bin/zkServer.shstatus

Mode:leader

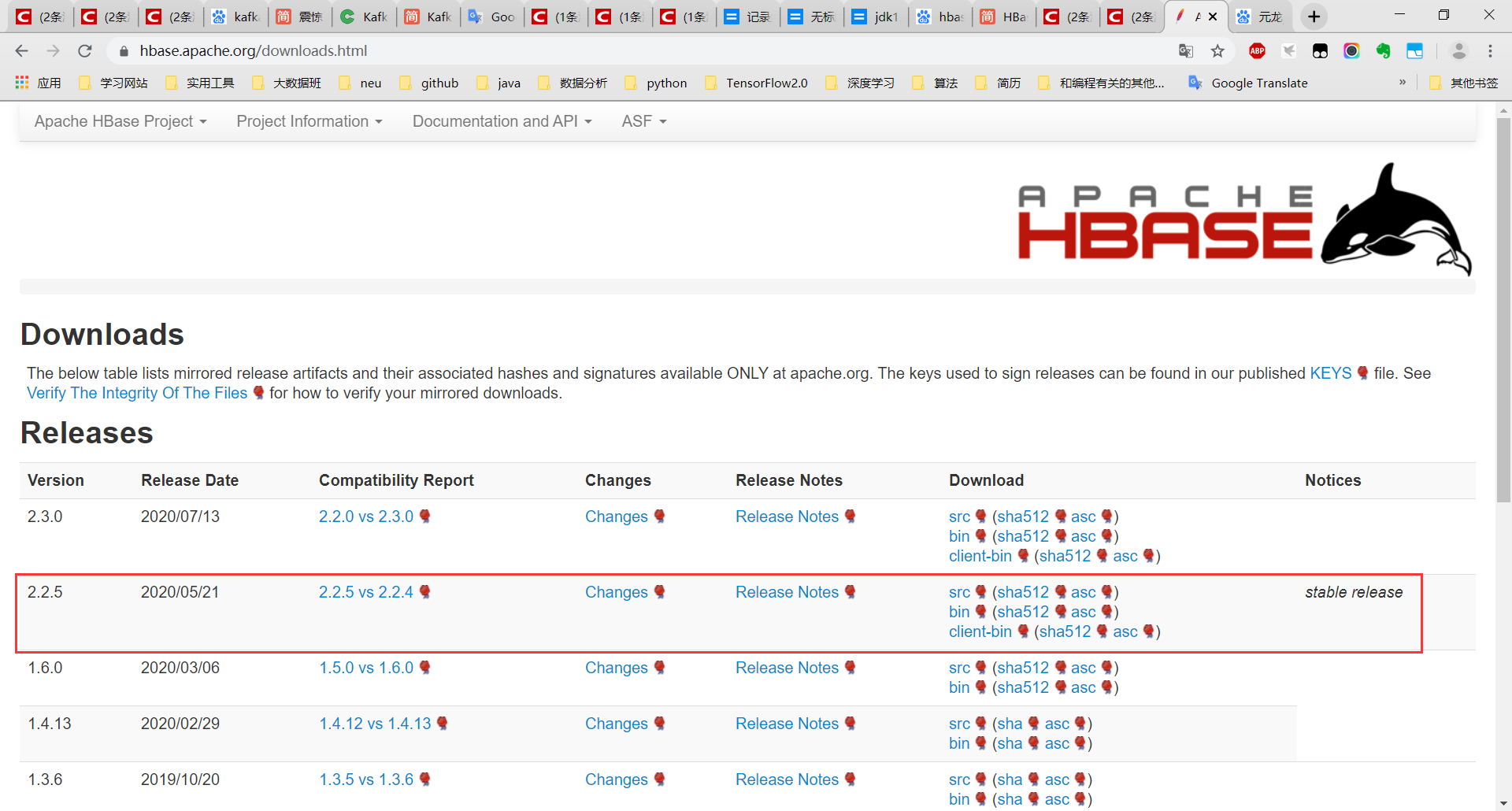
Mode:follower

可以看出哪台为leader了

### 1.3.5 HBase配置

* 安装步骤：

1. 下载



1. 解压

tar –C /hadoop/ –vxf hbase-2.2.5-bin.tar.gz

mv hbase-2.2.5-bin hbase-2.2.5

1. 配置环境变量

vi /etc/profile

#set hbase ervironment

export HBASE\_HOME=/hadoop/hbase-2.2.2

export PATH=$PATH:$HBASE\_HOME/bin

通过source /etc/profile使文件生效

1. 配置环境变量
   1. 配置hbase-env.sh文件

vi /usr/local/hbase/conf/hbase-env.sh

* 1. 添加环境变量

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_151

export HBASE\_LOG\_DIR=/var/log/hbase

export HBASE\_MANAGES\_ZK=false 如果使用HBase自带的Zookeeper值设成true如果使用自己安装的Zookeeper需要将该值设为false

* 1. 拷贝zookeeper的配置文件至hbase目录

cp /hadoop/zookeeper-3.5.6/conf/zoo.cfg /hadoop/hbase-2.2.2/conf/

* 1. 配置hbase-site.xml文件

vi /usr/local/hbase/conf/hbase-site.xml

在 <configuration> </configuration> 中添加如下内容:

<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,datanode</value>

</property>

<property>

<name>hbase.master.info.bindAddress</name>

<value>0.0.0.0</value>

</property>

<property>

<name>hbase.master.info.port</name>

<value>16010</value>

</property>

<property>

<name>hbase.master.port</name>

<value>16000</value>

</property>

* 1. 配置文件详解

hbase.rootdir指定Hbase数据存储目录

hbase.cluster.distributed 指定是否是完全分布式模式，单机模式和伪分布式模式需要将该值设为false

hbase.zookeeper.quorum 指定zookeeper的集群，多台机器以逗号分隔（建议使用单数）

Hhbase.master.info.bindAddress Base Master web 界面绑定的地址 默认: 0.0.0.0

hbase.master.info.port HBase Master web 界面端口.设置为-1 意味着你不想让他运行。默认: 60010

hbase.master.port Hbase的Master的端口.默认: 60000

配置regionservers文件

进入到hbase/conf目录下，vim ./regionservers打开文件,修改文件内容如下图所示

文件中保存的是hadoop集群的 namenode节点和datanode节点的主机名，需要根据实际情况修改

* 1. 修改hadoop的配置文件hdfs-site.xml

进入到hadoop的配置文件目录，vim ./hdfs-site.xml打开文件。

在文件中添加一下内容

<property>

<name>dfs.datanode.max.xcievers</name>

<value>4096</value>

</property>

该参数限制了dathadoop-defaults.xml中通常不设置这个参数。这个限制看有些偏小。

anode所允许同时执行的发送和接受任务的数量，缺省为256，注意：这一步需要在hadoop集群的所有主机上都执行该操作。

* 1. 将Hbase拷贝到所有节点

scp -r hbase datanode:/usr/local/

* 启动Hbase

启动顺序：hadoop->hbase,如果系统中使用了自己安装的zookeeper，则启动顺序是：hadoop->zookeeper->hbase。停止顺序与启动顺序正好相反

