Cdh-test使用demo及其httpAPI的保护

【目录】

【第一章】Cdh-test运行环境准备

【第二章】安装HA-hadoop集群、HA-HBASE集群、zookeeper集群

【第三章】安装Gnuplot、安装cdh-test

【第四章】通过Nginx对Cdh-test做用户验证，确保cdh-test http接口安全

本文提到的各种服务和操作都经过本人验证，期间的坑已经提你们踩过了，读者可以参照本文档来一步步实施。觉得有用可以收藏。

【第一章】Cdh-test运行环境准备

【操作系统】CentOS Linux release 7.2.1511 (Core)

【主机列表】

主机名 内网IP地址

cdh-test-1-4 192.168.25.2

cdh-test-1-5 192.168.25.3

cdh-test-1-1 192.168.25.4

cdh-test-1-3 192.168.25.5

cdh-test-1-2 192.168.25.6

【系统脚本语言版本】内置的Python 2.7.5

【cdh-test要求java版本Java-1.6+】

[root@cdh-test-1-4 ~]# yum makecache # 更新yum源

[root@cdh-test-1-4 tony\_soft]# for x in {5,1,3,2}; do ssh root@cdh-test-1-$x yum makecache; done

[root@cdh-test-1-4 tony\_soft]# yum install -y java-1.8.0-openjdk # 使用yum安装JDK-1.8

解决no jps in (/usr/local/sbin问题：yum install -y java-1.8.0-openjdk-devel.x86\_64

继续为其他四个节点分别安装openjdk和jps：

[root@cdh-test-1-4 tony\_soft]# for x in {5,1,3,2}; do ssh root@cdh-test-1-$x yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel.x86\_64 >> /tmp/jdk\_and\_jps\_INSTALLING.log 2>&1;java -version && jps;echo "FINISHED\_\_\_\_ $x";done

*（或者也可安装Oracle官方的版本yum install oracle-j2sdk1.8）*

【分别下载zookeeper-3.4.14、Hadoop-2.8.4、hbase-1.4.9】

wget <https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/zookeeper-3.4.14/zookeeper-3.4.14.tar.gz>

wget <https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-2.8.4/hadoop-2.8.4.tar.gz>

wget <https://mirrors.tuna.tsinghua.edu.cn/apache/hbase/stable/hbase-1.4.9-bin.tar.gz>

【配置ssh免密登录】

确保从cdh-test-1-4到其他节点能ssh免密登录，具体实施可参考博文：

<https://jingyan.baidu.com/article/bea41d43b558ffb4c51be6db.html>

【开发自定义批处理脚本】

脚本：

可执行文件callall 用于同时在集群所有节点执行一条命令

可执行文件send2others 用于向其他节点(除cdh-test-1-4之外的节点)传送文件

都放到/usr/local/bin里面，给一个合适的权限。

[root@cdh-test-1-4 etc]# ll -ht /usr/local/bin/ | head -n 3

total 8

-rwxr-xr-x 1 root root 278 4月 24 15:57 callall

-rwxr-xr-x 1 root root 384 4月 24 16:00 send2others

[root@cdh-test-1-4 etc]# cat /usr/local/bin/callall

#!/bin/bash

if [[ $# -lt 1 ]] ; then echo no params ; exit ; fi

cmd=$\*

for h in {"cdh-test-1-5","cdh-test-1-1","cdh-test-1-3","cdh-test-1-2"}

do

echo "正在$h 执行命令$cmd"

ssh $h "$cmd"

if [ $? -eq 0 ];then

echo "--------命令执行成功！"

fi

done ;

[root@cdh-test-1-4 etc]# cat /usr/local/bin/send2others

#!/bin/bash

if [[ $# -lt 1 ]] ; then echo no params ; exit ; fi

p=$1

dir=`dirname $p`

filename=`basename $p`

cd $dir

fullpath=`pwd -P .`

user=`whoami`

for h in {"cdh-test-1-5","cdh-test-1-1","cdh-test-1-3","cdh-test-1-2"}

do

echo "正在同步文件到$h======="

rsync -lr $p ${user}@$h:$fullpath

if [ $? -eq 0 ];then

echo "--------已成功同步到$h"

fi

done ;

【各节点安装NTP时间同步服务】

安装、配置可参考《CentOS 7 中使用NTP进行时间同步》<https://www.linuxidc.com/Linux/2015-11/124911.htm>

【同步环境变量到其他四个节点】

创建符号链接：

ln -s hadoop-2.8.4 hadoop && callall ln -s hadoop-2.8.4 hadoop

ln -s hbase-1.4.9 hbase && callall ln -s hbase-1.4.9 hbase

ln -s zookeeper-3.4.14 zookeeper && callall ln -s zookeeper-3.4.14 zookeeper

send2others hadoop && send2others hbase && send2others zookeeper # 同步软连接到其他节点的相应位置

callall 'ls -al /tony\_soft/'

先在/etc/profile配好JAVA\_HOME、HADOOP\_HOME、HBASE\_HOME、ZOOKEEPER\_HOME；再用send2others 同步到其他四个节点：

cd /etc && send2others profile && callall 'source /etc/profile'

callall echo $PATH && callall echo $HADOOP\_HOME && callall echo $HBASE\_HOME && callall echo $ZOOKEEPER\_HOME

send2others hadoop && send2others hbase && send2others zookeeper # 同步符号链接到其他节点

send2others hbase-1.4.9 && send2others zookeeper-3.4.14 && send2others hadoop-2.8.4 # 同步各个软件包到其他节点

