## Kerberos分布式安装部署

### 1安装rpm安装包

在server端（主节点）安装 krb5-libs、krb5-devel、krb5-server、krb5-workstation

[root@hadoop02 kerberos]# yum -y install krb5-libs krb5-devel krb5-server krb5-workstation

在client端（从节点）安装 krb5-libs、krb5-workstation

[root@hadoop02 ~]# yum -y install krb5-libs krb5-workstation

### 2配置Kerberos

需要配置的文件有两个为kdc.conf和krb5.conf,配置只是需要Server服务节点配置，即hadoop02主节点

1）kdc配置

[root@hadoop02 kerberos]# cd /var/kerberos/krb5kdc/

[root@hadoop02 krb5kdc]# vim kdc.conf

按如下内容修改

[kdcdefaults]

kdc\_ports = 88

kdc\_tcp\_ports = 88

[realms]

HADOOP.COM = {

#master\_key\_type = aes256-cts

acl\_file = /var/kerberos/krb5kdc/kadm5.acl

dict\_file = /usr/share/dict/words

admin\_keytab = /var/kerberos/krb5kdc/kadm5.keytab

#删除aes256-cts:normal

supported\_enctypes = aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal

}

1. krb5配置**（所有节点都要配置）**

[root@hadoop02 ~]# vim /etc/krb5.conf

按照如下内容修改

# Configuration snippets may be placed in this directory as well

includedir /etc/krb5.conf.d/

# Configuration snippets may be placed in this directory as well

includedir /etc/krb5.conf.d/

[logging]

default = FILE:/var/log/krb5libs.log

kdc = FILE:/var/log/krb5kdc.log

admin\_server = FILE:/var/log/kadmind.log

[libdefaults]

dns\_lookup\_realm = false

#dns需要网络

dns\_lookup\_kdc = false

ticket\_lifetime = 24h

renew\_lifetime = 7d

forwardable = true

rdns = false

pkinit\_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt

udp\_preference\_limit = 1

# default\_realm = HADOOP.COM

default\_realm = HADOOP.COM

#default\_ccache\_name = KEYRING:persistent:%{uid}

[realms]

# HADOOP.COM = {

# kdc = kerberos.HADOOP.COM

# admin\_server = kerberos.HADOOP.COM

# }

HADOOP.COM = {

kdc = hadoop02

admin\_server = hadoop02

}

[domain\_realm]

# .HADOOP.COM = HADOOP.COM

# HADOOP.COM = HADOOP.COM

.HADOOP.COM = HADOOP.COM

HADOOP.COM = HADOOP.COM

### 3 生成Kerberos数据库

设置密码为：123456

[root@hadoop02 ~]# kdb5\_util create -r HADOOP.COM -s

输入密码

确认密码

创建完成后/var/kerberos/krb5kdc 目录下会生成对应的文件

[root@hadoop02 ~]# ls /var/kerberos/krb5kdc/

kadm5.acl kdc.conf principal principal.kadm5 principal.kadm5.lock principal.ok

### 4 创建管理员主体/实例

设置密码为：123456

[root@hadoop02 ~]# kadmin.local -q "addprinc admin/admin"

Authenticating as principal root/admin@HADOOP.COM with password.

WARNING: no policy specified for admin/admin@HADOOP.COM; defaulting to no policy

Enter password for principal "admin/admin@HADOOP.COM":输入密码

Re-enter password for principal "admin/admin@HADOOP.COM": 确认密码

Principal "admin/admin@HADOOP.COM" created.

### 5 给管理员实例的所有主体授权

[root@hadoop02 ~]# vim /var/kerberos/krb5kdc/kadm5.acl

按如下内容修改

\*/admin@HADOOP.COM \*

### 6 启动服务

1）启动krb5kdc

[root@hadoop02 krb5kdc]# systemctl start krb5kdc

2）启动kadmin

[root@hadoop02 krb5kdc]# systemctl start kadmin

3）设置krb5kdc和kadmin开机自启

[root@hadoop02 krb5kdc]# systemctl enable kadmin

[root@hadoop02 krb5kdc]# systemctl enable krb5kdc

4）查看设置是否生效

[root@hadoop02 krb5kdc]# systemctl is-enabled krb5kdc

enabled

[root@hadoop02 krb5kdc]# systemctl is-enabled kadmin

enabled

### 7 kinit管理员验证

1）主节点验证

**主节点hadoop02**

[root@hadoop02 krb5kdc]# kinit admin/admin

Password for admin/admin@HADOOP.COM: （输入密码）

[root@hadoop02 module]# klist

Ticket cache: FILE:/tmp/krb5cc\_0

Default principal: admin/admin@HADOOP.COM

Valid starting Expires Service principal

11/02/2022 13:29:24 11/03/2022 13:29:24 krbtgt/HADOOP.COM@HADOOP.COM

2）从节点验证

**从节点hadoop01**

[root@hadoop01 kerberos]# kinit admin/admin

Password for admin/admin@HADOOP.COM: （输入密码）

[root@hadoop01 kerberos]# klist

Ticket cache: FILE:/tmp/krb5cc\_0

Default principal: admin/admin@HADOOP.COM

Valid starting Expires Service principal

11/02/2022 14:04:52 11/03/2022 14:04:52 krbtgt/HADOOP.COM@HADOOP.COM

[root@hadoop01 kerberos]#

**从节点hadoop03**

[root@hadoop03 kerberos]# kinit admin/admin

Password for admin/admin@HADOOP.COM: （输入密码）

[root@hadoop03 kerberos]# klist

Ticket cache: FILE:/tmp/krb5cc\_0

Default principal: admin/admin@HADOOP.COM

Valid starting Expires Service principal

11/02/2022 14:02:26 11/03/2022 14:02:26 krbtgt/HADOOP.COM@HADOOP.COM

### 8 kerberos常用命令

1）销毁凭证

[root@hadoop03 temp]# kdestroy

You have new mail in /var/spool/mail/root

[root@hadoop03 temp]# klist

klist: No credentials cache found (filename: /tmp/krb5cc\_0)

、 2）创建kerberos主体

[root@hadoop02 opt]# kadmin.local -q "addprinc test"

Authenticating as principal admin/admin@HADOOP.COM with password.

WARNING: no policy specified for test@HADOOP.COM; defaulting to no policy

Enter password for principal "test@HADOOP.COM": (输入密码)

Re-enter password for principal "test@HADOOP.COM": (输入密码)

Principal "test@HADOOP.COM" created.

3）修改主体密码

[root@hadoop02 opt]# kadmin.local -q "cpw test"

Authenticating as principal admin/admin@HADOOP.COM with password.

Enter password for principal "test@HADOOP.COM": (输入密码)

Re-enter password for principal "test@HADOOP.COM": (输入密码)

**Password for "test@HADOOP.COM" changed**.

4）查看所有主体

[root@hadoop02 opt]# kadmin.local -q "list\_principals"

Authenticating as principal test/admin@HADOOP.COM with password.

K/M@HADOOP.COM

admin/admin@HADOOP.COM

kadmin/admin@HADOOP.COM

kadmin/changepw@HADOOP.COM

kadmin/hadoop02@HADOOP.COM

kiprop/hadoop02@HADOOP.COM

krbtgt/HADOOP.COM@HADOOP.COM

root/admin@HADOOP.COM

root@HADOOP.COM

test@HADOOP.COM

5）查看密钥文件主体

[root@hadoop02 opt]# klist -kte /opt/root.keytab

Keytab name: FILE:/opt/root.keytab

KVNO Timestamp Principal

---- ------------------- ------------------------------------------------------

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (aes128-cts-hmac-sha1-96)

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (des3-cbc-sha1)

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (arcfour-hmac)

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (camellia256-cts-cmac)

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (camellia128-cts-cmac)

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (des-hmac-sha1)

2 11/02/2022 14:19:44 root/admin@HADOOP.COM (des-cbc-md5)

[root@hadoop02 opt]#

6）删除用户主体

[root@hadoop02 opt]# kadmin.local -q "delprinc test"

### 9 使用keytab密钥文件登录

1）在主节点生成电子凭证

[root@hadoop02 opt]# kadmin.local -q "xst -k /opt/root.keytab root/admin@HADOOP.COM"

Authenticating as principal admin/admin@HADOOP.COM with password.

Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type aes128-cts-hmac-sha1-96 added to keytab WRFILE:/opt/root.keytab.

Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type des3-cbc-sha1 added to keytab WRFILE:/opt/root.keytab.