cd /usr/local/hadoop/sbin

./start-all.sh

cd /usr/local/zookeeper/bin/

./zkServer.sh start

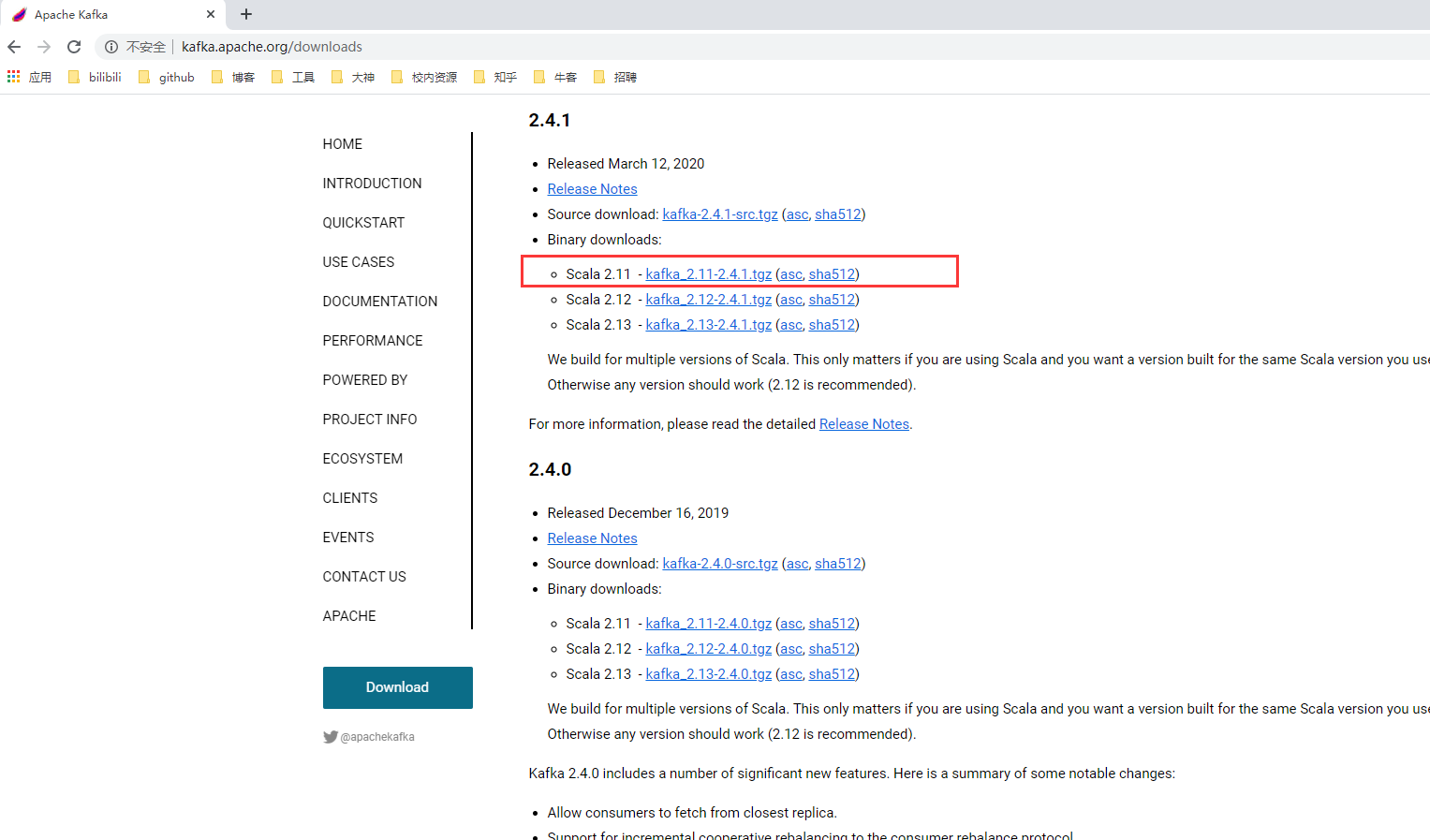
cd /usr/local/hbase/bin/

./start-hbase.sh

### 1.3.6 Kafka配置

* 安装步骤：
  + 下载安装包

<http://kafka.apache.org/downloads>



* + 解压安装包

tar -zxvf kafka\_2.11-2.4.1.tgz -C ./

* + 在/BigData/kafka-2.4.1下创建logs文件夹

[root@slave1 kafka-2.4.1]# mkdir logs

* + 修改配置文件

cd config

vim server.properties

1、输入以下内容

# The id of the broker. This must be set to a unique integer for each broker.

broker.id=1

delete.topic.enable=true

#处理网络请求的线程数量

num.network.threads=3

#用来处理磁盘 IO 的现成数量

num.io.threads=8#发送套接字的缓冲区大小socket.send.buffer.bytes=102400

#接收套接字的缓冲区大小

socket.receive.buffer.bytes=102400

#请求套接字的缓冲区大小socket.request.max.bytes=104857600

#kafka 运行日志存放的路径

log.dirs=/BigData/kafka-2.4.1/logs

#topic 在当前 broker 上的分区个数

num.partitions=1

#用来恢复和清理 data 下数据的线程数量

num.recovery.threads.per.data.dir=1

#segment 文件保留的最长时间，超时将被删除

log.retention.hours=168

#配置连接 Zookeeper 集群地址

zookeeper.connect==master:2181,slave1:2181,slave2:2181

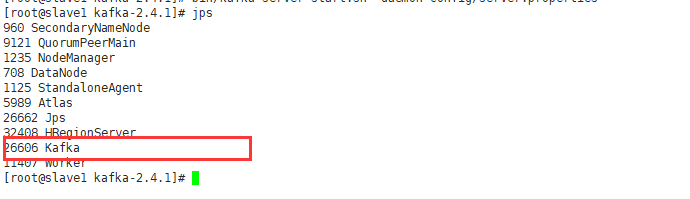
2、分发安装包

[root@slave1 BigData]# scp kafka-2.4.1/ master:/BigData/

[root@slave1 BigData]# scp kafka-2.4.1/ slave2:/BigData/

* + - 修改master、slave2上的config/server.properties，将broker.id=2,broker.id=3，因为broker.id不能重复复
* 检测安装成功，启动kafka

[root@slave1 kafka-2.4.1]# bin/kafka-server-start.sh -daemon config/server.properties