【第二章】安装HA-hadoop集群、HA-HBASE集群、zookeeper集群

【配置HA-hadoop集群】

角色规划：

cdh-test-1-4 NameNode, QPM(zk), Hmaster, ResourceManager

cdh-test-1-5 NameNode, QPM(zk), Hmaster, ResourceManager

cdh-test-1-1 JournalNode, QPM(zk), DataNode, HRegionServer

cdh-test-1-3 JournalNode, DataNode, HRegionServer

cdh-test-1-2 JournalNode, DataNode, HRegionServer

配置静态IP(略)、关闭防火墙

callall cat /etc/selinux/config | grep SELINUX # 确保每个主机的防火墙已关闭SELINUX=disabled

禁用Transparent Hugepage(为了提升hadoop集群性能)：

配置省略, 可参考<https://www.staroon.dev/2017/11/05/SetEnv/>）

为每个节点添加用户hadoop：

callall groupadd hadoop && callall useradd -g hadoop hadoop

创建hdfs相关目录(name目录、jn日志目录)并赋权：

callall 'mkdir -p **/dfs/data**' && callall 'chown -R hadoop:hadoop **/dfs**'

for i in {1,3,2}; do ssh cdh-test-1-$i mkdir **/jn\_edits**; done

for i in {1,3,2}; do ssh cdh-test-1-$i chown -R hadoop:hadoop **/jn\_edits**; done

拷贝hadoop相关命令到/usr/local/bin/目录下并赋权：

callall 'cp /tony\_soft/hadoop-2.8.4/bin/hdfs /usr/local/bin/ && chown hadoop:hadoop /usr/local/bin/hdfs'

callall ls -lh /usr/local/bin/hdfs

修改主要配置文件：

**core-site.xml**

<configuration>

<!-- 指定hdfs nameservice -->

<property>

<name>fs.defaultFS</name>

<value>**hdfs://tony/**</value>

</property>

<!-- 开启垃圾回收站功能，值为检查点被删除的分钟数，设为0表示禁用 -->

<property>

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

<value>1440</value>

</property>

<!-- 指定Zookeeper地址及端口 -->

<property>

<name>ha.zookeeper.quorum</name>

<value>cdh-test-1-4:2181,cdh-test-1-5:2181,cdh-test-1-1:2181</value>

</property>

<property>

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

<value>file:///dfs/data</value>

<description>namenode存放数据的目录,所有权是hadoop:hadoop</description>

</property>

</configuration>

**hdfs-site.xml**

<configuration>

<property>

<name>dfs.nameservices</name>

<value>tony</value>

</property>

<!—配置故障自动转移的实现方式 -->

<property>

<name>dfs.client.failover.proxy.provider.tony</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

<property>

<name>dfs.ha.namenodes.tony</name>

<value>nn1,nn2</value>

</property>

<property>

<name>dfs.namenode.rpc-address.tony.nn1</name>

<value>cdh-test-1-4:8020</value>

</property>

<property>

<name>dfs.namenode.http-address.tony.nn1</name>

<value>cdh-test-1-4:50070</value>

</property>

<property>

<name>dfs.namenode.rpc-address.tony.nn2</name>

<value>cdh-test-1-5:8020</value>

</property>

<property>

<name>dfs.namenode.http-address.tony.nn2</name>

<value>cdh-test-1-5:50070</value>

</property>

<!-- 指定各个JN节点的URI (hostname**:**port)-->

<property>

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

<value>qjournal://cdh-test-1-1:8485;cdh-test-1-3:8485;cdh-test-1-2:8485/tony</value>

</property>

<!-- 指定JN数据在本地磁盘的存放位置 -->

<property>

<name>dfs.journalnode.edits.dir</name>

<value>**/jn\_edits**</value>

</property>

<!-- 开启NameNode自动故障切换 -->

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

<property>

<name>dfs.ha.fencing.methods</name>

<value>

sshfence

shell(/bin/true)

</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_rsa</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.connect-timeout</name>

<value>30000</value>

</property>

<!-- 配置block副本数 -->

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

</configuration>

**mapred-site.xml**

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

**yarn-site.xml**

<configuration>

<!-- 启用RM高可用 -->

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

<!-- 自定义RM的id -->

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>tonyyarn</value>

</property>

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

</property>

<!-- 指定分配RM服务的地址 -->

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>cdh-test-1-4</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>cdh-test-1-5</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm1</name>

<value>cdh-test-1-4:8088</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm2</name>

<value>cdh-test-1-5:8088</value>

</property>

<!-- 指定zk集群地址 -->

<property>

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

<value>cdh-test-1-4:2181,cdh-test-1-5:2181,cdh-test-1-1:2181</value>

</property>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

</configuration>

**slaves**

cdh-test-1-1

cdh-test-1-3

cdh-test-1-2

【启动HDFS之前的准备】

send2others hadoop-env.sh # hadoop-env.sh添加完JAVA\_HOME之后，同步到其他节点

cp /tony\_soft/hadoop-2.8.4/bin/hdfs /usr/local/bin/hdfs

cp /tony\_soft/hadoop-2.8.4/bin/hadoop /usr/local/bin/

cd /tony\_soft/hadoop-2.8.4/bin && send2others hdfs && send2others hadoop

[root@cdh-test-1-4 hadoop]# ll /usr/local/bin/

-rwxr-xr-x 1 root root 278 4月 24 15:57 callall

-rwxr-xr-x 1 hadoop hadoop 6567 4月 24 19:44 hadoop

-rwxr-xr-x 1 hadoop hadoop 12383 4月 24 19:43 hdfs

-rwxr-xr-x 1 root root 389 4月 24 16:27 send2others

-rwxr-xr-x 1 root root 6773 4月 25 13:52 zkServer.sh

以hadoop用户启动HDFS的前提是hadoop用户能ssh免密登录到其他节点

先生成密钥对：

su hadoop

ssh-keygen -t rsa # 一路回车即可

以root用户在/home/hadoop/.ssh目录下：

callall 'mkdir -m 700 /home/hadoop/.ssh' # 批量创建权限码为700的hadoop家目录

send2others id\_rsa.pub # 同步公钥到其他hadoop家目录下

callall 'chmod 644 /home/hadoop/.ssh/authorized\_keys'

callall 'chown hadoop:hadoop /home/hadoop/.ssh/authorized\_keys'

callall 'chmod 600 /home/hadoop/.ssh/id\_rsa.pub'