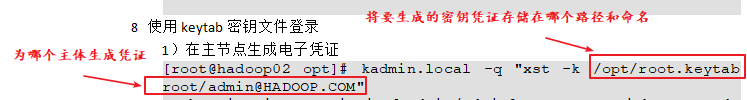
Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type arcfour-hmac added to keytab WRFILE:/opt/root.keytab.

Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type camellia256-cts-cmac added to keytab WRFILE:/opt/root.keytab.

Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type camellia128-cts-cmac added to keytab WRFILE:/opt/root.keytab.

Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type des-hmac-sha1 added to keytab WRFILE:/opt/root.keytab.

Entry for principal root/admin@HADOOP.COM with kvno 2, encryption type des-cbc-md5 added to keytab WRFILE:/opt/root.keytab.



2）转发到从节点

[root@hadoop02 opt]# scp /opt/root.keytab root@hadoop03:/opt/

root.keytab 100% 458 450.0KB/s 00:00

3）使用密钥文件进行认证测试

[root@hadoop03 temp]# kinit -kt /opt/root.keytab root/admin

[root@hadoop03 temp]# klist

Ticket cache: FILE:/tmp/krb5cc\_0

Default principal: root/admin@HADOOP.COM

Valid starting Expires Service principal

11/02/2022 15:00:18 11/03/2022 15:00:18 krbtgt/HADOOP.COM@HADOOP.COM

## Zookeeper开启kerberos认证

### 1 生成Zookeeper的keytab证书

创建统一存放keytab的目录

[root@hadoop02 opt]# mkdir /etc/security/keytab

创建zookeeper所需keytab证书

[root@hadoop01 keytab]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/zookeeper.keytab zookeeper/hadoop01"

[root@hadoop02 keytab]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/zookeeper.keytab zookeeper/hadoop02"

[root@hadoop03 keytab]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/zookeeper.keytab zookeeper/hadoop03"

### 2 修改zoo.cfg配置文件

修改/opt/module/zookeeper-3.5.7/conf/目录下的zoo.cfg配置文件，在原有配置文件的末尾添加如下内容：

[root@hadoop01 ~]# cd /opt/module/zookeeper-3.5.7/conf

[root@hadoop01 conf]# vim zoo.cfg

#添加如下内容

#kerberos认证配置

authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider

jaasLoginRenew=3600000

sessionRequireClientSASLAuth=true

### 3创建 jaas.conf配置文件

在/opt/module/zookeeper-3.5.7/conf/目录下创建jaas.conf配置文件

[root@hadoop01 conf]# vim jaas.conf

#添加如下内容

#标红部分修改为所在节点的hostname

Server {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/etc/security/keytab/zookeeper.keytab"

storeKey=true

useTicketCache=false

principal="zookeeper/hadoop01@HADOOP.COM";

};

Client {

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

keyTab="/etc/security/keytab/zookeeper.keytab"

storeKey=true

useTicketCache=false

principal="zookeeper/hadoop01@HADOOP.COM";

};

### 4 创建java.env配置文件

在/opt/module/zookeeper-3.5.7/conf/目录下创建java.env配置文件

[root@hadoop01 conf]# vim java.env

#添加如下内容

#标红部分修改为所在节点的hostname

export SERVER\_JVMFLAGS="-Djava.security.auth.login.config=/opt/test/zookeeper-3.7.1/conf/jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf"

export CLIENT\_JVMFLAGS="${CLIENT\_JVMFLAGS} -Djava.security.auth.login.config=/opt/test/zookeeper-3.7.1/conf/jaas.conf -Djava.security.krb5.conf=/etc/krb5.conf -Dzookeeper.server.principal=zookeeper/hadoop01@HADOOP.COM"

### 5分发配置文件

[root@hadoop02 ~]# cd /opt/module/zookeeper-3.5.7/conf

[root@hadoop01 conf]# xrysrc java.env

[root@hadoop01 conf]# xrysrc jass.conf

## Hadoop HA开启kerberos认证

### 1 创建Hadoop系统用户

为Hadoop开启Kerberos，需为不同服务准备不同的用户，启动服务时需要使用相应的用户。须在**所有节点**创建以下用户和用户组。

|  |  |
| --- | --- |
| **User:Group** | **Daemons** |
| **hdfs:hadoop** | NameNode, Secondary NameNode, JournalNode, DataNode |
| **yarn:hadoop** | ResourceManager, NodeManager |
| **mapred:hadoop** | MapReduce JobHistory Server |

创建hadoop组

[root@hadoop01 ~]# groupadd hadoop

[root@hadoop02 ~]# groupadd hadoop

[root@hadoop03 ~]# groupadd hadoop

创建各用户并设置密码

[root@hadoop01 ~]# useradd hdfs -g hadoop

[root@hadoop01 ~]# echo hdfs | passwd --stdin hdfs

[root@hadoop01 ~]# useradd yarn -g hadoop

[root@hadoop01 ~]# echo yarn | passwd --stdin yarn

[root@hadoop01 ~]# useradd mapred -g hadoop

[root@hadoop01 ~]# echo mapred | passwd --stdin mapred

[root@hadoop02 ~]# useradd hdfs -g hadoop

[root@hadoop02 ~]# echo hdfs | passwd --stdin hdfs

[root@hadoop02 ~]# useradd yarn -g hadoop

[root@hadoop02 ~]# echo yarn | passwd --stdin yarn

[root@hadoop02 ~]# useradd mapred -g hadoop

[root@hadoop02 ~]# echo mapred | passwd --stdin mapred

[root@hadoop03 ~]# useradd hdfs -g hadoop

[root@hadoop03 ~]# echo hdfs | passwd --stdin hdfs

[root@hadoop03 ~]# useradd yarn -g hadoop

[root@hadoop03 ~]# echo yarn | passwd --stdin yarn

[root@hadoop03 ~]# useradd mapred -g hadoop

[root@hadoop03 ~]# echo mapred | passwd --stdin mapred

### 2 为Hadoop各服务创建Kerberos主体（Principal）

主体格式如下：ServiceName/HostName@REALM，例如 dn/hadoop01@HADOOP.COM

**①各服务所需主体如下**

环境：3台节点，主机名分别为hadoop01，hadoop02，hadoop03

|  |  |  |
| --- | --- | --- |
| **服务** | **所在主机** | **主体（Principal）** |
| **NameNode1** | hadoop01 | nn/hadoop01 |
| **NameNode2** | hadoop02 | nn/hadoop02 |
| **DataNode** | hadoop01 | dn/hadoop01 |
| **DataNode** | hadoop02 | dn/hadoop02 |
| **DataNode** | hadoop03 | dn/hadoop03 |
| **Secondary NameNode** | hadoop03 | sn/hadoop03 |
| **ResourceManager1** | hadoop02 | rm/hadoop02 |
| **ResourceManager2** | hadoop01 | rm/hadoop01 |
| **NodeManager** | hadoop01 | nm/hadoop01 |
| **NodeManager** | hadoop02 | nm/hadoop02 |
| **NodeManager** | hadoop03 | nm/hadoop03 |
| **JobHistory Server** | hadoop01 | jhs/hadoop01 |
| **Web UI** | hadoop01 | HTTP/hadoop01 |
| **Web UI** | hadoop02 | HTTP/hadoop02 |
| **Web UI** | hadoop03 | HTTP/hadoop03 |
| **JournalNode** | hadoop01 | jan/hadoop01 |
| **JournalNode** | hadoop02 | jan/hadoop02 |
| **DFSZKFailoverController** | hadoop01 | zkfc/hadoop01 |
| **DFSZKFailoverController** | hadoop02 | zkfc/hadoop02 |

**②创建主体说明**

1）路径准备

为服务创建的主体，需要通过密钥文件keytab文件进行认证，故需为各服务准备一个安全的路径用来存储keytab文件。

[root@hadoop01 ~]# mkdir /etc/security/keytab/

[root@hadoop01 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop01 ~]# chmod 770 /etc/security/keytab/

2）管理员主体认证

为执行创建主体的语句，需登录Kerberos 数据库客户端，登录之前需先使用Kerberos的管理员用户进行认证，执行以下命令并根据提示输入密码。

[root@hadoop01 ~]# kinit admin/admin

3）登录数据库客户端

[root@hadoop01 ~]# kadmin

4）执行创建主体的语句

kadmin: addprinc -randkey test/test

kadmin: xst -k /etc/security/keytab/test.keytab test/test

说明：

（1）addprinc test/test：作用是新建主体

addprinc：增加主体

-randkey：密码随机，因hadoop各服务均通过keytab文件认证，故密码可随机生成

test/test：新增的主体

（2）xst -k /etc/security/keytab/test.keytab test/test：作用是将主体的密钥写入keytab文件

xst：将主体的密钥写入keytab文件

-k /etc/security/keytab/test.keytab：指明keytab文件路径和文件名

test/test：主体

（3）为方便创建主体，可使用如下命令

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey test/test"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/test.keytab test/test"