### 1.3.7 Flulme配置

* Flume安装地址

1） Flume官网地址

http://flume.apache.org/](http://flume.apache.org/

2）文档查看地址

http://flume.apache.org/FlumeUserGuide.html

3）下载地址

http://archive.apache.org/dist/flume/

* 安装部署

1）将apache-flume-1.9.0-bin.tar.gz上传到linux的/BigData目录下

2）解压apache-flume-1.9.0-bin.tar.gz到/opt/module/目录下

tar -zxf apache-flume-1.9.0-bin.tar.gz

3）修改apache-flume-1.9.0-bin的名称为flume-1.9.0

mv apache-flume-1.9.0-bin flume-1.9.0



4）将flume/conf下的flume-env.sh.template文件修改为flume-env.sh，并配置flume-env.sh文件

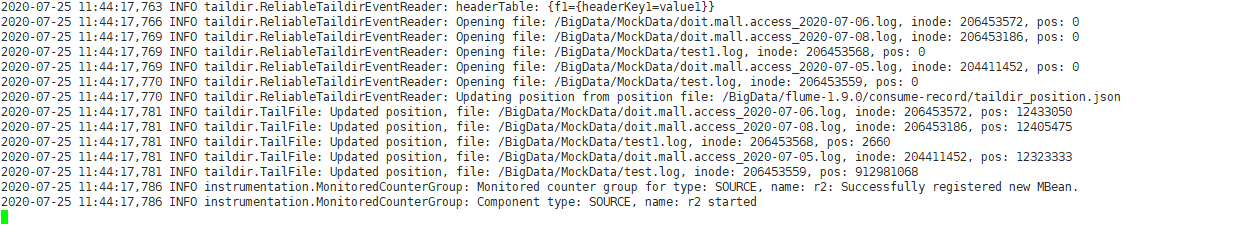
mv flume-env.sh.template flume-env.sh

vim flume-env.sh

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_151

5）检测启动成功



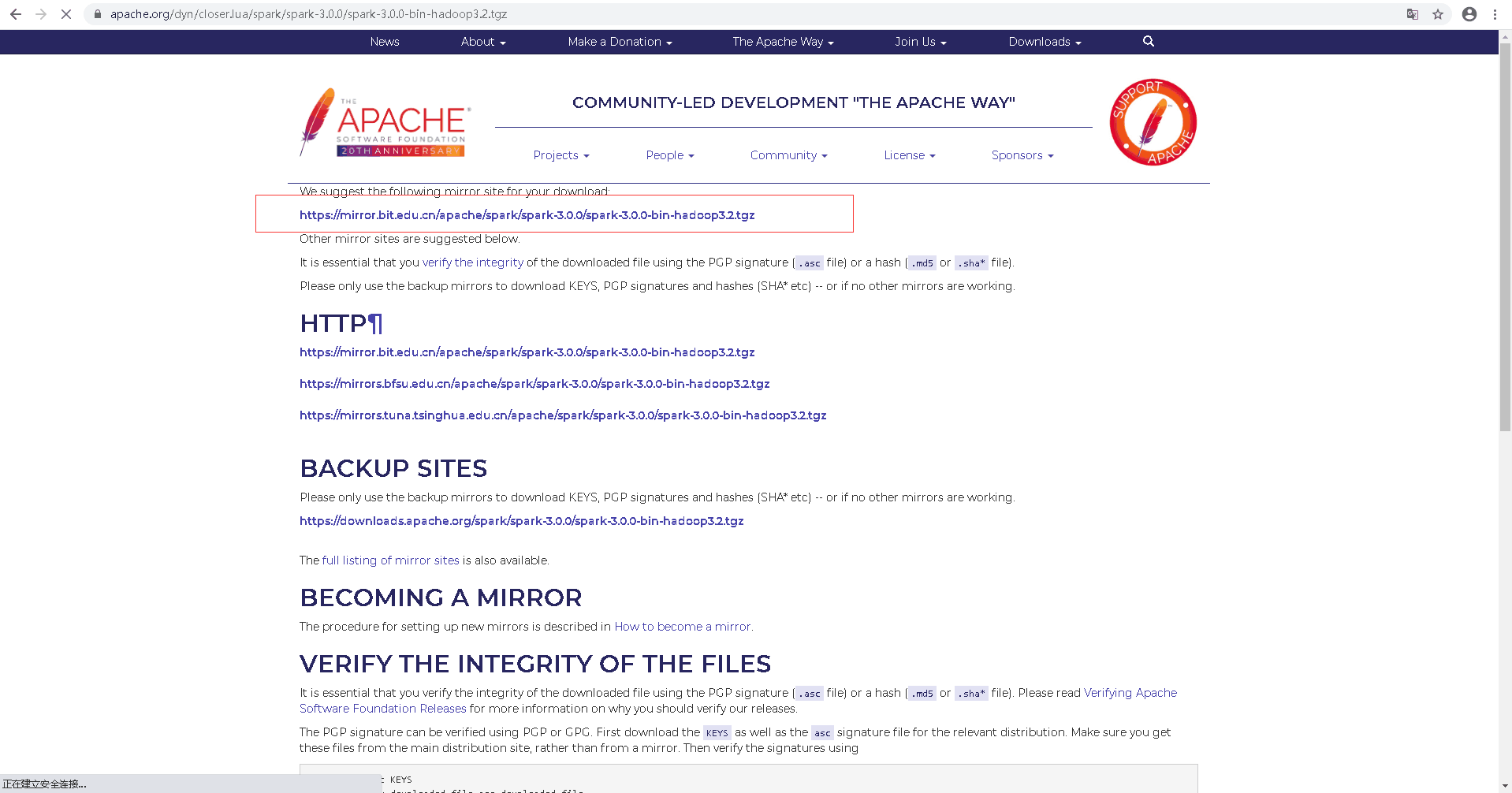


### 1.3.8 Spark配置

* 安装步骤

1. 下载安装包

<https://www.apache.org/dyn/closer.lua/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz>