分别切换到hadoop用户,将cdh-test-1-4机器上的hadoop用户下用ssh-keygen生成的

/home/hadoop/.ssh/id\_rsa.pub公钥分发到cdh-test-1-{5,1,3,2}机器上的/home/hadoop/.ssh/authorized\_keys

分发后，验证是否可免密ssh登录：for h in {5,1,3,2}; do ssh cdh-test-1-$h hostname; done

【配置zookeeper集群】

ZK三个节点：cdh-test-1-4、cdh-test-1-5、cdh-test-1-1

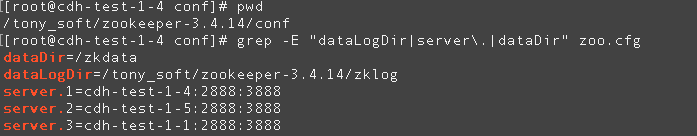
cd $ZOOKEEPER\_HOME

for i in {4,5,1}; do ssh cdh-test-1-$i mkdir -m 755 /zkdata;done

for i in {4,5,1}; do ssh cdh-test-1-$i chown -R hadoop:hadoop /zkdata; done

cp zoo\_sample.cfg zoo.cfg

vim zoo.cfg #指定部署zookeeper的3个节点; 指定zk数据存放目录和日志存放目录



for i in {3,4}; do scp conf/zoo.cfg 192.168.25.$i:/tony\_soft/zookeeper-3.4.14/conf/ ; done # 同步到其他两个节点

mkdir -m 755 zklog # 在$ZOOKEEPER\_HOME目录下创建zk日志存放目录

chown -R hadoop:hadoop zklog

# 为其他两个zk节点创建zk日志存放目录并改权限

for i in {3,4};do ssh 192.168.25.$i cd $ZOOKEEPER\_HOME && mkdir -m 755 zklog && chown -R hadoop:hadoop zklog; done

按照zoo.cfg配置的3个zk节点，分别以hadoop用户在每个节点的/zkdata目录下创建myid文件：

cdh-test-1-4机器上执行cd /zkdata && echo 1 >> myid

cdh-test-1-5机器上执行cd /zkdata && echo 2 >> myid

cdh-test-1-1机器上执行cd /zkdata && echo 3 >> myid

for h in {5,1}; do ssh cdh-test-1-$h 'cp /tony\_soft/zookeeper-3.4.14/bin/zkServer.sh /usr/local/bin/'; done

for h in {5,1}; do ssh cdh-test-1-$h 'chown hadoop:hadoop /usr/local/bin/zkServer.sh'; done

【以hadoop用户按顺序启动zk集群、启动JN集群、格式化HDFS、格式化zkfc、启动hdfs集群】

#必须用原生的**zkServer.sh启动**，如果用放在/usr/local/bin/zkServer.sh启动就会报各种错

[hadoop@cdh-test-1-4 root]$ for h in {4,5,1}; do ssh cdh-test-1-$h /tony\_soft/zookeeper/bin/zkServer.sh start; done

[hadoop@cdh-test-1-4 root]$ for h in {4,5,1}; do ssh cdh-test-1-$h /tony\_soft/zookeeper/bin/zkServer.sh status; done # 若成功启动则能看到follower或leader角色

**启动JN三个节点**：cdh-test-1-1、cdh-test-1-3、cdh-test-1-2

[hadoop@cdh-test-1-4 ~]$ for h in {1,3,2}; do ssh cdh-test-1-$h /tony\_soft/hadoop-2.8.4/sbin/hadoop-daemon.sh start journalnode; done

[hadoop@cdh-test-1-4 ~]$ callall jps # 成功启动，在1、3、2机器上可看到JournalNode进程

在4号主机上仍然**以hadoop用户格式化HDFS**：

[hadoop@cdh-test-1-4 ~]$ hdfs namenode –format # 若看到Exiting with status 0，则格式化成功

由于配了hdfs高可用，需要进行下一步：

**在4号机(nn1)启动第一个NN:**

[hadoop@cdh-test-1-4 ~]$ $HADOOP\_HOME/sbin/hadoop-daemon.sh start namenode

**在5号机(nn2)格式化HDFS:**

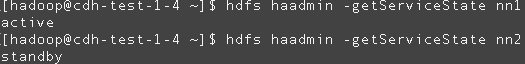
[hadoop@cdh-test-1-4 ~]$ ssh cdh-test-1-5 hdfs namenode –bootstrapStandby # 若Exiting with status 0则格式化成功

**在nn1格式化zkfc:**

[hadoop@cdh-test-1-4 ~]$ hdfs zkfc –formatZK

**在nn1启动HDFS集群：**

[hadoop@cdh-test-1-4 ~]$ $HADOOP\_HOME/sbin/start-dfs.sh # 验证结果: 和预期的一致

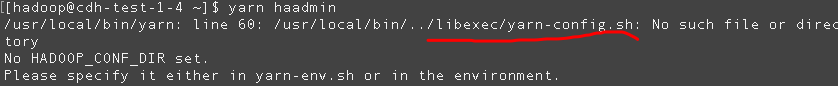


**在nn1启动YARN：**

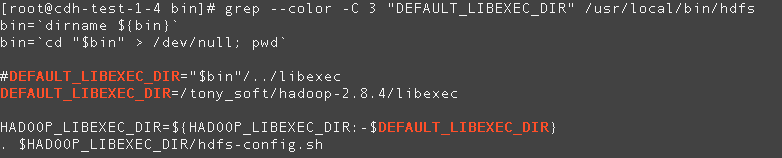
[hadoop@cdh-test-1-4 ~]$ $HADOOP\_HOME/sbin/start-yarn.sh # yarn配置了高可用,rm1和rm2要分别启动