说明：

-p：主体

-w：密码

-q：执行语句

**③创建主体**

**1）在所有节点创建keytab文件目录**

[root@hadoop01 ~]# mkdir /etc/security/keytab/

[root@hadoop01 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop01 ~]# chmod 770 /etc/security/keytab/

[root@hadoop02 ~]# mkdir /etc/security/keytab/

[root@hadoop02 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop02 ~]# chmod 770 /etc/security/keytab/

[root@hadoop03 ~]# mkdir /etc/security/keytab/

[root@hadoop03 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop03 ~]# chmod 770 /etc/security/keytab/

**2）以下命令在hadoop01节点执行**

**NameNode（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey nn/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/nn.service.keytab nn/hadoop01"

**DataNode（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey dn/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/dn.service.keytab dn/hadoop01"

**NodeManager（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey nm/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/nm.service.keytab nm/hadoop01"

**JobHistory Server（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey jhs/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/jhs.service.keytab jhs/hadoop01"

**Web UI（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey HTTP/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/spnego.service.keytab HTTP/hadoop01"

**ResourceManager（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey rm/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/rm.service.keytab rm/hadoop01"

**JournalNode（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey jan/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/jan.service.keytab jan/hadoop01"

**DFSZKFailoverController（hadoop01）**

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey zkfc/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/zkfc.service.keytab zkfc/hadoop01"

**2）以下命令在hadoop02执行**

**ResourceManager（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey rm/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/rm.service.keytab rm/hadoop02"

**DataNode（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey dn/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/dn.service.keytab dn/hadoop02"

**NodeManager（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey nm/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/nm.service.keytab nm/hadoop02"

**Web UI（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey HTTP/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/spnego.service.keytab HTTP/hadoop02"

**NameNode（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey nn/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/nn.service.keytab nn/hadoop02"

**JournalNode（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey jan/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/jan.service.keytab jan/hadoop02"

**DFSZKFailoverController（hadoop02）**

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey zkfc/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/zkfc.service.keytab zkfc/hadoop02"

**3）以下命令在hadoop03执行**

**DataNode（hadoop03）**

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey dn/hadoop03"

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/dn.service.keytab dn/hadoop03"

**Secondary NameNode（hadoop03）**

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey sn/hadoop03"

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/sn.service.keytab sn/hadoop03"

**NodeManager（hadoop03）**

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey nm/hadoop03"

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/nm.service.keytab nm/hadoop03"

**Web UI（hadoop03）**

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey HTTP/hadoop03"

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/spnego.service.keytab HTTP/hadoop03"

**④修改所有节点keytab文件的所有者和访问权限**

[root@hadoop01 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop01 ~]# chmod 660 /etc/security/keytab/\*

[root@hadoop02 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop02 ~]# chmod 660 /etc/security/keytab/\*