1. 解压安装包

[root@slave1 BigData]# tar -zxvf spark-3.0.0-bin-hadoop3.2.tgz -C ./

1. 修改配置文件

[root@slave1 conf]# vim slaves

* 检测安装成功

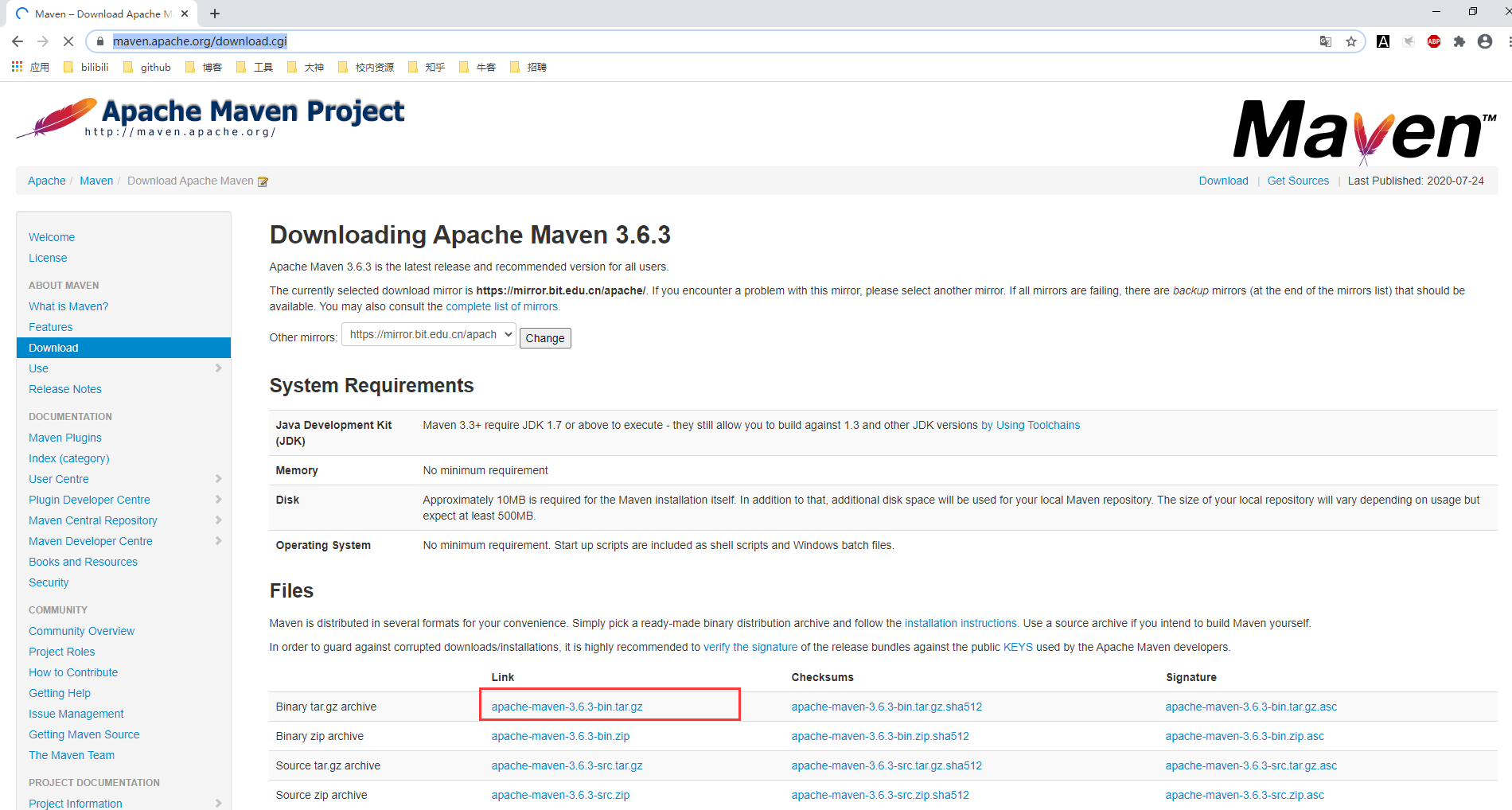
[root@slave1 sbin]# ./start-all.sh

### 1.3.9 Atlas配置

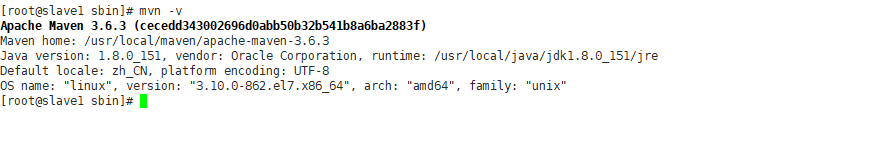
* 安装步骤

1. 下载Maven

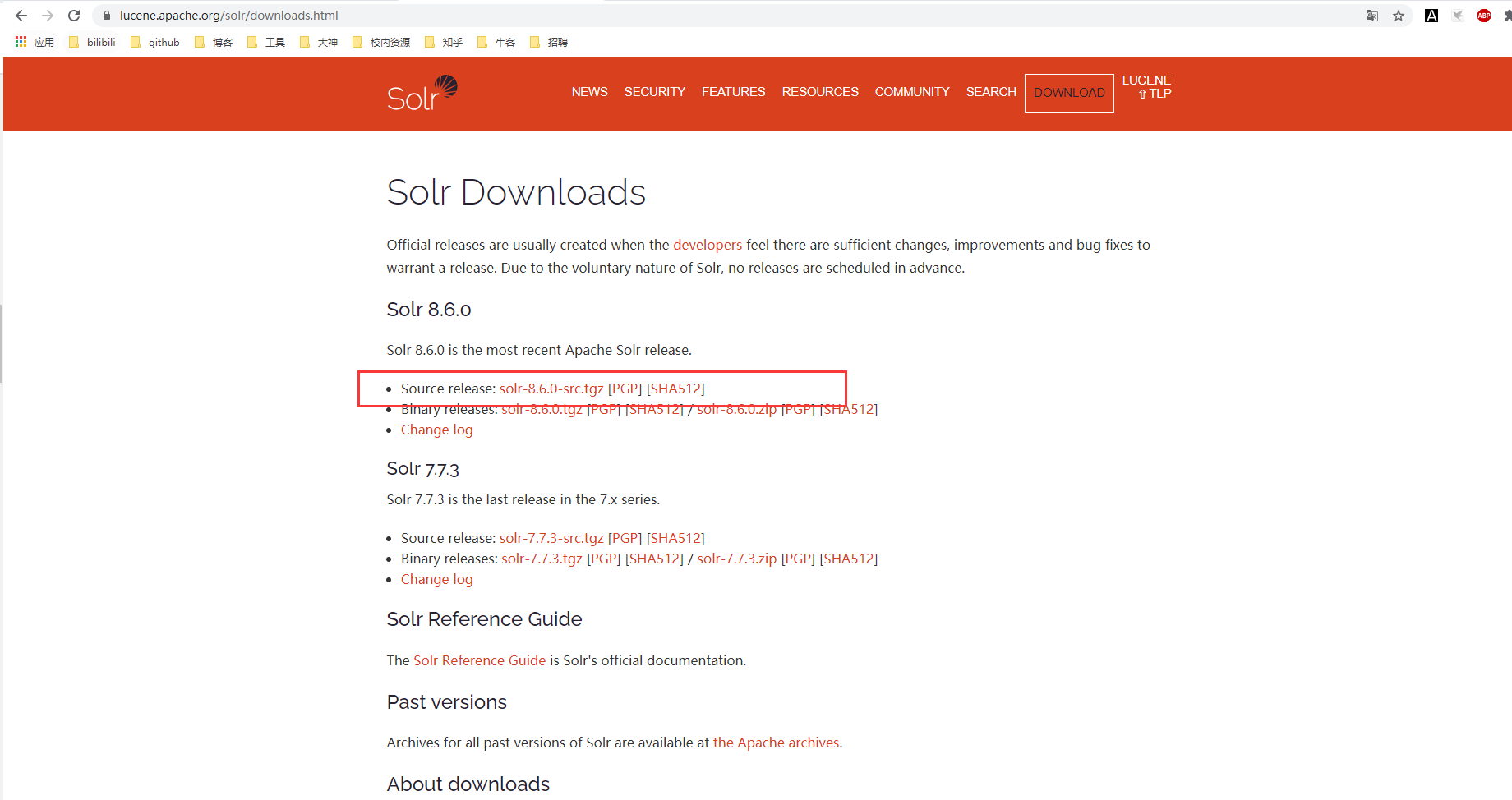
<https://maven.apache.org/download.cgi>



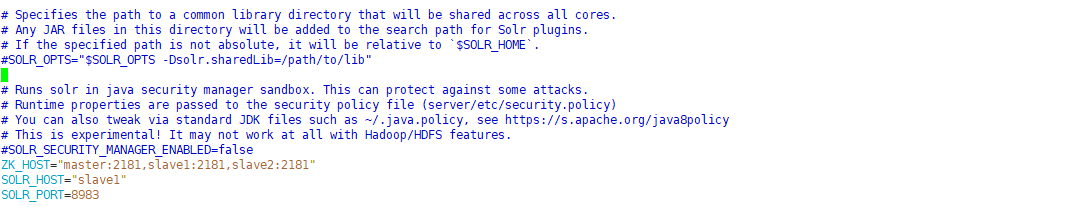
下载之后将压缩包解压，添加环境变量，测试安装结果：



1. 安装Solr

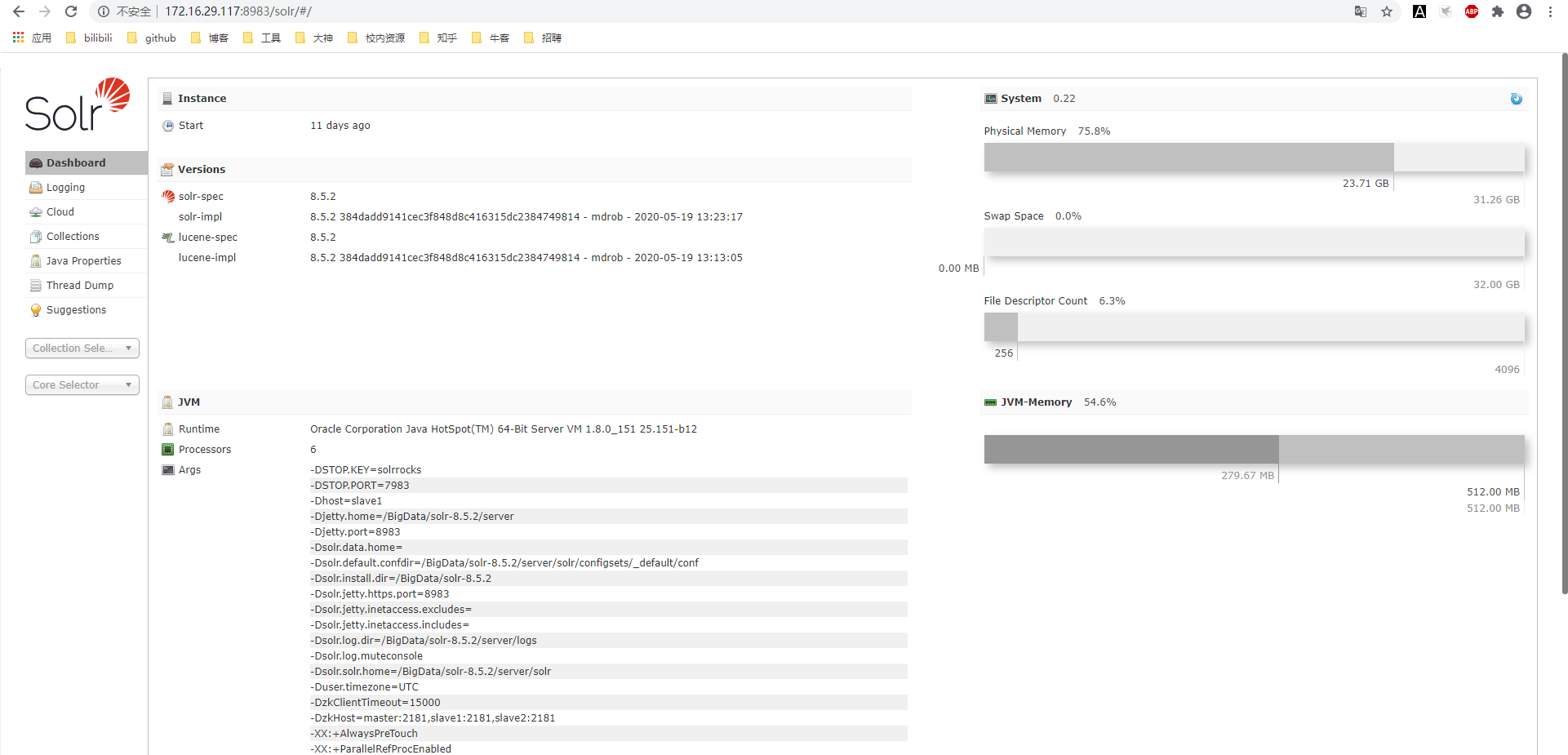


[root@slave1 bin]# vim solr.in.sh



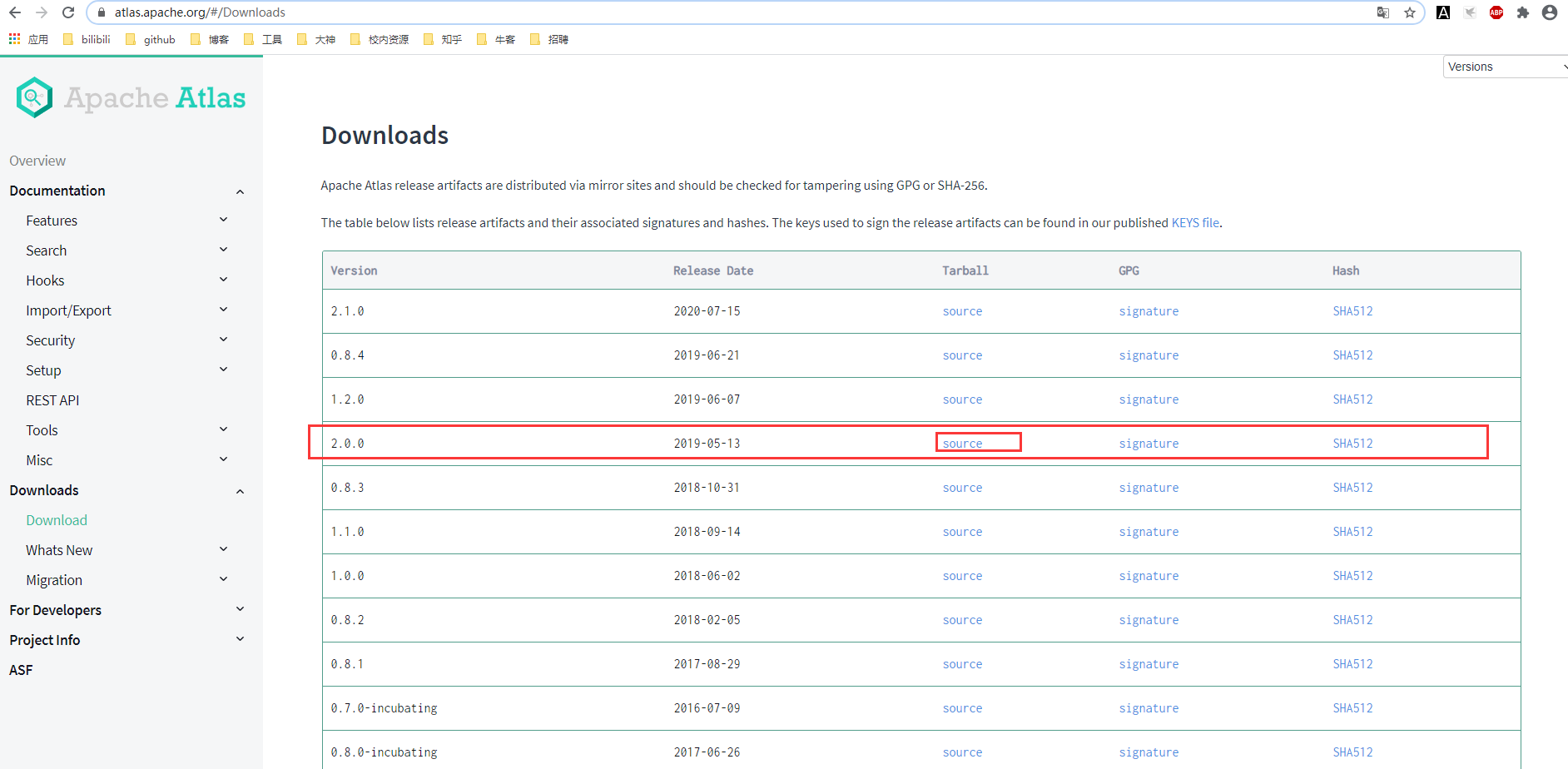
将solr分发到其余两台节点，并启动，进入172.16.29.117:8983查看web界面