**rm2需要单独启动：**

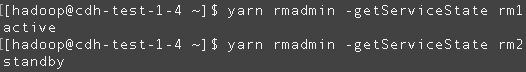
[hadoop@cdh-test-1-4 ~]$ ssh cdh-test-1-5 $HADOOP\_HOME/sbin/yarn-daemon.sh start resourcemanager



原因是缺少libexec目录下的shell脚本,可在hdfs脚本找到DEFAULT\_LIBEXEC\_DIR目录，放到yarn脚本里：



再次尝试启动rm2：ssh cdh-test-1-5 $HADOOP\_HOME/sbin/yarn-daemon.sh start resourcemanager



[hadoop@cdh-test-1-4 ~]$ callall jps

正在cdh-test-1-4 执行命令jps

Warning: Permanently added 'cdh-test-1-4,192.168.25.2' (ECDSA) to the list of known hosts.

17873 Jps

14482 NameNode

16132 ResourceManager

13179 QuorumPeerMain

15646 DFSZKFailoverController

--------命令执行成功！

正在cdh-test-1-5 执行命令jps

Warning: Permanently added 'cdh-test-1-5,192.168.25.3' (ECDSA) to the list of known hosts.

10053 NameNode

7961 QuorumPeerMain

10169 DFSZKFailoverController

10873 ResourceManager

11772 Jps

--------命令执行成功！

正在cdh-test-1-1 执行命令jps

Warning: Permanently added 'cdh-test-1-1,192.168.25.4' (ECDSA) to the list of known hosts.

11168 Jps

9616 DataNode

7735 QuorumPeerMain

8491 JournalNode

10110 NodeManager

--------命令执行成功！

正在cdh-test-1-3 执行命令jps

Warning: Permanently added 'cdh-test-1-3,192.168.25.5' (ECDSA) to the list of known hosts.

7717 JournalNode

9323 NodeManager

10379 Jps

8829 DataNode

--------命令执行成功！

正在cdh-test-1-2 执行命令jps

Warning: Permanently added 'cdh-test-1-2,192.168.25.6' (ECDSA) to the list of known hosts.

8769 DataNode

9258 NodeManager

10315 Jps

7660 JournalNode

--------命令执行成功

【配置并启动hbase集群】

[hadoop@cdh-test-1-4 hbase-1.4.9]$ hdfs dfs -mkdir -p **/tony/hbase**

hbase-env.sh 添加下列配置项

export JAVA\_HOME= /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64

export HBASE\_MANAGES\_ZK=false

# Extra Java CLASSPATH elements. Optional.

export **HBASE\_CLASSPATH**=/tony\_soft/hadoop-2.8.4/etc/hadoop

hbase-site.xml高可用配置：

<configuration>

<property>

<name>hbase.rootdir</name>

<value>hdfs://tony/hbase</value>

<description>指定hdfs上的HRegionServers共享目录(hbase数据存放地)</description>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

</property>

<property>

<name>hbase.tmp.dir</name>

<value>**/tony\_soft/hbase-1.4.9/logs**</value>

<description>指定缓存文件存储的路径</description>

</property>

<property>

<name>hbase.zookeeper.quorum</name>

<value>cdh-test-1-4:2181,cdh-test-1-5:2181,cdh-test-1-1:2181</value>

<description>指定ZooKeeper集群socket</description>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/zkdata</value>

<description>Zookeeper数据目录，与上述ZooKeeper集群相关配置一致</description>

</property>

</configuration>

regionservers

$HBASE\_HOME/conf/regionserver的内容和/tony\_soft/hadoop-2.8.4/etc/hadoop/slaves中的一致:

cdh-test-1-1

cdh-test-1-3

cdh-test-1-2

# 分发配置文件到其他节点

[root@cdh-test-1-4 conf]# send2others hbase-env.sh && send2others hbase-site.xml && send2others regionservers

启动HA-HBASE集群：

[hadoop@cdh-test-1-4 conf]$ /tony\_soft/hbase-1.4.9/bin/start-hbase.sh # 启动失败，进一步查看日志

[hadoop@cdh-test-1-4 hbase-1.4.9]$ vim /tony\_soft/hbase-1.4.9/logs/hbase-hadoop-master-cdh-test-1-4.log

Caused by: java.lang.RuntimeException: Failed to create local dir /opt/hbase-2.0.0-alpha3/tmp/local/jars, DynamicClassLoader failed to init

排错1：修改hbase-env.sh，把hadoop的配置文件所在路径添加到HBASE\_CLASSPATH

export HBASE\_CLASSPATH=/tony\_soft/hadoop-2.8.4/etc/hadoop

[root@cdh-test-1-4 conf]# send2others hbase-env.sh

再次启动，屏幕报错：

cdh-test-1-2: mkdir: cannot create directory ‘/tony\_soft/hbase-1.4.9/bin/../logs’: Permission denied

cdh-test-1-1: mkdir: cannot create directory ‘/tony\_soft/hbase-1.4.9/bin/../logs’: Permission denied

cdh-test-1-3: mkdir: cannot create directory ‘/tony\_soft/hbase-1.4.9/bin/../logs’: Permission denied

排错2：修复权限问题：

[root@cdh-test-1-4 hbase-1.4.9]# callall chown -R hadoop:hadoop /tony\_soft/hbase-1.4.9 && callall chown -R hadoop:hadoop /tony\_soft/hbase/

[hadoop@cdh-test-1-4 hbase-1.4.9]$ for i in {5,1,3,2}; do ssh cdh-test-1-$i mkdir -p /tony\_soft/hbase-1.4.9/logs; done

[hadoop@cdh-test-1-4 hbase-1.4.9]$ callall ls -l /tony\_soft/hbase-1.4.9 | grep --color logs