[root@hadoop03 ~]# chown -R root:hadoop /etc/security/keytab/

[root@hadoop03 ~]# chmod 660 /etc/security/keytab/\*

### 3修改Hadoop配置文件

**需要修改的内容如下，修改完毕需要分发所改文件。**

**1）修改core-site.xml**

[root@hadoop01 ~]# vim /opt/module/hadoop-3.1.3/etc/hadoop/core-site.xml

增加以下内容

<!-- Kerberos主体到系统用户的映射机制 -->

<property>

<name>hadoop.security.auth\_to\_local.mechanism</name>

<value>MIT</value>

</property>

<!-- Kerberos主体到系统用户的具体映射规则 -->

<property>

<name>hadoop.security.auth\_to\_local</name>

<value>

RULE:[2:$1/$2@$0]([ndj]n\/.\*@HADOOP\.COM)s/.\*/hdfs/

RULE:[2:$1/$2@$0]([rn]m\/.\*@HADOOP\.COM)s/.\*/yarn/

RULE:[2:$1/$2@$0](jhs\/.\*@HADOOP\.COM)s/.\*/mapred/

DEFAULT

</value>

</property>

<!-- 启用Hadoop集群Kerberos安全认证 -->

<property>

<name>hadoop.security.authentication</name>

<value>kerberos</value>

</property>

<!-- 启用Hadoop集群授权管理 -->

<property>

<name>hadoop.security.authorization</name>

<value>true</value>

</property>

<!-- Hadoop集群间RPC通讯设为仅认证模式 -->

<property>

<name>hadoop.rpc.protection</name>

<value>authentication</value>

</property>

**2）修改hdfs-site.xml**

[root@hadoop01 ~]# vim /opt/module/hadoop-3.1.3/etc/hadoop/hdfs-site.xml

增加以下内容

<!-- 访问DataNode数据块时需通过Kerberos认证 -->

<property>

<name>dfs.block.access.token.enable</name>

<value>true</value>

</property>

<!-- NameNode服务的Kerberos主体,\_HOST会自动解析为服务所在的主机名 -->

<property>

<name>dfs.namenode.kerberos.principal</name>

<value>nn/\_HOST@HADOOP.COM</value>

</property>

<!-- NameNode服务的Kerberos密钥文件路径 -->

<property>

<name>dfs.namenode.keytab.file</name>

<value>/etc/security/keytab/nn.service.keytab</value>

</property>

<!-- JournalNode服务的Kerberos主体,\_HOST会自动解析为服务所在的主机名 -->

<property>

<name>dfs.journalnode.kerberos.principal</name>

<value>jan/\_HOST@HADOOP.COM</value>

</property>

<!-- JournalNode服务的Kerberos密钥文件路径 -->

<property>

<name>dfs.journalnode.keytab.file</name>

<value>/etc/security/keytab/jan.service.keytab</value>

</property>

<!-- Secondary NameNode服务的Kerberos主体 -->

<property>

<name>dfs.secondary.namenode.keytab.file</name>

<value>/etc/security/keytab/sn.service.keytab</value>

</property>

<!-- Secondary NameNode服务的Kerberos密钥文件路径 -->

<property>

<name>dfs.secondary.namenode.kerberos.principal</name>

<value>sn/\_HOST@HADOOP.COM</value>

</property>

<!-- NameNode Web服务的Kerberos主体 -->

<property>

<name>dfs.namenode.kerberos.internal.spnego.principal</name>

<value>HTTP/\_HOST@HADOOP.COM</value>

</property>

<!-- JournalNode Web服务的Kerberos主体 -->

<property>

<name>dfs.journalnode.kerberos.internal.spnego.principal</name>

<value>HTTP/\_HOST@HADOOP.COM</value>

</property>

<!-- Hadoop Web UI的Kerberos密钥文件路径 -->

<property>

<name>dfs.web.authentication.kerberos.keytab</name>

<value>/etc/security/keytab/spnego.service.keytab</value>

</property>

<!-- Secondary NameNode Web UI服务的Kerberos主体 -->

<property>

<name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>

<value>HTTP/\_HOST@HADOOP.COM</value>

</property>

<!-- DataNode服务的Kerberos主体 -->

<property>

<name>dfs.datanode.kerberos.principal</name>

<value>dn/\_HOST@HADOOP.COM</value>

</property>

<!-- DataNode服务的Kerberos密钥文件路径 -->

<property>

<name>dfs.datanode.keytab.file</name>

<value>/etc/security/keytab/dn.service.keytab</value>

</property>

<!-- WebHDFS REST服务的Kerberos主体 -->

<property>

<name>dfs.web.authentication.kerberos.principal</name>

<value>HTTP/\_HOST@HADOOP.COM</value>

</property>

<!-- 配置NameNode Web UI 使用HTTPS协议 -->

<property>

<name>dfs.http.policy</name>

<value>HTTPS\_ONLY</value>

</property>

<!-- 配置DataNode数据传输保护策略为仅认证模式 -->

<property>

<name>dfs.data.transfer.protection</name>

<value>authentication</value>

</property>

**3）修改yarn-site.xml**

[root@hadoop01 ~]# vim /opt/module/hadoop-3.1.3/etc/hadoop/yarn-site.xml

增加以下内容

<!-- Resource Manager 服务的Kerberos主体 -->

<property>

<name>yarn.resourcemanager.principal</name>

<value>rm/\_HOST@HADOOP.COM</value>

</property>

<!-- Resource Manager 服务的Kerberos密钥文件 -->

<property>

<name>yarn.resourcemanager.keytab</name>

<value>/etc/security/keytab/rm.service.keytab</value>

</property>

<!-- Node Manager 服务的Kerberos主体 -->

<property>

<name>yarn.nodemanager.principal</name>

<value>nm/\_HOST@HADOOP.COM</value>

</property>

<!-- Node Manager 服务的Kerberos密钥文件 -->

<property>

<name>yarn.nodemanager.keytab</name>

<value>/etc/security/keytab/nm.service.keytab</value>

</property>

**4）修改mapred-site.xml**

[root@hadoop01 ~]# vim /opt/module/hadoop-3.1.3/etc/hadoop/mapred-site.xml

增加以下内容

<!-- 历史服务器的Kerberos主体 -->

<property>

<name>mapreduce.jobhistory.keytab</name>

<value>/etc/security/keytab/jhs.service.keytab</value>

</property>

<!-- 历史服务器的Kerberos密钥文件 -->

<property>

<name>mapreduce.jobhistory.principal</name>

<value>jhs/\_HOST@HADOOP.COM</value>

</property>

**（5）分发以上修改的配置文件**

[root@hadoop01 ~]# xsync /opt/module/hadoop-3.1.3/etc/hadoop/core-site.xml

[root@hadoop01 ~]# xsync /opt/module/hadoop-3.1.3/etc/hadoop/hdfs-site.xml

[root@hadoop01 ~]# xsync /opt/module/hadoop-3.1.3/etc/hadoop/yarn-site.xml

[root@hadoop01 ~]# xsync /opt/module/hadoop-3.1.3/etc/hadoop/mapred-site.xml

**（6）修改linux环境变量**

[root@hadoop01 ~]# vim /etc/profile.d/my\_env.sh

#按红色部分修改

#HADOOP\_HOME

export HADOOP\_HOME=/opt/module2/hadoop-3.1.3

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

export PATH=$PATH:$HADOOP\_HOME/sbin

export HADOOP\_CONF\_DIR=${HADOOP\_HOME}/etc/hadoop

export HADOOP\_CLASSPATH=`hadoop classpath`

#hadoop

export HDFS\_NAMENODE\_USER=hdfs

export HDFS\_DATANODE\_USER=hdfs

export HDFS\_JOURNALNODE\_USER=hdfs

export HDFS\_SECONDARYNAMENODE\_USER=hdfs

export YARN\_RESOURCEMANAGER\_USER=yarn

export YARN\_NODEMANAGER\_USER=yarn

export HDFS\_ZKFC\_USER=hdfs

### 4 配置HDFS使用HTTPS安全传输协议

**1.生成密钥对**

Keytool是java数据证书的管理工具，使用户能够管理自己的公/私钥对及相关证书。

-keystore    指定密钥库的名称及位置(产生的各类信息将存在.keystore文件中)

-genkey(或者-genkeypair)      生成密钥对

-alias 为生成的密钥对指定别名，如果没有默认是mykey

-keyalg  指定密钥的算法 RSA/DSA 默认是DSA

**1）生成 keystore的密码及相应信息的密钥库**

**#例子如下**

[root@hadoop01 ~]# keytool -keystore /etc/security/keytab/keystore -alias jetty -genkey -keyalg RSA

输入密钥库口令: 123456

再次输入新口令: 123456

您的名字与姓氏是什么?

[Unknown]:

您的组织单位名称是什么?

[Unknown]:

您的组织名称是什么?

[Unknown]:

您所在的城市或区域名称是什么?

[Unknown]:

您所在的省/市/自治区名称是什么?

[Unknown]:

该单位的双字母国家/地区代码是什么?

[Unknown]:

CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown是否正确?

[否]: y

输入 <jetty> 的密钥口令

(如果和密钥库口令相同, 按回车):

再次输入新口令:

**#正式如下**

[root@hadoop01 ~]# name="CN=$HOSTNAME, OU=zhtx, O=zhtx, L=shijiazhuang, ST=hebei, C=CN"

[root@hadoop01 ~]# keytool -keystore /etc/security/keytab/newkeystore -alias hatest -genkey -keyalg RSA -dname "$name"

**#注：这里的$HOSTNAME 指hadoop01，指定域名是为了后面namenode HA用，同步nn时ssl会检查域名是否匹配**

（注意）查看密钥库keystore的命令

keytool -keystore keystore -list

**2）修改keystore文件的所有者和访问权限**

[root@hadoop01 ~]# chown -R root:hadoop /etc/security/keytab/newkeystore

[root@hadoop01 ~]# chmod 660 /etc/security/keytab/newkeystore

**注意：**

（1）密钥库的密码至少6个字符，可以是纯数字或者字母或者数字和字母的组合等等

（2）确保hdfs用户（HDFS的启动用户）具有对所生成keystore文件的读权限

**3）将该证书分发到集群中的每台节点的相同路径**

[root@hadoop01 ~]# xsync /etc/security/keytab/newkeystore

**4）修改hadoop配置文件ssl-server.xml.example，**

**该文件位于$HADOOP\_HOME/etc/hadoop目录**

**修改文件名为ssl-server.xml**

[root@hadoop01 ~]# mv $HADOOP\_HOME/etc/hadoop/ssl-server.xml.example $HADOOP\_HOME/etc/hadoop/ssl-server.xml

**修改以下内容**

[root@hadoop01 ~]# vim $HADOOP\_HOME/etc/hadoop/ssl-server.xml

修改以下参数

<!-- SSL可信任密钥库路径 -->

<property>

<name>ssl.server.truststore.location</name>

<value>/etc/security/keytab/newkeystore</value>

</property>

<!-- SSL可信任密钥库密码 -->

<property>

<name>ssl.server.truststore.password</name>

<value>123456</value>

</property>

<!-- SSL密钥库路径 -->

<property>

<name>ssl.server.keystore.location</name>

<value>/etc/security/keytab/newkeystore</value>

</property>

<!-- SSL密钥库密码 -->

<property>

<name>ssl.server.keystore.password</name>

<value>123456</value>

</property>

<!-- SSL密钥库中密钥的密码 -->

<property>

<name>ssl.server.keystore.keypassword</name>

<value>123456</value>

</property>

**5）分发ssl-server.xml文件**

[root@hadoop01 ~]# xsync $HADOOP\_HOME/etc/hadoop/ssl-server.xml

**6）将证书导入到jdk中**

先将证书导出为hatest.crt文件

[root@hadoop01 ~]# keytool -export -trustcacerts -alias hatest -file /opt/software/hatest.crt -keystore /etc/security/keytab/newkeystore

解释

-alias : 创建证书时起用的别名

-file : 导出保存时的文件名称

-keystore:证书库位置

然后将证书导入到jdk环境中

[root@hadoop01 ~]# keytool -import -trustcacerts -alias hatest -file /opt/software/hatest.crt -keystore /opt/module2/jdk1.8.0\_212/jre/lib/security/cacerts -storepass changeit

解释：

-alias : 创建证书时起用的别名

-file : 从哪里获取crt文件

-keystore: 本地存储jdk证书库的位置

-storepass: jdk的权限密码（默认为changeit）

查看证书是否被存储

[root@hadoop01 ~]# keytool -list -keystore /opt/module2/jdk1.8.0\_212/jre/lib/security/cacerts -storepass changeit |grep hatest



分发jdk的证书库

[root@hadoop01 ~]# cd /opt/module2/jdk1.8.0\_212/jre/lib/security

[root@hadoop01 security]# xsync cacerts

### 5 配置Yarn使用LinuxContainerExecutor

1）修改**所有节点**的container-executor所有者和权限，要求其所有者为root，所有组为hadoop（启动NodeManger的yarn用户的所属组），权限为6050。其默认路径为$HADOOP\_HOME/bin

[root@hadoop01 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/bin/container-executor

[root@hadoop01 ~]# chmod 6050 /opt/module/hadoop-3.1.3/bin/container-executor

[root@hadoop02 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/bin/container-executor

[root@hadoop02 ~]# chmod 6050 /opt/module/hadoop-3.1.3/bin/container-executor

[root@hadoop03 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/bin/container-executor

[root@hadoop03 ~]# chmod 6050 /opt/module/hadoop-3.1.3/bin/container-executor

2）修改**所有节点**的container-executor.cfg文件的所有者和权限，要求该文件及其所有的上级目录的所有者均为root，所有组为hadoop（启动NodeManger的yarn用户的所属组），权限为400。其默认路径为$HADOOP\_HOME/etc/hadoop

[root@hadoop01 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc/hadoop/container-executor.cfg