[root@slave1 solr-8.5.2]# bin/solr start -force



1. 下载Atlas源码

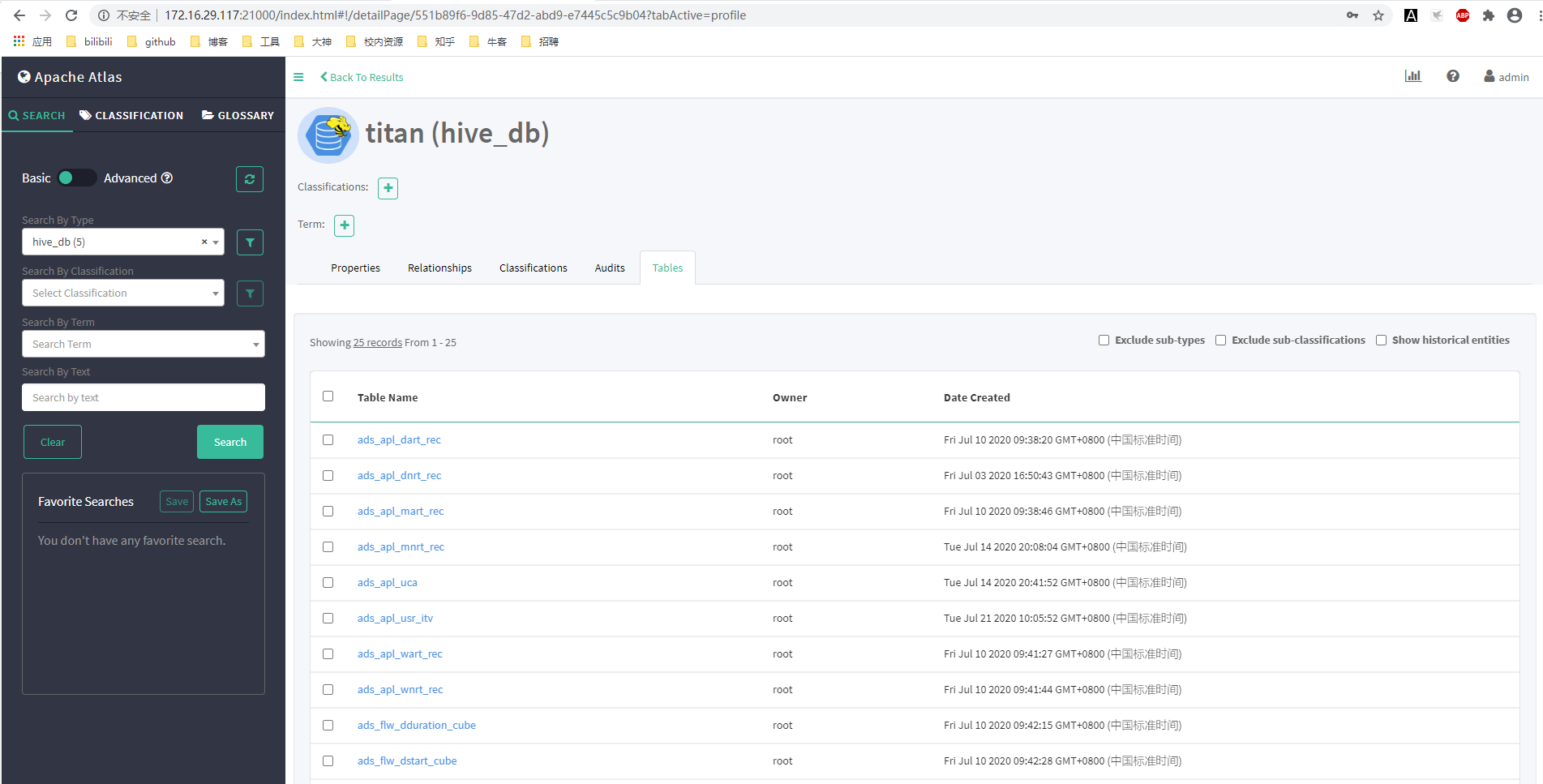
<https://atlas.apache.org/#/Downloads>



编译：

[root@slave1 apache-atlas-sources-2.0.0]# mvn clean -DskipTests install

将solr、kafka、hbase、hive集成atlas后，启动atlas，查看web界面

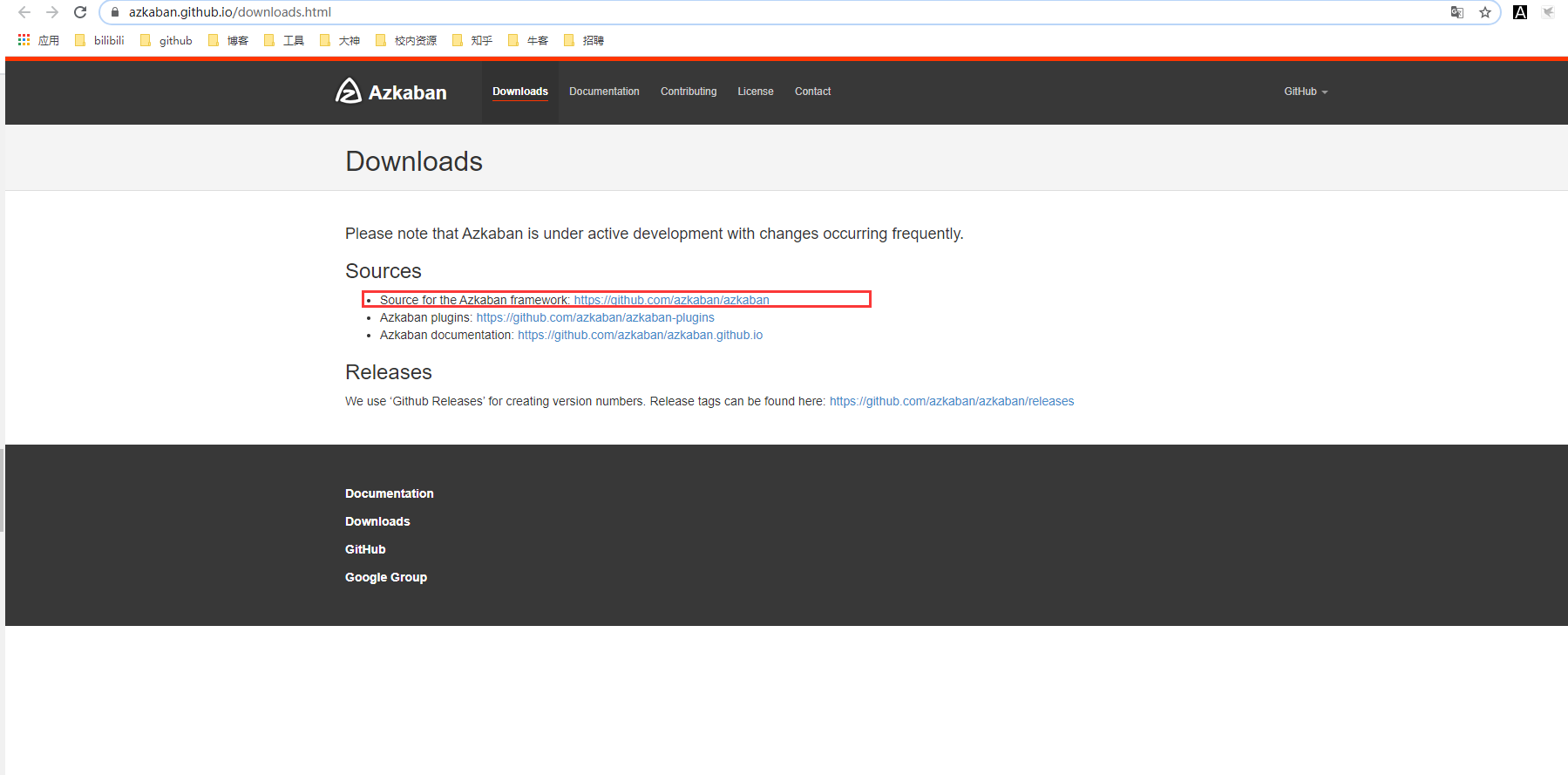


### 1.3.10 Azkaban配置

* **安装步骤**

1. 下载安装包

<https://azkaban.github.io/downloads.html>



1. 压缩包解压
2. 进入mysql，创建azkaban数据库，将数据库脚本导入到azkaban数据库

mysql> use azkaban;

mysql> mysql> source /BigData/azkaban-3.90.0/azkaban-db/create-all-sql-0.1.0-SNAPSHOT.sql