[hadoop@cdh-test-1-4 hbase-1.4.9]$ /tony\_soft/hbase-1.4.9/bin/start-hbase.sh # 若成功则本机能看到Hmaster服务，5号机没有Hmaster(因为需手动启动,不会自启)；另外能看到1、3、2号机上都有HregionServer服务

尝试在5号机上启动另一个Hmaster:

[hadoop@cdh-test-1-4 conf]$ ssh cdh-test-1-5 /tony\_soft/hbase/bin/hbase-daemon.sh start master # 失败

启用高可用hbase——

先关闭hbase集群

[hadoop@cdh-test-1-4 conf]$ /tony\_soft/hbase-1.4.9/bin/stop-hbase.sh

再添加另一个主机名(或IP地址)到$HBASE\_HOME/conf/**backup-masters**并同步到其他节点

[hadoop@cdh-test-1-4 conf]$ cat $HBASE\_HOME/conf/backup-master

cdh-test-1-5

[root@cdh-test-1-4 conf]# send2others backup-masters

[root@cdh-test-1-4 conf]# callall chown hadoop:hadoop /tony\_soft/hbase-1.4.9/conf/backup-masters

再次启动hbase集群

[hadoop@cdh-test-1-4 conf]$ /tony\_soft/hbase-1.4.9/bin/start-hbase.sh # 成功后可由jps看到相关进程

[hadoop@cdh-test-1-4 hbase-1.4.9]$ callall jps

正在cdh-test-1-4 执行命令jps

Warning: Permanently added 'cdh-test-1-4,192.168.25.2' (ECDSA) to the list of known hosts.

577 HMaster

14482 NameNode

16132 ResourceManager

13179 QuorumPeerMain

844 Jps

15646 DFSZKFailoverController

--------命令执行成功！

正在cdh-test-1-5 执行命令jps

Warning: Permanently added 'cdh-test-1-5,192.168.25.3' (ECDSA) to the list of known hosts.

10053 NameNode

26182 Jps

7961 QuorumPeerMain

10169 DFSZKFailoverController

10873 ResourceManager

26012 HMaster

--------命令执行成功！

正在cdh-test-1-1 执行命令jps

Warning: Permanently added 'cdh-test-1-1,192.168.25.4' (ECDSA) to the list of known hosts.

9616 DataNode

7735 QuorumPeerMain

23529 **HRegionServer**

23739 Jps

8491 JournalNode

10110 NodeManager

--------命令执行成功！

正在cdh-test-1-3 执行命令jps

Warning: Permanently added 'cdh-test-1-3,192.168.25.5' (ECDSA) to the list of known hosts.

7717 JournalNode

22695 **HRegionServer**

22904 Jps

9323 NodeManager

8829 DataNode

--------命令执行成功！

正在cdh-test-1-2 执行命令jps

Warning: Permanently added 'cdh-test-1-2,192.168.25.6' (ECDSA) to the list of known hosts.

8769 DataNode

9258 NodeManager

7660 JournalNode

22909 Jps

22686 **HRegionServer**

--------命令执行成功！

【第三章】安装Gnuplot、安装cdh-test

【安装Gnuplot】

为什么要安装gnuplot：Cdh-test依赖gnuplot 插件

[root@cdh-test-1-4 conf]# yum install –y gnuplot

版本要求：cdh-test-2.3.0及以上，我这里下载一个2.4版的rpm包，以rpm方式安装Cdh-test：

[root@cdh-test-1-4 tony\_soft]# wget

[https://github.com/Cdh-test/cdh-test/releases/download/v2.4.0/cdh-test-2.4.0.noarch.rpm](https://github.com/OpenTSDB/opentsdb/releases/download/v2.4.0/opentsdb-2.4.0.noarch.rpm)

**# 开始分布式安装cdh-test**

分发到其他节点send2others cdh-test-2.4.0.noarch.rpm

然后**每个节点安装**即可callall rpm –ivh cdh-test-2.4.0.noarch.rpm # 默认会安装到 /usr/share/ 下

cdh-test本身没有分布式的实现方案，而是借助于HBase的分布式集群方案。也就是说cdh-test访问同一个HBase集群，返回相同的数据镜像。TSDMain是cdh-test的进程名。

vim /etc/cdh-test/cdh-test.conf

[root@cdh-test-1-4 cdh-test]# egrep --color "^tsd" /etc/cdh-test/cdh-test.conf

tsd.network.port = 4040 # 自定义HTTP端口号

tsd.http.staticroot = /usr/share/cdh-test/static/ # 使用默认值即可

tsd.http.cachedir = /tmp/cdh-test # 配置cdh-test的缓存目录

# false改为true那么上传数据时会自动创建metric，否则会提示Unknown metric 的错误。也可设置为false但使用tsdb mkmetric proc.loadavg.1m来手动添加 metric

tsd.core.auto\_create\_metrics = true

tsd.core.plugin\_path = /usr/share/cdh-test/plugins # 使用默认cdh-test的插件所在路径

tsd.storage.hbase.zk\_quorum = cdh-test-1-4,cdh-test-1-5,cdh-test-1-1 # 配置zk三个主机名

# 启动cdh-test之前，需要在Hbase中创建cdh-test所需的表

[root@cdh-test-1-4 ~]# cd /usr/share/cdh-test/

[root@cdh-test-1-4 cdh-test]# env COMPRESSION=NONE HBASE\_HOME=/tony\_soft/hbase-1.4.9 ./tools/**create\_table.sh** > /tmp/creatingForTbls4cdh-test.log 2>&1 &

# 脚本会自动在default名称空间下创建4个表tsdb-uid,tsdb,tsdb-tree,tsdb-meta

# 脚本运行失败，查看日志发现在创建default:tsdb表时有错误

create 'tsdb',

{NAME => 't', VERSIONS => 1, COMPRESSION => 'NONE', BLOOMFILTER => 'ROW', DATA\_BLOCK\_ENCODING => 'DIFF', TTL => 'FOREVER'}