[root@hadoop01 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc/hadoop

[root@hadoop01 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc

[root@hadoop01 ~]# chown root:hadoop /opt/module/hadoop-3.1.3

[root@hadoop01 ~]# chown root:hadoop /opt/module

[root@hadoop01 ~]# chmod 400 /opt/module/hadoop-3.1.3/etc/hadoop/container-executor.cfg

[root@hadoop02 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc/hadoop/container-executor.cfg

[root@hadoop02 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc/hadoop

[root@hadoop02 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc

[root@hadoop02 ~]# chown root:hadoop /opt/module/hadoop-3.1.3

[root@hadoop02 ~]# chown root:hadoop /opt/module

[root@hadoop02 ~]# chmod 400 /opt/module/hadoop-3.1.3/etc/hadoop/container-executor.cfg

[root@hadoop03 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc/hadoop/container-executor.cfg

[root@hadoop03 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc/hadoop

[root@hadoop03 ~]# chown root:hadoop /opt/module/hadoop-3.1.3/etc

[root@hadoop03 ~]# chown root:hadoop /opt/module/hadoop-3.1.3

[root@hadoop03 ~]# chown root:hadoop /opt/module

[root@hadoop03 ~]# chmod 400 /opt/module/hadoop-3.1.3/etc/hadoop/container-executor.cfg

3）修改$HADOOP\_HOME/etc/hadoop/container-executor.cfg

[root@hadoop01 ~]# vim $HADOOP\_HOME/etc/hadoop/container-executor.cfg

内容如下

yarn.nodemanager.linux-container-executor.group=hadoop

#为了安全，将超级用户禁用提交任务权限

banned.users=hdfs,yarn,mapred

#允许提交任务的用户id，系统用户0~1000，自建用户>=1000

min.user.id=1000

#为了和min.user.id参数区分加，允许一些系统用户也可以提交

allowed.system.users=root

feature.tc.enabled=false

4）修改$HADOOP\_HOME/etc/hadoop/yarn-site.xml文件

[root@hadoop01 ~]# vim $HADOOP\_HOME/etc/hadoop/yarn-site.xml

增加以下内容

<!-- 配置Node Manager使用LinuxContainerExecutor管理Container -->

<property>

<name>yarn.nodemanager.container-executor.class</name>

<value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</value>

</property>

<!-- 配置Node Manager的启动用户的所属组 -->

<property>

<name>yarn.nodemanager.linux-container-executor.group</name>

<value>hadoop</value>

</property>

<!-- LinuxContainerExecutor脚本路径 -->

<property>

<name>yarn.nodemanager.linux-container-executor.path</name>

<value>/opt/module/hadoop-3.1.3/bin/container-executor</value>

</property>

5）分发container-executor.cfg和yarn-site.xml文件

[root@hadoop01 ~]# xsync $HADOOP\_HOME/etc/hadoop/container-executor.cfg

[root@hadoop01 ~]# xsync $HADOOP\_HOME/etc/hadoop/yarn-site.xml

### 6 安全模式下启动Hadoop集群

#### 6.1 修改特定本地路径权限

|  |  |  |  |
| --- | --- | --- | --- |
| **local** | $HADOOP\_LOG\_DIR | hdfs:hadoop | drwxrwxr-x |
| **local** | dfs.namenode.name.dir | hdfs:hadoop | drwx------ |
| **local** | dfs.datanode.data.dir | hdfs:hadoop | drwx------ |
| **local** | dfs.namenode.checkpoint.dir | hdfs:hadoop | drwx------ |
| **local** | yarn.nodemanager.local-dirs | yarn:hadoop | drwxrwxr-x |
| **local** | yarn.nodemanager.log-dirs | yarn:hadoop | drwxrwxr-x |

1）$HADOOP\_LOG\_DIR（所有节点）

该变量位于hadoop-env.sh文件，默认值为 ${HADOOP\_HOME}/logs

[root@hadoop01 ~]# chown hdfs:hadoop /opt/module/hadoop-3.1.3/logs/

[root@hadoop01 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/

[root@hadoop02 ~]# chown hdfs:hadoop /opt/module/hadoop-3.1.3/logs/

[root@hadoop02 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/

[root@hadoop03 ~]# chown hdfs:hadoop /opt/module/hadoop-3.1.3/logs/

[root@hadoop03 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/

2）dfs.namenode.name.dir（NameNode节点）

该参数位于hdfs-site.xml文件，默认值为file://${hadoop.tmp.dir}/dfs/name

[root@hadoop01 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/name/

[root@hadoop01 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/name/

NameNode高可用的其他节点（本架构只有hadoop02）

[root@hadoop02 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/name/

[root@hadoop02 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/name/

3）dfs.datanode.data.dir（DataNode节点）

该参数为于hdfs-site.xml文件，默认值为file://${hadoop.tmp.dir}/dfs/data

[root@hadoop01 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/data/

[root@hadoop01 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/data/

[root@hadoop02 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/data/

[root@hadoop02 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/data/

[root@hadoop03 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/data/

[root@hadoop03 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/data/

4）【可不配，因为Hadoop HA没开启2nn】dfs.namenode.checkpoint.dir（SecondaryNameNode节点）

该参数位于hdfs-site.xml文件，默认值为file://${hadoop.tmp.dir}/dfs/namesecondary

[root@hadoop03 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/dfs/namesecondary/

[root@hadoop03 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/dfs/namesecondary/

5）yarn.nodemanager.local-dirs（NodeManager节点）

该参数位于yarn-site.xml文件，默认值为file://${hadoop.tmp.dir}/nm-local-dir

[root@hadoop01 ~]# chown -R yarn:hadoop /opt/module/hadoop-3.1.3/data/nm-local-dir/

[root@hadoop01 ~]# chmod -R 775 /opt/module/hadoop-3.1.3/data/nm-local-dir/

[root@hadoop02 ~]# chown -R yarn:hadoop /opt/module/hadoop-3.1.3/data/nm-local-dir/

[root@hadoop02 ~]# chmod -R 775 /opt/module/hadoop-3.1.3/data/nm-local-dir/

[root@hadoop03 ~]# chown -R yarn:hadoop /opt/module/hadoop-3.1.3/data/nm-local-dir/

[root@hadoop03 ~]# chmod -R 775 /opt/module/hadoop-3.1.3/data/nm-local-dir/

6）yarn.nodemanager.log-dirs（NodeManager节点）

该参数位于yarn-site.xml文件，默认值为$HADOOP\_LOG\_DIR/userlogs

[root@hadoop01 ~]# chown yarn:hadoop /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop01 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop02 ~]# chown yarn:hadoop /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop02 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop03 ~]# chown yarn:hadoop /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop03 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/userlogs/

该参数位于yarn-site.xml文件，默认值为${hadoop.tmp.dir}/jn

[root@hadoop01 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/jn/

[root@hadoop01 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/jn/

[root@hadoop02 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/jn/

[root@hadoop02 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/jn/

[root@hadoop03 ~]# chown -R hdfs:hadoop /opt/module/hadoop-3.1.3/data/jn/

[root@hadoop03 ~]# chmod 700 /opt/module/hadoop-3.1.3/data/jn/

**修改权限，保证hadoop组的成员能够正常运行**

[root@hadoop01 ~]# chown yarn:hadoop /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop01 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop02 ~]# chown yarn:hadoop /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop02 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop03 ~]# chown yarn:hadoop /opt/module/hadoop-3.1.3/logs/userlogs/

[root@hadoop03 ~]# chmod 775 /opt/module/hadoop-3.1.3/logs/userlogs/

#### 6.2 启动HDFS

需要注意的是，启动不同服务时需要使用对应的用户

1. **修改/etc/profile.d/my\_env.sh配置文件，然后分发配置配置文件**

[root@hadoop01 ~]# vim /etc/profile.d/my\_env.sh

#按如下内容修改

export HDFS\_NAMENODE\_USER=hdfs

export HDFS\_DATANODE\_USER=hdfs

export HDFS\_SECONDARYNAMENODE\_USER=hdfs

export YARN\_RESOURCEMANAGER\_USER=yarn

export YARN\_NODEMANAGER\_USER=yarn

[root@hadoop01 ~]# xrysrc /etc/profile.d/my\_env.sh

**1.单点启动**

（1）启动NameNode

[root@hadoop01 ~]# sudo -i -u hdfs hdfs --daemon start namenode

（2）启动DataNode

[root@hadoop01 ~]# sudo -i -u hdfs hdfs --daemon start datanode

[root@hadoop02 ~]# sudo -i -u hdfs hdfs --daemon start datanode

[root@hadoop03 ~]# sudo -i -u hdfs hdfs --daemon start datanode

（3）启动SecondaryNameNode

[root@hadoop03 ~]# sudo -i -u hdfs hdfs --daemon start secondarynamenode

**说明：**

* -i：重新加载环境变量
* -u：以特定用户的身份执行后续命令

**2.群起**

1）在主节点（hadoop01）配置hdfs用户到所有节点的免密登录

切换用户

[root@hadoop01 ~]# su hdfs

生成公私密钥

[hdfs@hadoop01 root]$ ssh-keygen

#(回车三次)