1. 生成密钥库

[root@slave2 azkaban-3.90.0]# keytool -keystore -alias jetty -genkey -keyalg RSA

[root@slave2 azkaban-3.90.0]# mv keystore azkaban-web-server/

1. 进入azkaban-web-server的conf中，修改azkaban.properties文件

# Azkaban Personalization Settings

azkaban.name=Test

azkaban.label=My Local Azkaban

azkaban.color=#FF3601

azkaban.default.servlet.path=/index

web.resource.dir=/BigData/azkaban-3.90.0/azkaban-web-server/web

default.timezone.id=Asia/Shanghai

# Azkaban UserManager class

user.manager.class=azkaban.user.XmlUserManager

user.manager.xml.file=/BigData/azkaban-3.90.0/azkaban-web-server/conf/azkaban-users.xml

# Loader for projects

executor.global.properties=/BigData/azkaban-3.90.0/azkaban-web-server/conf/global.properties

azkaban.project.dir=projects

# Velocity dev mode

velocity.dev.mode=false

# Azkaban Jetty server properties.

jetty.use.ssl=false

jetty.maxThreads=25

jetty.port=8082

#SSL 文件名（绝对路径）

jetty.keystore=/BigData/azkaban-3.90.0/azkaban-web-server/keystore

#SSL 文件密码

jetty.password=titan123

#Jetty 主密码与 keystore 文件相同

jetty.keypassword=titan123

#SSL 文件名（绝对路径）

jetty.truststore=/BigData/azkaban-3.90.0/azkaban-web-server/keystore

#SSL 文件密码

jetty.trustpassword=titan123

# Azkaban Executor settings

# mail settings

mail.sender=

mail.host=

# User facing web server configurations used to construct the user facing server URLs. They are useful when there is a reverse proxy between Azkaban web servers and users.

# enduser -> myazkabanhost:443 -> proxy -> localhost:8081

# when this parameters set then these parameters are used to generate email links.

# if these parameters are not set then jetty.hostname, and jetty.port(if ssl configured jetty.ssl.port) are used.

# azkaban.webserver.external\_hostname=myazkabanhost.com

#azkaban.webserver.external\_ssl\_port=443

azkaban.webserver.external\_port=8083

job.failure.email=

job.success.email=

lockdown.create.projects=false

cache.directory=cache

# JMX stats

jetty.connector.stats=true

executor.connector.stats=true

# Azkaban mysql settings by default. Users should configure their own username and password.

database.type=mysql

mysql.port=3306

mysql.host=localhost

mysql.database=azkaban

mysql.user=root

mysql.password=12345678

mysql.numconnections=100

#Multiple Executor

azkaban.use.multiple.executors=true

azkaban.executorselector.filters=StaticRemainingFlowSize,MinimumFreeMemory,CpuStatus

azkaban.executorselector.comparator.NumberOfAssignedFlowComparator=1

azkaban.executorselector.comparator.Memory=1

azkaban.executorselector.comparator.LastDispatched=1

azkaban.executorselector.comparator.CpuUsage=1

1. 进入azkaban-exec-server的conf目录，修改azkaban.properties

# Azkaban Personalization Settings

azkaban.name=Test

azkaban.label=My Local Azkaban

azkaban.color=#FF3601

azkaban.default.servlet.path=/index

web.resource.dir=web/

default.timezone.id=Asia/Shanghai

# Azkaban UserManager class

user.manager.class=azkaban.user.XmlUserManager

user.manager.xml.file=conf/azkaban-users.xml

# Loader for projects

executor.global.properties=conf/global.properties

azkaban.project.dir=projects

# Velocity dev mode

velocity.dev.mode=false

# Azkaban Jetty server properties.

jetty.use.ssl=false

jetty.maxThreads=25

jetty.port=8081

# Where the Azkaban web server is located

azkaban.webserver.url=http://localhost:8083

# mail settings

mail.sender=

mail.host=

# User facing web server configurations used to construct the user facing server URLs. They are useful when there is a reverse proxy between Azkaban web servers and users.