ERROR: For input string: "FOREVER"

# 排错：建表语句不被hbase认可(我安装的是hbase-1.4.9版本)，那么去掉FOREVER或者修改其值为2147483647

[root@cdh-test-1-4 ~]# grep -C 3 --color "TSDB\_TTL" /usr/share/cdh-test/tools/create\_table.sh

# This can save a lot of storage space.

DATA\_BLOCK\_ENCODING=${DATA\_BLOCK\_ENCODING-'DIFF'}

DATA\_BLOCK\_ENCODING=`echo "$DATA\_BLOCK\_ENCODING" | tr a-z A-Z`

TSDB\_TTL=${TSDB\_TTL-'FOREVER'}

case $COMPRESSION in

(NONE|LZO|GZIP|SNAPPY) :;; # Known good.

--

{NAME => 'name', COMPRESSION => '$COMPRESSION', BLOOMFILTER => '$BLOOMFILTER', DATA\_BLOCK\_ENCODING => '$DATA\_BLOCK\_ENCODING'}

create '$TSDB\_TABLE',

{NAME => 't', VERSIONS => 1, COMPRESSION => '$COMPRESSION', BLOOMFILTER => '$BLOOMFILTER', DATA\_BLOCK\_ENCODING => '$DATA\_BLOCK\_ENCODING', TTL => '$TSDB\_TTL'}

create '$TREE\_TABLE',

{NAME => 't', VERSIONS => 1, COMPRESSION => '$COMPRESSION', BLOOMFILTER => '$BLOOMFILTER', DATA\_BLOCK\_ENCODING => '$DATA\_BLOCK\_ENCODING'}

# 排错之后重试

[root@cdh-test-1-4 tools]# cd /usr/share/cdh-test/tools && vim create\_table.sh

[root@cdh-test-1-4 tools]# grep -C 1 --color "TSDB\_TTL" create\_table.sh

DATA\_BLOCK\_ENCODING=`echo "$DATA\_BLOCK\_ENCODING" | tr a-z A-Z`

#TSDB\_TTL=${TSDB\_TTL-'FOREVER'}

TSDB\_TTL=${TSDB\_TTL-'2147483647'}

--

create '$TSDB\_TABLE',

{NAME => 't', VERSIONS => 1, COMPRESSION => '$COMPRESSION', BLOOMFILTER => '$BLOOMFILTER', DATA\_BLOCK\_ENCODING => '$DATA\_BLOCK\_ENCODING', TTL => '$TSDB\_TTL'}

[root@cdh-test-1-4 tools]# send2others create\_table.sh

[root@cdh-test-1-4 cdh-test]# pwd

/usr/share/cdh-test

[root@cdh-test-1-4 cdh-test]# env COMPRESSION=NONE HBASE\_HOME=/tony\_soft/hbase-1.4.9 ./tools/create\_table.sh > /tmp/re\_creatingForTbls4cdh-test.log 2>&1 &

hbase(main):009:0> list **# hbase shell中可看到，成功建表**

TABLE

tsdb

tsdb-meta

tsdb-tree

tsdb-uid

4 row(s) in 0.0080 seconds

=> ["tsdb", "tsdb-meta", "tsdb-tree", "tsdb-uid"]

# 启动cdh-test集群(也可以仅启动其中1、2台)

[root@cdh-test-1-4 bin]# callall "/usr/share/cdh-test/bin/tsdb tsd > /tmp/tsd\_running.log 2>&1 &"

[root@cdh-test-1-4 bin]# callall "jps | grep -v grep | grep -i tsdmain" # 可在所有节点看到TSDMain进程

[root@cdh-test-1-4 bin]# callall "netstat -nltp | grep -i 4242" # 可看到4、5、1、3、2号节点都有对应的4242端口号

# 使用python脚本插入示例数据

[root@cdh-test-1-4 tmp]# cat cdh-test\_test.py

import requests

payload = {

"metric": "sys.cpu.nice",

"timestamp": '1519544801',

"value": '29',

"tags": {

"host": "web01",

"dc": "lga"

}

}

payload1 = {

"metric": "sys.cpu.nice",

"timestamp": '1519544802',

"value": '30',

"tags": {

"host": "web01",

"dc": "lga"

}

}

payload2 = {

"metric": "sys.cpu.nice",

"timestamp": '1519544803',

"value": '29',

"tags": {

"host": "web01",

"dc": "lga"

}

}

payload3 = {

"metric": "sys.cpu.nice",

"timestamp": '1519544804',

"value": '30',

"tags": {

"host": "web01",

"dc": "lga"

}

}

ls = [payload, payload1, payload2, payload3]

def send\_json(json):

# 向主机cdh-test-1-5写入数据(因为该节点也有TSDMain)

r = requests.post("http://cdh-test-1-5:4242/api/put?details", json=json)

return r.text

def main():

print send\_json(ls)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# tsd返回成功，表明成功写入数据到hbase

[root@cdh-test-1-4 tmp]# python cdh-test\_test.py

{"success":4,"failed":0,"errors":[]}

# 到4、5、1其中一台主机(这三台有zk服务QuorumPeerMain)上的hbase shell检查是否已插入数据

[root@cdh-test-1-4 ~]# ssh cdh-test-1-1

hbase(main):001:0> list

TABLE

tsdb

tsdb-meta

tsdb-tree

tsdb-uid

4 row(s) in 0.1660 seconds

=> ["tsdb", "tsdb-meta", "tsdb-tree", "tsdb-uid"]

hbase(main):002:0> scan 'default:tsdb' # 有数据

hbase(main):003:0> scan 'default:tsdb-meta' # 没数据

hbase(main):004:0> scan 'default:tsdb-tree' # 没数据

hbase(main):005:0> scan 'default:tsdb-uid' # 有数据

以下是关于cdh-test的web restFul api安全使用的整理。这里是基于nginx做用户验证(也就是用户名+密码的方式)，是为了(云主机)内网用户在使用tsd的api过程中做个身份验证，只允许内网指定用户调用cdh-test的api。