分发密钥**(密码hdfs)**

[hdfs@hadoop01 root]$ ssh-copy-id hadoop01

[hdfs@hadoop01 root]$ ssh-copy-id hadoop02

[hdfs@hadoop01 root]$ ssh-copy-id hadoop03

退出

[hdfs@hadoop01 root]$ exit

exit

2）修改主节点（hadoop01）节点的$HADOOP\_HOME/sbin/start-dfs.sh脚本，在顶部增加以下环境变量。

[root@hadoop01 ~]# vim $HADOOP\_HOME/sbin/start-dfs.sh

在顶部增加如下内容

#kerberos认证后群起需要加的参数

HDFS\_DATANODE\_USER=hdfs

HDFS\_NAMENODE\_USER=hdfs

HDFS\_JOURNALNODE\_USER=hdfs

HDFS\_ZKFC\_USER=hdfs

**注：$HADOOP\_HOME/sbin/stop-dfs.sh也需在顶部增加上述环境变量才可使用。**

3）以root用户执行群起脚本，即可启动HDFS集群。

[root@hadoop01 ~]# start-dfs.sh

**3.查看HFDS web页面**

访问地址为[https://hadoop01:9871](https://hadoop102:9871)

#### 6.3 修改HDFS特定路径访问权限

|  |  |  |  |
| --- | --- | --- | --- |
| **hdfs** | / | hdfs:hadoop | drwxr-xr-x |
| **hdfs** | /tmp | hdfs:hadoop | drwxrwxrwxt |
| **hdfs** | /user | hdfs:hadoop | drwxrwxr-x |
| **hdfs** | yarn.nodemanager.remote-app-log-dir | yarn:hadoop | drwxrwxrwxt |
| **hdfs** | mapreduce.jobhistory.intermediate-done-dir | mapred:hadoop | drwxrwxrwxt |
| **hdfs** | mapreduce.jobhistory.done-dir | mapred:hadoop | drwxrwx--- |

说明：

若上述路径不存在，需手动创建

1）创建hdfs/hadoop主体，执行以下命令并按照提示输入密码**（123456）**

[root@**hadoop02** ~]# kadmin.local -q "addprinc hdfs/hadoop"

2）认证hdfs/hadoop主体，执行以下命令并按照提示输入密码

[root@hadoop01 ~]# kinit hdfs/hadoop

**注：只有hdfs/hadoop账号能够修改hdfs目录的权限**

3）按照上述要求修改指定路径的所有者和权限

（1）修改/、/tmp、/user路径

[root@hadoop01 ~]# hadoop fs -chown hdfs:hadoop / /tmp /user

[root@hadoop01 ~]# hadoop fs -chmod 755 /

[root@hadoop01 ~]# hadoop fs -chmod 1777 /tmp

[root@hadoop01 ~]# hadoop fs -chmod 775 /user

（2）参数yarn.nodemanager.remote-app-log-dir位于yarn-site.xml文件，默认值/tmp/logs

[root@hadoop01 ~]# hadoop fs -chown yarn:hadoop /tmp/logs

[root@hadoop01 ~]# hadoop fs -chmod 1777 /tmp/logs

（3）参数mapreduce.jobhistory.intermediate-done-dir位于mapred-site.xml文件，默认值为/tmp/hadoop-yarn/staging/history/done\_intermediate，需保证该路径的所有上级目录（除/tmp）的所有者均为mapred，所属组为hadoop，权限为770

[root@hadoop01 ~]# hadoop fs -chown -R mapred:hadoop /tmp/hadoop-yarn/staging/history/done\_intermediate

[root@hadoop01 ~]# hadoop fs -chmod -R 1777 /tmp/hadoop-yarn/staging/history/done\_intermediate

[root@hadoop01 ~]# hadoop fs -chown mapred:hadoop /tmp/hadoop-yarn/staging/history/

[root@hadoop01 ~]# hadoop fs -chown mapred:hadoop /tmp/hadoop-yarn/staging/

[root@hadoop01 ~]# hadoop fs -chown mapred:hadoop /tmp/hadoop-yarn/

[root@hadoop01 ~]# hadoop fs -chmod 770 /tmp/hadoop-yarn/staging/history/

[root@hadoop01 ~]# hadoop fs -chmod 770 /tmp/hadoop-yarn/staging/

[root@hadoop01 ~]# hadoop fs -chmod 770 /tmp/hadoop-yarn/

（4）参数mapreduce.jobhistory.done-dir位于mapred-site.xml文件，默认值为/tmp/hadoop-yarn/staging/history/done，需保证该路径的所有上级目录（除/tmp）的所有者均为mapred，所属组为hadoop，权限为770

[root@hadoop01 ~]# hadoop fs -chown -R mapred:hadoop /tmp/hadoop-yarn/staging/history/done

[root@hadoop01 ~]# hadoop fs -chmod -R 750 /tmp/hadoop-yarn/staging/history/done

[root@hadoop01 ~]# hadoop fs -chown mapred:hadoop /tmp/hadoop-yarn/staging/history/

[root@hadoop01 ~]# hadoop fs -chown mapred:hadoop /tmp/hadoop-yarn/staging/

[root@hadoop01 ~]# hadoop fs -chown mapred:hadoop /tmp/hadoop-yarn/

[root@hadoop01 ~]# hadoop fs -chmod 770 /tmp/hadoop-yarn/staging/history/

[root@hadoop01 ~]# hadoop fs -chmod 770 /tmp/hadoop-yarn/staging/

[root@hadoop01 ~]# hadoop fs -chmod 770 /tmp/hadoop-yarn/

#### 6.4 启动Yarn

**1.单点启动**

**启动ResourceManager**

[root@hadoop02 ~]# sudo -i -u yarn yarn --daemon start resourcemanager

**启动NodeManager**

[root@hadoop01 ~]# sudo -i -u yarn yarn --daemon start nodemanager

[root@hadoop02 ~]# sudo -i -u yarn yarn --daemon start nodemanager

[root@hadoop03 ~]# sudo -i -u yarn yarn --daemon start nodemanager

**2.群起**

1）在Yarn主节点（hadoop02）配置**yarn**用户到所有节点的免密登录

切换用户

[root@hadoop02 ~]# su yarn

生成公私密钥

[yarn@hadoop01 root]$ ssh-keygen

#(回车三次)

分发密钥**(密码yarn)**

[yarn@hadoop02 root]$ ssh-copy-id hadoop01

[yarn@hadoop02 root]$ ssh-copy-id hadoop02

[yarn@hadoop02 root]$ ssh-copy-id hadoop03

退出

[yarn@hadoop02 root]$ exit

exit

2）修改主节点（hadoop02）的$HADOOP\_HOME/sbin/start-yarn.sh，在顶部增加以下环境变量。

[root@hadoop02 ~]# vim $HADOOP\_HOME/sbin/start-yarn.sh

在顶部增加如下内容

#kerberos认证后群起需要加的参数

YARN\_RESOURCEMANAGER\_USER=yarn

YARN\_NODEMANAGER\_USER=yarn

**注：stop-yarn.sh也需在顶部增加上述环境变量才可使用。**

3）以root用户执行$HADOOP\_HOME/sbin/start-yarn.sh脚本即可启动yarn集群。

[root@hadoop02 ~]# start-yarn.sh

**3.访问Yarn web页面**

访问地址为[http://hadoop02:8088](http://hadoop103:8088)

#### 6.5 启动HistoryServer

**1.启动历史服务器**

[root@hadoop01 ~]# sudo -i -u mapred mapred --daemon start historyserver

**2.查看历史服务器web页面**

访问地址为[http://hadoop01:19888](http://hadoop102:19888)