# enduser -> myazkabanhost:443 -> proxy -> localhost:8081

# when this parameters set then these parameters are used to generate email links.

# if these parameters are not set then jetty.hostname, and jetty.port(if ssl configured jetty.ssl.port) are used.

# azkaban.webserver.external\_hostname=myazkabanhost.com

# azkaban.webserver.external\_ssl\_port=443

# azkaban.webserver.external\_port=8081

job.failure.email=

job.success.email=

lockdown.create.projects=false

cache.directory=cache

# JMX stats

jetty.connector.stats=true

executor.connector.stats=true

# Azkaban plugin settings

azkaban.jobtype.plugin.dir=plugins/jobtypes

# Azkaban mysql settings by default. Users should configure their own username and password.

database.type=mysql

mysql.port=3306

mysql.host=localhost

mysql.database=azkaban

mysql.user=root

mysql.password=12345678

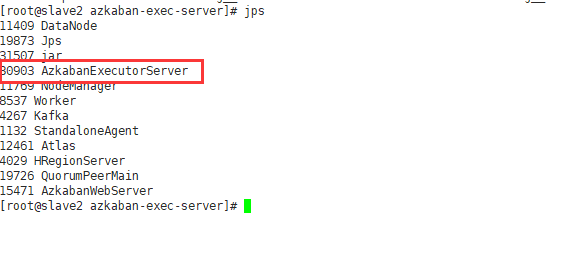
mysql.numconnections=100

# Azkaban Executor settings

executor.maxThreads=50

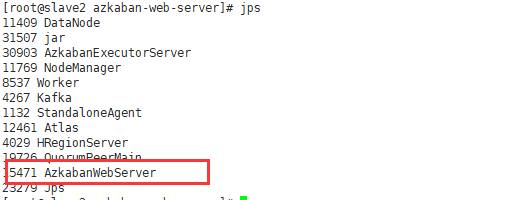
executor.flow.threads=30

1. 启动executor服务器

[root@slave2 azkaban-exec-server]# bin/start-exec.sh

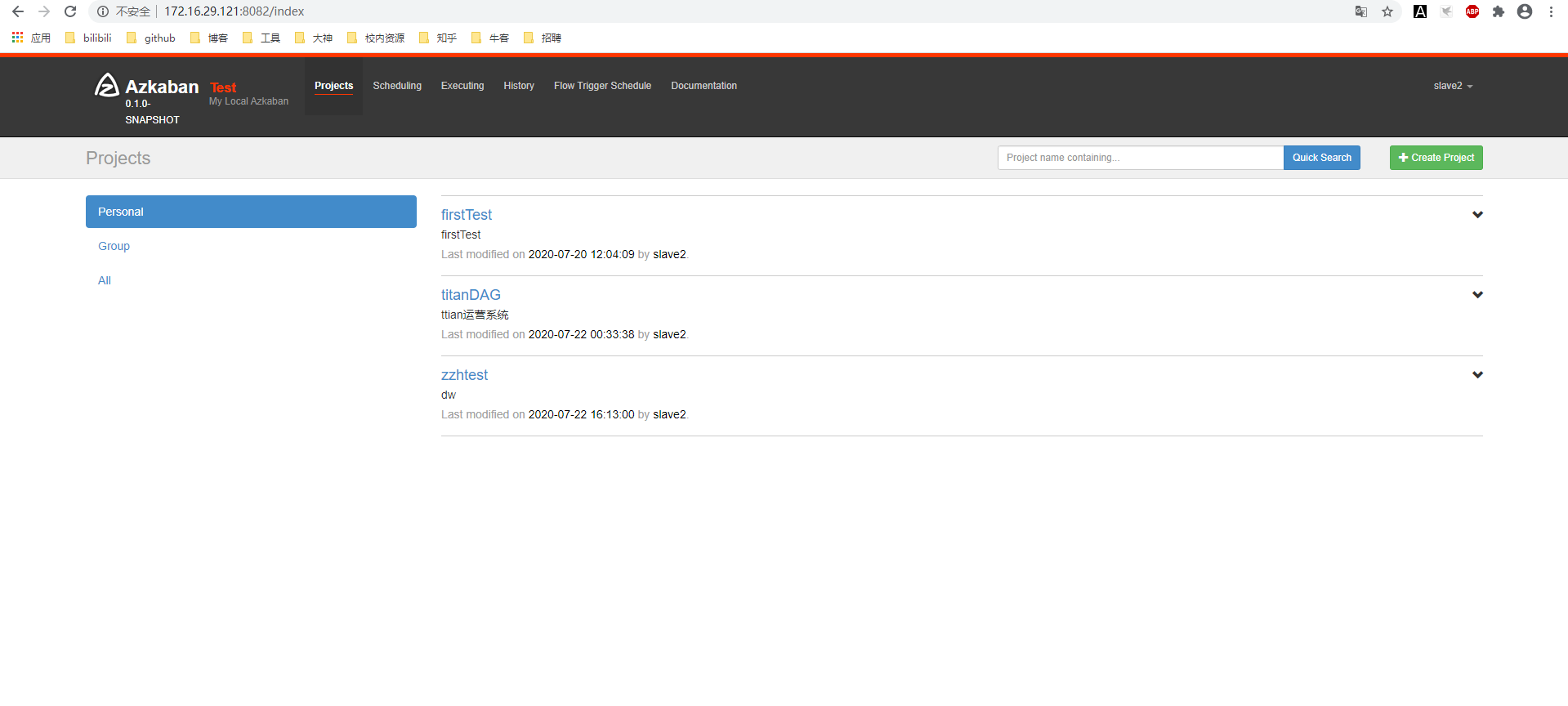
1. 启动web服务器

[root@slave2 azkaban-web-server]# bin/start-web.sh



* 检测安装成功

http://172.16.29.121:8082/index



# 第2章 模块安装与配置

## 2.1 总体说明

本项目主要包含三个子模块

* 数据提取、存储模块
* 分析、迁移定时任务模块
* 分析结果可视化展示模块

安装启动顺序：

1. 启动数据提取、存储模块
2. 启动分析结果可视化展示模块
3. 启动定时任务模块

## 2.2 数据库数据初始化

* Hive数仓初始化：

1. 将hive建表语句中的 hive.sql 文件上传至集群。
2. 在控制台输入 hive -e hive.sql 文件进行数仓表的初始化。

* MYSQL数据库初始化：

1. 将mysql建表语句中的 titan.sql 文件上传至集群。
2. 使用 mysql -u $USER -p，输入密码后 进入mysql数据库。
3. 运行source titan.sql 命令进行数据库初始化。

## 2.3 系统安装部署

### 2.3.1 数据提取、存储子系统

1. 子系统介绍

负责对埋点日志数据进行监控，并将增量数据最终添加到高可用的HDFS存储系统中，作为数据分析的数据源。

1. 获取包

无

1. 安装

无

1. 配置

①flume 监听本地文件变化，将数据上传至kafka的配置

a2.sources = r2

a2.channels = c2

# source

a2.sources.r2.type = TAILDIR

a2.sources.r2.positionFile = /BigData/flume-1.9.0/consume-record/taildir\_position.json

a2.sources.r2.filegroups = f1

a2.sources.r2.filegroups.f1 = /BigData/MockData/.\*log

a2.sources.r2.headers.f1.headerKey1 = value1

a2.sources.r2.fileHeader = true

a2.sources.r2.maxBatchCount = 1000

#channel

a2.channels.c2.type = org.apache.flume.channel.kafka.KafkaChannel

#设置到kafka的哪个topic以及flume消费者组

a2.channels.c2.kafka.bootstrap.servers = master:9092,slave1:9092,slave2:9092

a2.channels.c2.parseAsFlumeEvent=false

a2.channels.c2.kafka.topic = titan-log-topic

a2.channels.c2.kafka.consumer.group.id = titan-flume-consumer

a2.sources.r2.channels = c2

②flume监听kafka，将数据写到hdfs

a3.sources = r3

a3.channels = c3

a3.sinks = k3

# source

a3.sources.r3.type = org.apache.flume.source.kafka.KafkaSource

a3.sources.r3.batchSize = 5000

a3.sources.r3.batchDurationMillis = 2000

a3.sources.r3.kafka.bootstrap.servers = master:9092,slave1:9092,slave2:9092

a3.sources.r3.kafka.topics = titan-log-topic

a3.sources.r3.kafka.consumer.group.id = titan-flume-consumer

#channel

a3.channels.c3.type = file

a3.channels.c3.checkpointDir = /BigData/flume-1.9.0/data/checkpoint

a3.channels.c3.dataDirs = /BigData/flume-1.9.0/data/realData

a3.channels.c3.maxFileSize = 2146435071

a3.channels.c3.capacity = 1000000

a3.channels.c3.keep-alive = 6

#sink

a3.sinks.k3.type = hdfs

#设置随时间滚动

a3.sinks.k3.hdfs.path = /data/titan/events\_logs/%Y-%m-%d/

#上传文件前缀

a3.sinks.k3.hdfs.filePrefix = events-

# 设置压缩格式

a3.sinks.k3.hdfs.fileType = CompressedStream

a3.sinks.k3.hdfs.codeC = lzop

#不要产生大量小文件

a3.sinks.k3.hdfs.rollInterval = 3600

#刚好是128M

a3.sinks.k3.hdfs.rollSize = 134217728

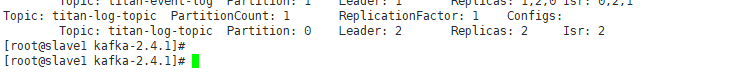
a3.sinks.k3.hdfs.rollCount = 0

#拼装

a3.sources.r3.channels = c3

a3.sinks.k3.channel = c3

③kafka配置



1. 启停

①启动监听本地文件的flume

nohup /BigData/flume-1.9.0/bin/flume-ng agent -c conf -f /BigData/flume-1.9.0/job/titan-file-flume-kafka.conf -name a2 -Dflume.root.logger=INFO,LOGFILE > /BigData/flume-1.9.0/logs/f1.log 2>&1 &

②启动监听kafka的flume

nohup /BigData/flume-1.9.0/bin/flume-ng agent --conf-file /BigData/flume-1.9.0/job/titan-kafka-flume-hdfs.conf --name a3 -Dflume.root.logger=INFO,LOGFILE >/BigData/flume-1.9.0/logs/f2.log 2>&1 &

③停止监听本地文件的flume

ps -ef | grep titan-file-flume-kafka | grep -v grep | awk '{print \$2}' | xargs kill

④停止监听kafka的flume

ps -ef | grep titan-kafka-flume-hdfs.conf | grep -v grep | awk '{print \$2}' | xargs kill

### 2.3.2分析结果可视化展示子系统

1. 子系统介绍

负责对存储在mysql数据库的最终分析结果、报表数据进行网页形式的可视化展示。

1. 获取包

将 titanApp-0.0.1-SNAPSHOT.jar 包上传至集群。

1. 安装：无需安装。
2. 配置

打开jar包，编辑titan.properties文件进行配置，主要修改项如下：

* spring.datasource.url ：数据库URL
* spring.datasource.username ：数据库用户名
* spring.datasource.password ：用户名对应密码
* spring.datasource.driverClassName：数据库驱动对应类名
* server.port：端口号

1. 启停

使用 java -jar titanApp-0.0.1-SNAPSHOT.jar 命令进行启动。

使用 kill -9 {pid} 进行停止。

### 2.3.3分析、迁移定时任务子系统

1. 子系统介绍

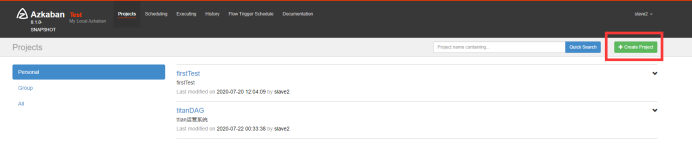
每天对hdfs上存储的日志数据在数仓环境中进行分析计算，并将分析结果从数仓迁移至mysql数据库。