【第四章】通过Nginx对Cdh-test做用户验证，确保cdh-test http接口安全

CentOS7安装Nginx可参考: <https://www.cnblogs.com/kaid/p/7640723.html>

Nginx 配置用户名密码登录可参考: <https://my.oschina.net/AmosWang/blog/2873864>

【step 1】

cdh-test主要配置项：



批量启动TSDMain(即tsd服务)：callall "/usr/share/cdh-test/bin/tsdb tsd > /tmp/tsd\_running.log 2>&1 &"

【step 2】

安装Nginx过程省略。

安装htpasswd工具(生成用户名和密码并保存到文件)

在/usr/local/nginx/conf/nginx.conf的http代码块里的子块加入相关配置项：

server {

# 此处监听的端口号4242供外网访问

listen 4242;

auth\_basic "Password please";

#密码地址

auth\_basic\_user\_file /usr/local/nginx/conf/ngpwd;

location / {

#代理地址：192.168.x.x是VPC内网,4040是cdh-test在内网的端口；外网需访问10.39.46.49:4242

proxy\_pass http://192.168.25.2:4040/;

proxy\_redirect default;

}

}

启动Nginx服务(略)。

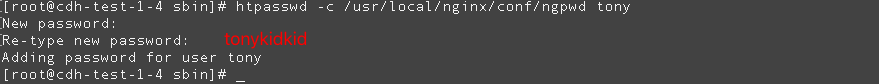
在4号机使用htpasswd工具生成密码文件：

htpasswd -c /usr/local/soft/nginx/conf/pwd tony 接着输入两次密码即可

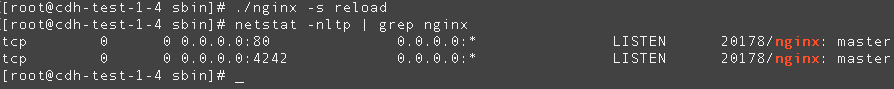
#htpasswd -c：意思是创建一个加密文件

#/usr/local/nginx/conf/ngpwd：表示加密文件的绝对路径(与上述nginx.conf中的auth\_basic\_user\_file一致)