### 7 Kerberos安全集群使用说明

#### 7.1 用户要求

**1.具体要求**

以下使用说明均基于普通用户，安全集群对用户有以下要求：

1）集群中的每个节点都需要创建该用户

2）该用户需要属于hadoop用户组

3）需要创建该用户对应的Kerberos主体

**2.实操**

此处以zhtx用户为例，具体操作如下

1）创建用户（存在可跳过），须在所有节点执行

[root@hadoop01 ~]# useradd zhtx

[root@hadoop01 ~]# echo zhtx | passwd --stdin zhtx

[root@hadoop02 ~]# useradd zhtx

[root@hadoop02 ~]# echo zhtx| passwd --stdin zhtx

[root@hadoop03 ~]# useradd zhtx

[root@hadoop03 ~]# echo zhtx| passwd --stdin zhtx

2）加入hadoop组，须在所有节点执行

[root@hadoop01 ~]# usermod -a -G hadoop zhtx

[root@hadoop02 ~]# usermod -a -G hadoop zhtx

[root@hadoop03 ~]# usermod -a -G hadoop zhtx

3）创建主体

[root@hadoop02 ~]# kadmin -p admin/admin -w123456 -q"addprinc -pw 123456 zhtx"

#### 7.3 提交MapReduce任务

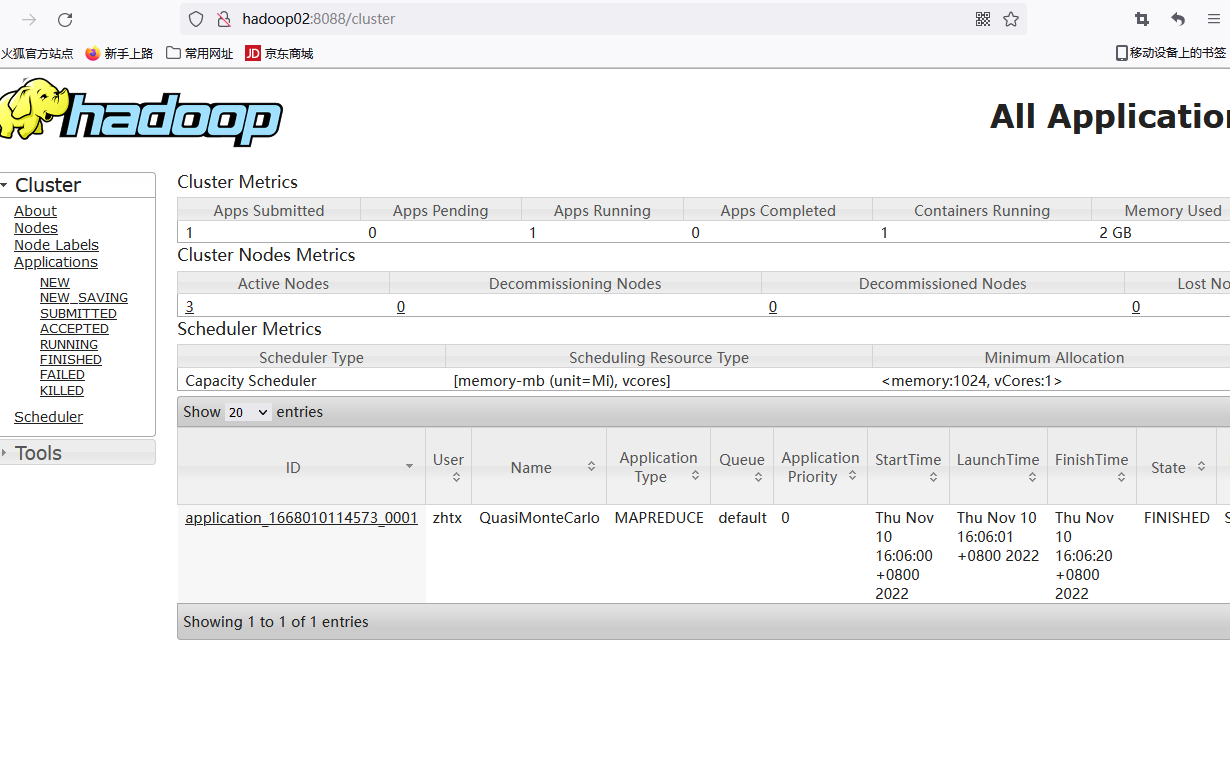
1.认证

[zhtx@hadoop01 ~]$ kinit zhtx

2.提交任务

[zhtx@hadoop01 ~]$ hadoop jar /opt/module/hadoop-3.1.3/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar pi 1 1

运行成功



#### 7.2 访问HDFS集群文件

**7.2.1 web页面访问**

**1.安装Kerberos客户端**

下载地址：<http://web.mit.edu/kerberos/dist/kfw/4.1/kfw-4.1-amd64.msi>

1）下载之后按照提示安装

2）编辑C:\ProgramData\MIT\Kerberos5\krb5.ini文件，添加内容如下

[libdefaults]

dns\_lookup\_realm = false

ticket\_lifetime = 24h

forwardable = true

rdns = false

default\_realm = HADOOP.COM

[realms]

HADOOP.COM = {

kdc = hadoop02

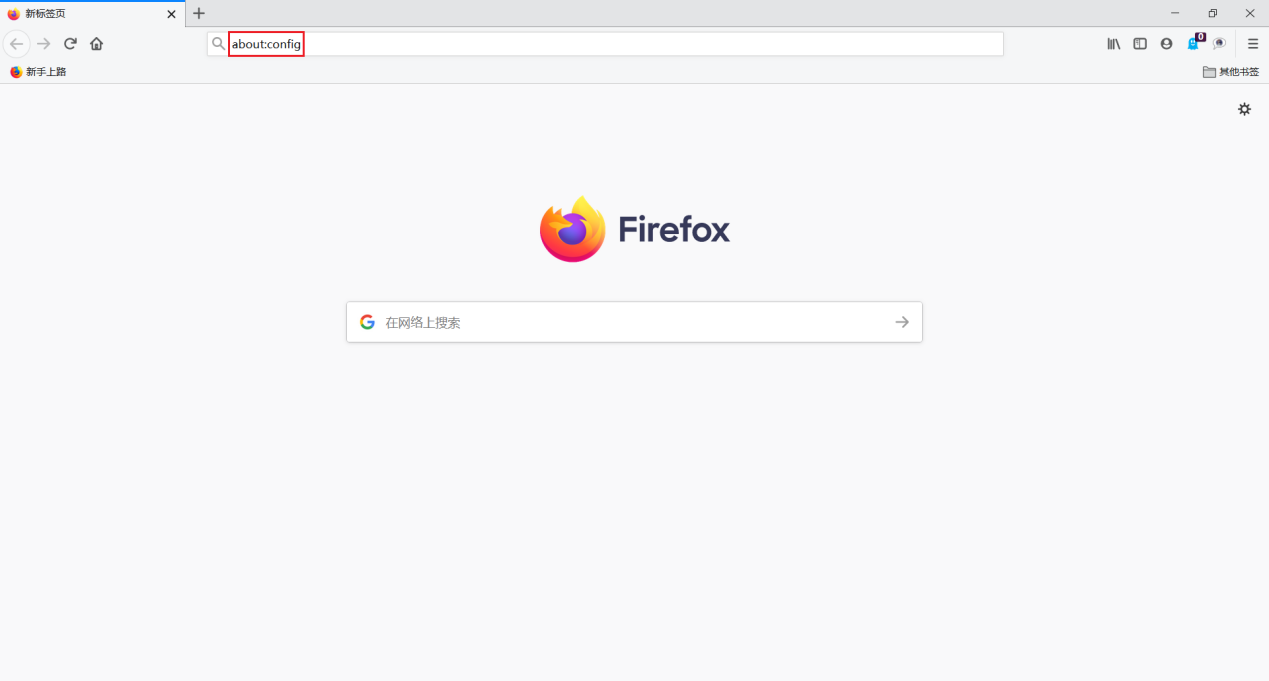
admin\_server = hadoop02

}

[domain\_realm]