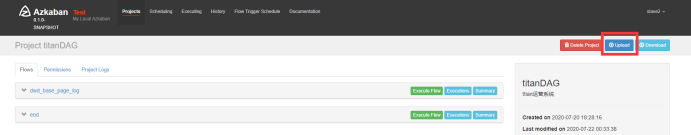
1. 获取包

将 titanSpark-1.0-SNAPSHOT.jar、job.zip 上传至集群。

1. 安装

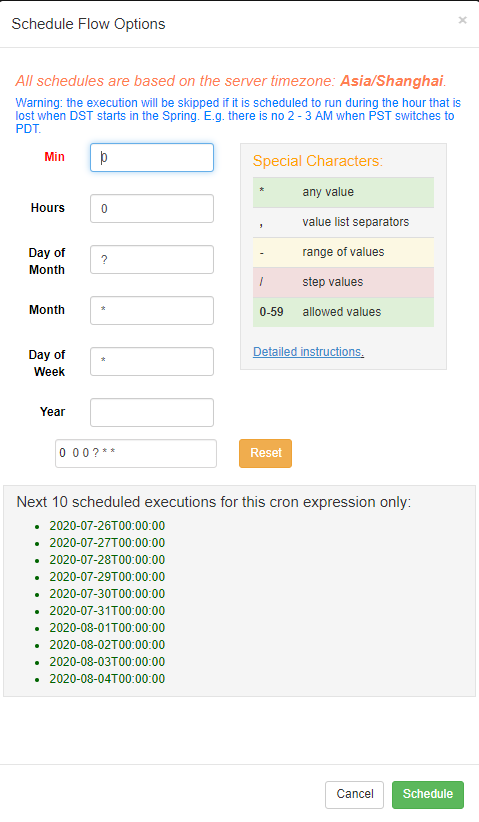
在 azkaban 的 web 管理界面创建工程并上传 zip 包



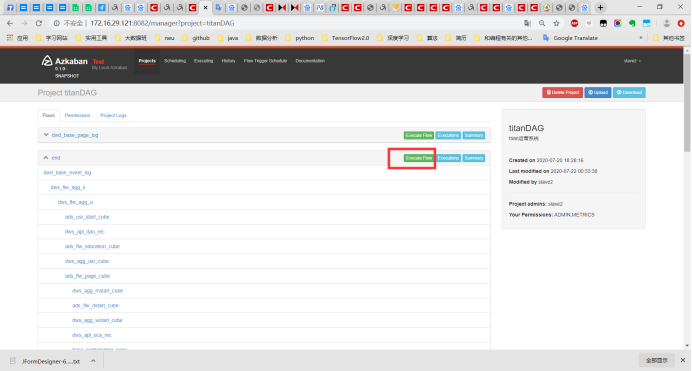


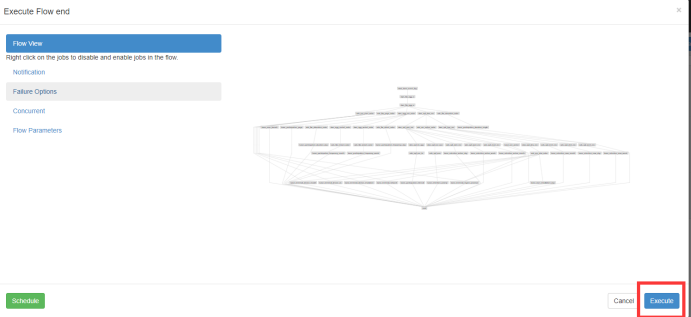
1. 配置

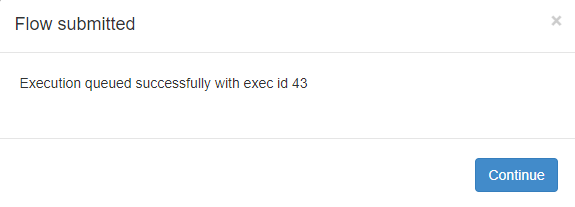
任务执行时间配置



1. 启停







## 2.4 模块使用

* 数据提取、存储模块启动后无需进一步操作。
* 分析、迁移定时任务模块启动后定时运行无需进一步操作。
* 分析结果可视化展示模块启动后可通过

{启动主机名}:{端口号}/src/material/trend\_summary.html 访问主页，供用户交互使用。

# 第3章 其他事项

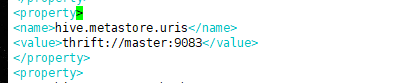
## 3.1 故障排查

### 3.1.1 hive metastroe 的问题

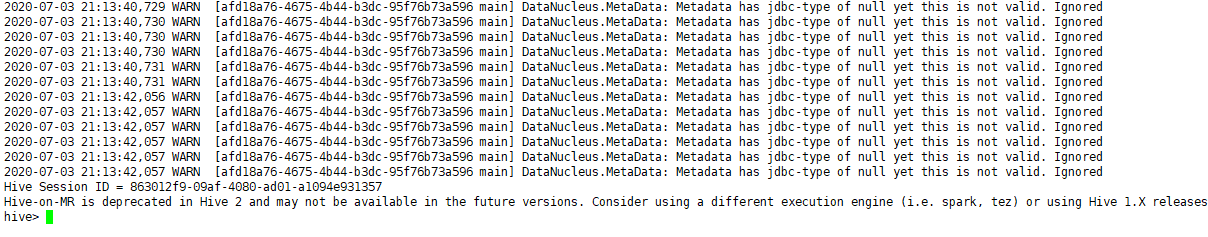
这个问题最早的时候是在启动 hive 的时候遇到的，当时报错是：



我把 hive-site.xml 中的这部分配置注释掉之后就不再报错了，但是会在启动 hive 的时候有警告，当时把这个问题直接忽略过去了，因为已经可以 hive 已经可以正常运行了

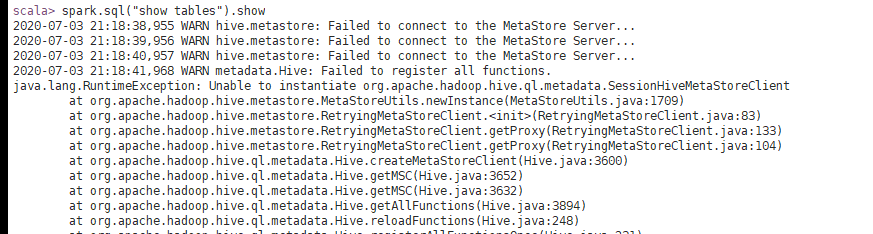


（被注释掉的代码）



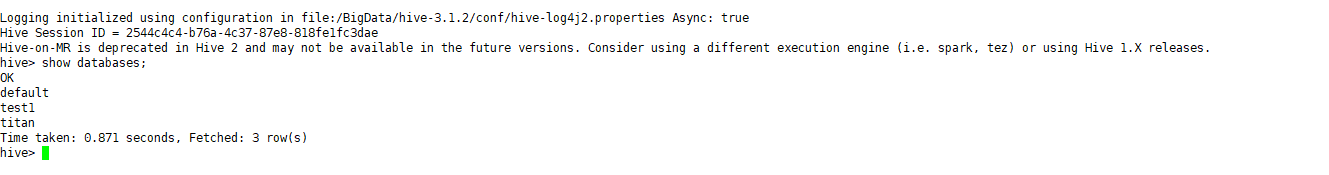
（启动 hive 时的警告）

后来在尝试 spark on hive 的时候，报了一个这个错误：

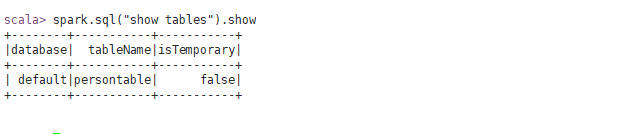


通过查资料，发现是需要启动 service metastore，看到 metastore 就很熟悉，上面遇到过，于是我把上面注释掉的代码解除注释，并在 master（与配置文件里的 ip 对应） 上运行 **hive --service metastore** 命令，重新启动 spark-shell，问题解决。

nohup hive --service metastore 2>/dev/null &（运行命令）



（hive 正常）



（spark 正常）

### 3.1.2hiveserver2 的启动问题

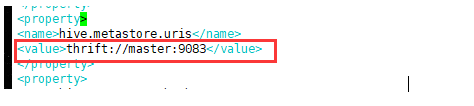
我们启动 hiveserver2 来开启 10000 端口，通过 idea 连接到 hive，方便地写建表语句，第一次启动没有任何问题，idea 成功连接，但是过了一段时间，idea 就突然连接不上去了，我到 linux 把进程先 kill 掉，然后重新运行，发现报错：

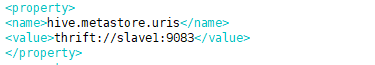


我当时从网上查了一些资料，发现与我们的情况都不符合，我们不可能是因为缺少包导致的，否则第一次就启动不起来，所以我就尝试在 master 上也启动 hiverserver2，发现报了同样的错误，但是在该错误的下面还有其他正常信息：



看起来是正常的样子，于是我尝试用 idea 连接 master 的 hive，发现连上去了，**果然和缺少Tez 没有关系，**master 可以启动 hiveSerer2，但是 slave1 不可以，二者关于 hive 的差别就是在这里（图一），master 就位于 metastore 服务的主机上，而 slave1 不是。因为我们要把 slave1 当作 hive 的主机，所以我就在三台节点上都做了更改，把 master 改成 slave1（图二）

（图一）

 （图二）

更改完配置之后，kill 掉 master 上的 hive metastore service，然后在 slave1 上启动 hive metastore service 和 hiveserver2，问题解决！

## 3.2 Q&A

1. Q：依赖组件众多，是否均为必须安装。

A：HBase、Solar、Atlas为非必须安装，主要是为了提高数据资产的管理能力，其余均必须安装。

1. Q: 为什么采用的采集架构为 flume – kafka – flume – hdfs，没有直接采用 fume-hdfs ？

A：在真实环境中，日志可能来源于多个服务器，如果直接用 flume 传输到 hdfs，那么会因为日志的输入量及速度过大，导致接收服务器承受不了这么高的压力而丢失数据，kafka作为一种高可靠的消息队列，可以很好的解决这个问题。

1. Q：能否使用更加稳定的版本代替本项目中的最新的组件版本。

A：不能，本项目中采用了最新版本组件的一些特性，且各个组件之间的版本依赖关系较为复杂。本项目在开发过程中已经经过较严密的测试，如更换版本可能会造成一些不可预见的问题。