# tony：用户名（密码tonykidkid）

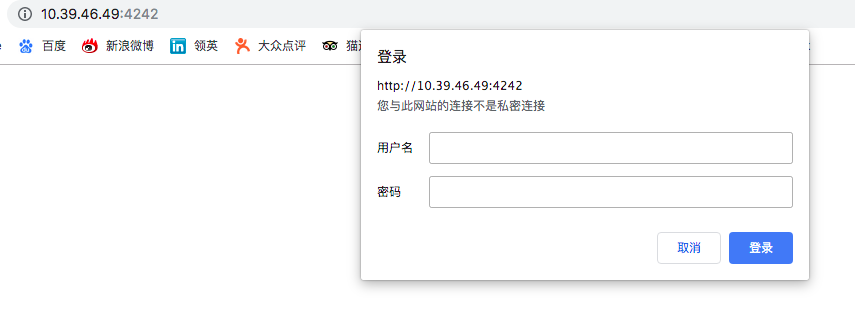


生成用户密码后，重启nginx服务使配置生效

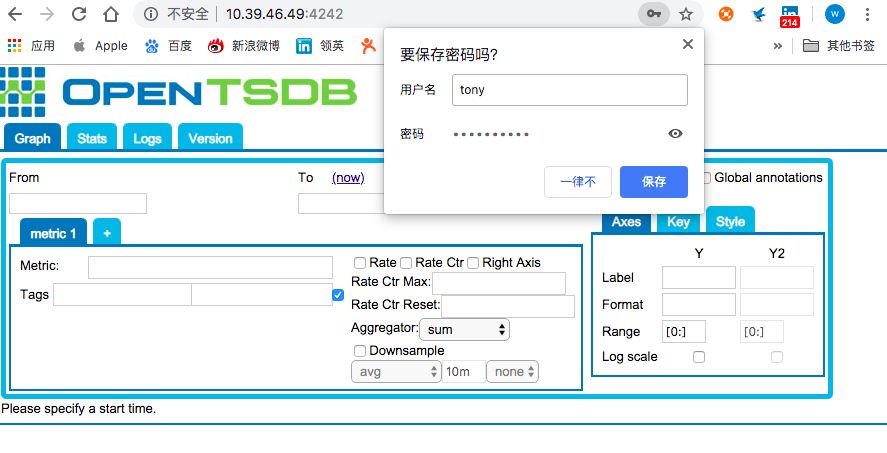


验证用户认证功能：

（1）从外网浏览器访问<http://10.39.46.49:4242/>



此时需要输入密码才能继续访问：

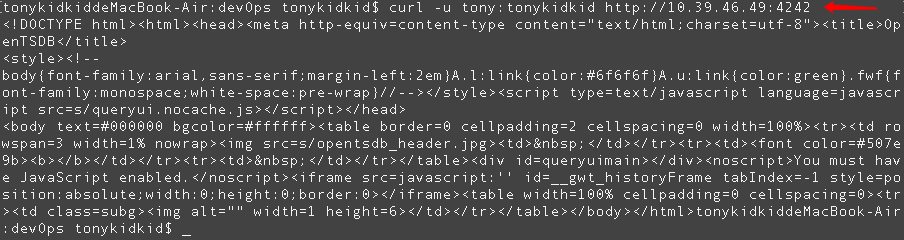


（2）从外网采用curl <http://10.39.46.49:4242> 未发送认证信息访问cdh-test时，HTTP返回码401：



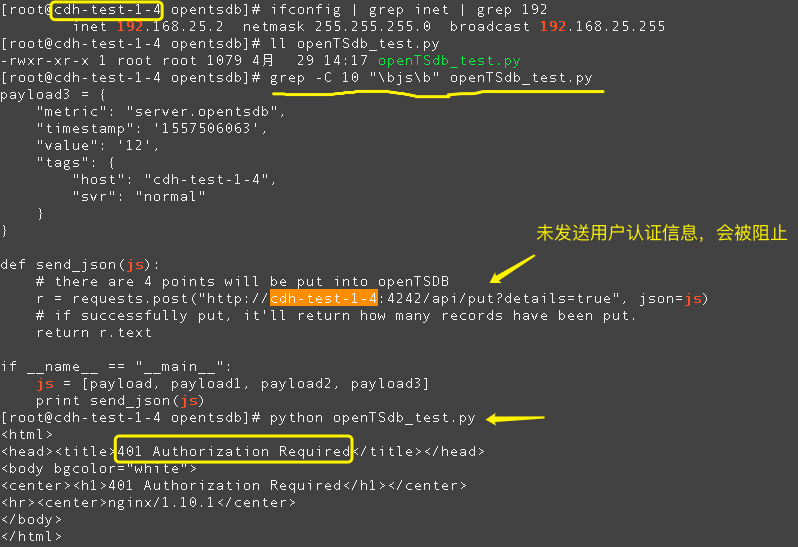
验证成功。

从外网curl -u tony:tonykidkid <http://10.39.46.49:4242> 发送用户认证信息访问cdh-test时，正常返回：



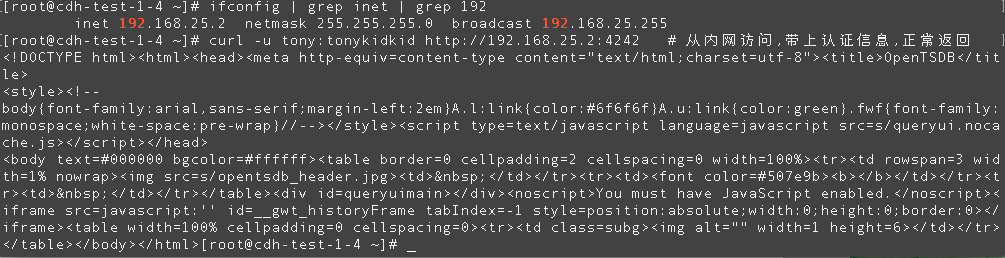
验证成功。

（3）从VPC内网采用客户端脚本向cdh-test put数据，HTTP返回码401：



验证成功。

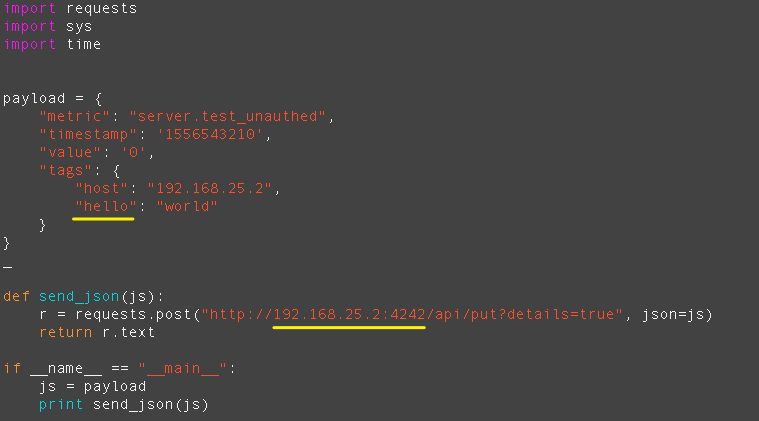
（4）从内网访问,带上认证信息,正常返回：



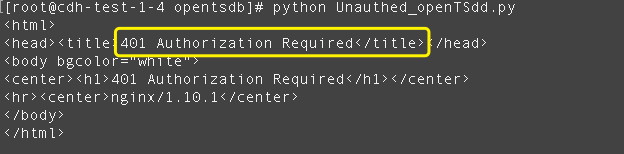
验证成功。

（5）使用客户端脚本向<http://192.168.25.2:4242/api/put?details=true> 写入单点数据

python脚本Unauthed\_openTSdd.py源代码：



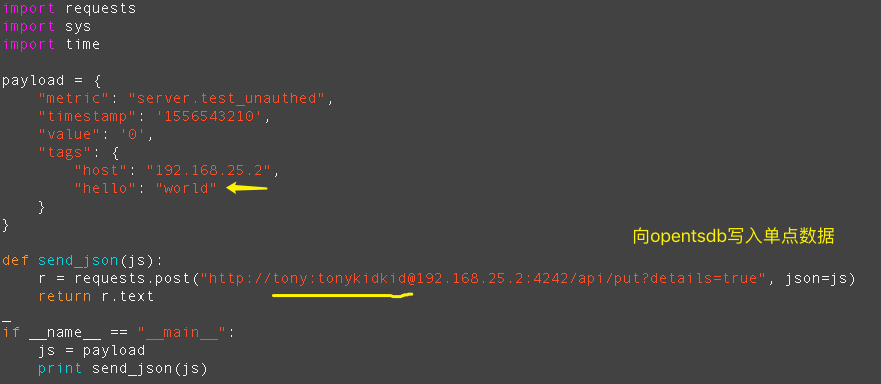
未带认证信息时，返回401：



验证成功。

（6）使用客户端脚本以 <http://username:password@ip:port/api> 方式传入认证信息

python脚本authed\_cdh-test\_test.py源码：



认证通过，成功写入数据：



在hbase可查到：



验证成功。

【全文结束】读者觉得有什么要补充的，或者哪里不明白的可以给我留言，觉得没问题的可以点个赞 ^\_^