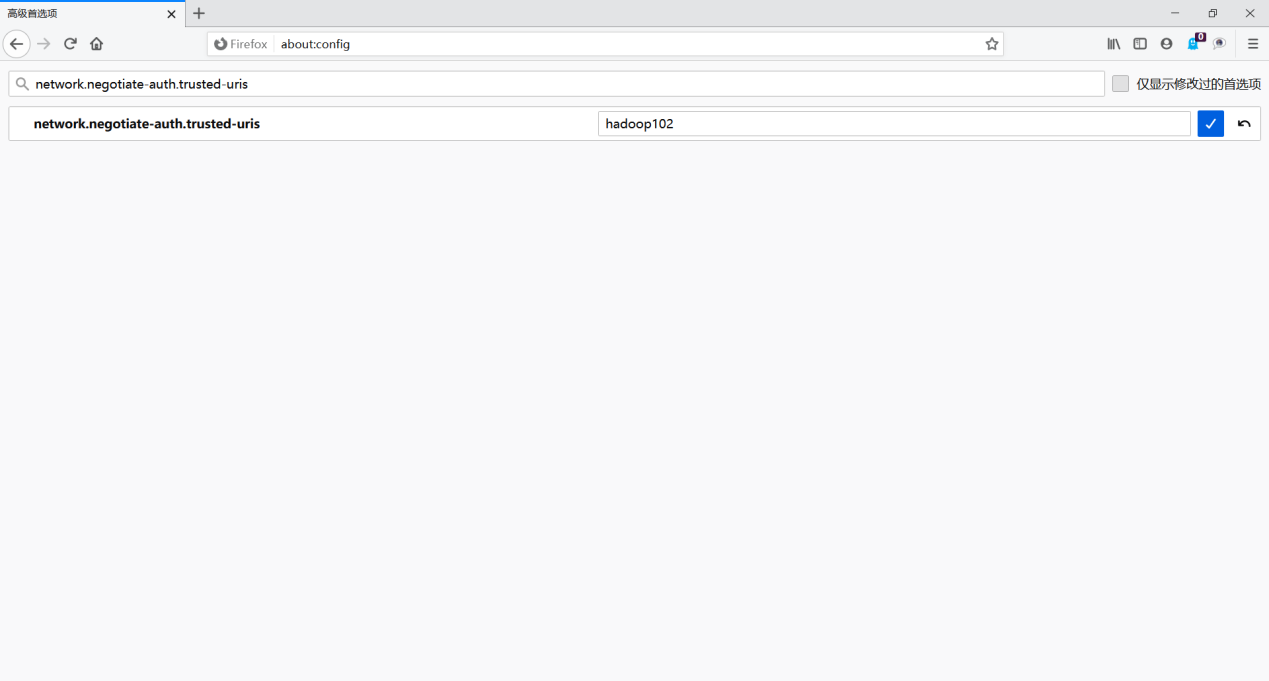
**2.配置火狐浏览器**

1）打开浏览器，在地址栏输入“about:config”，点击回车

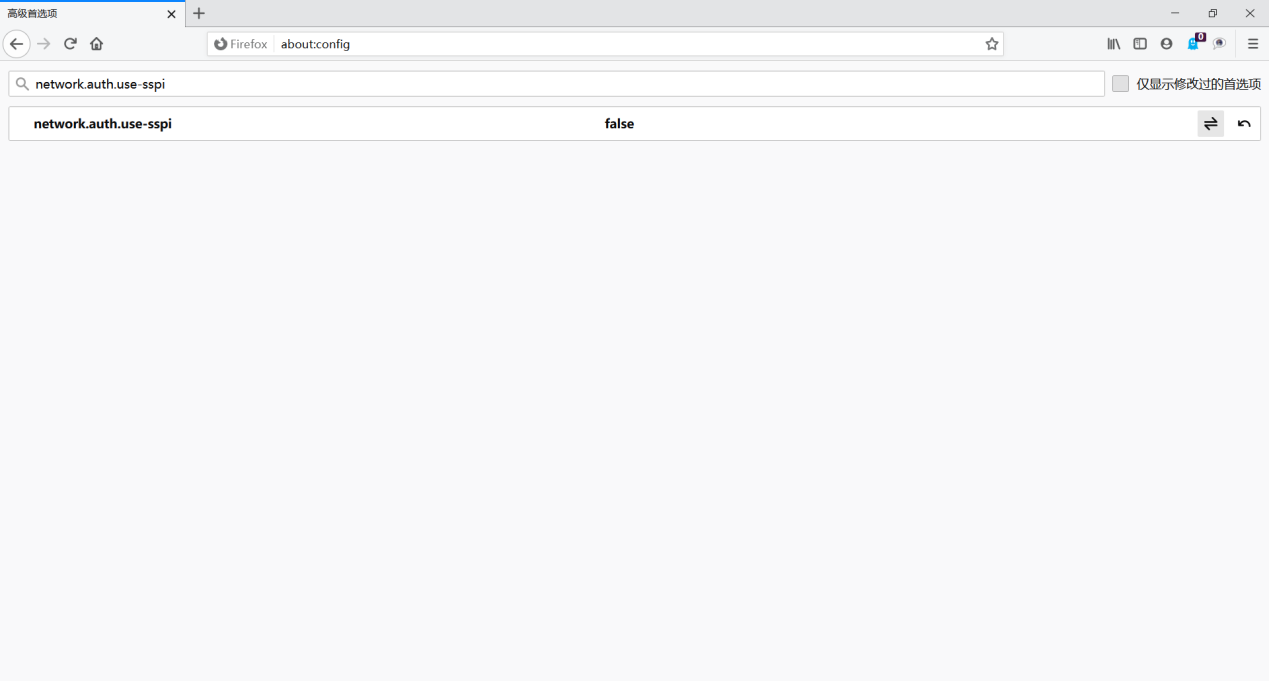




２）搜索“network.negotiate-auth.trusted-uris”，修改值为要访问的主机名（hadoop01）

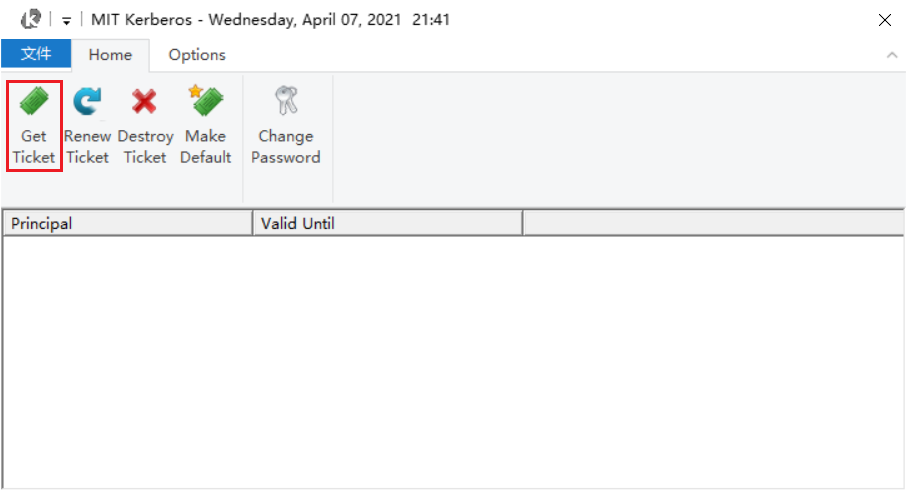


3）搜索“network.auth.use-sspi”，双击将值变为false

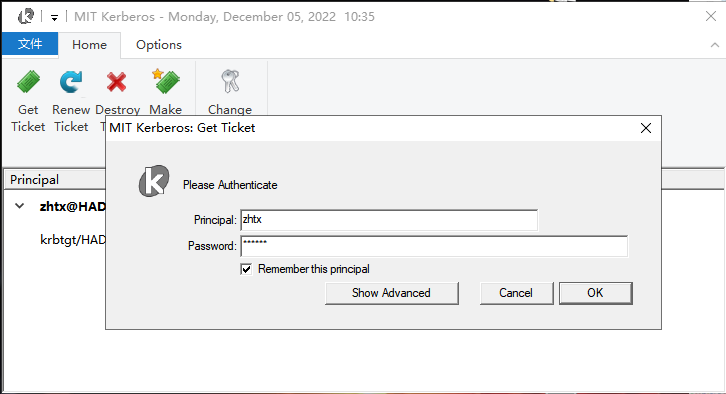


**3.认证**

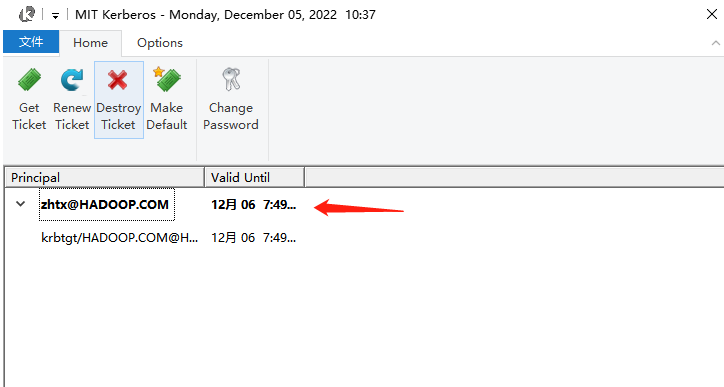
**１）启动Kerberos客户端，点击Get Ticket**



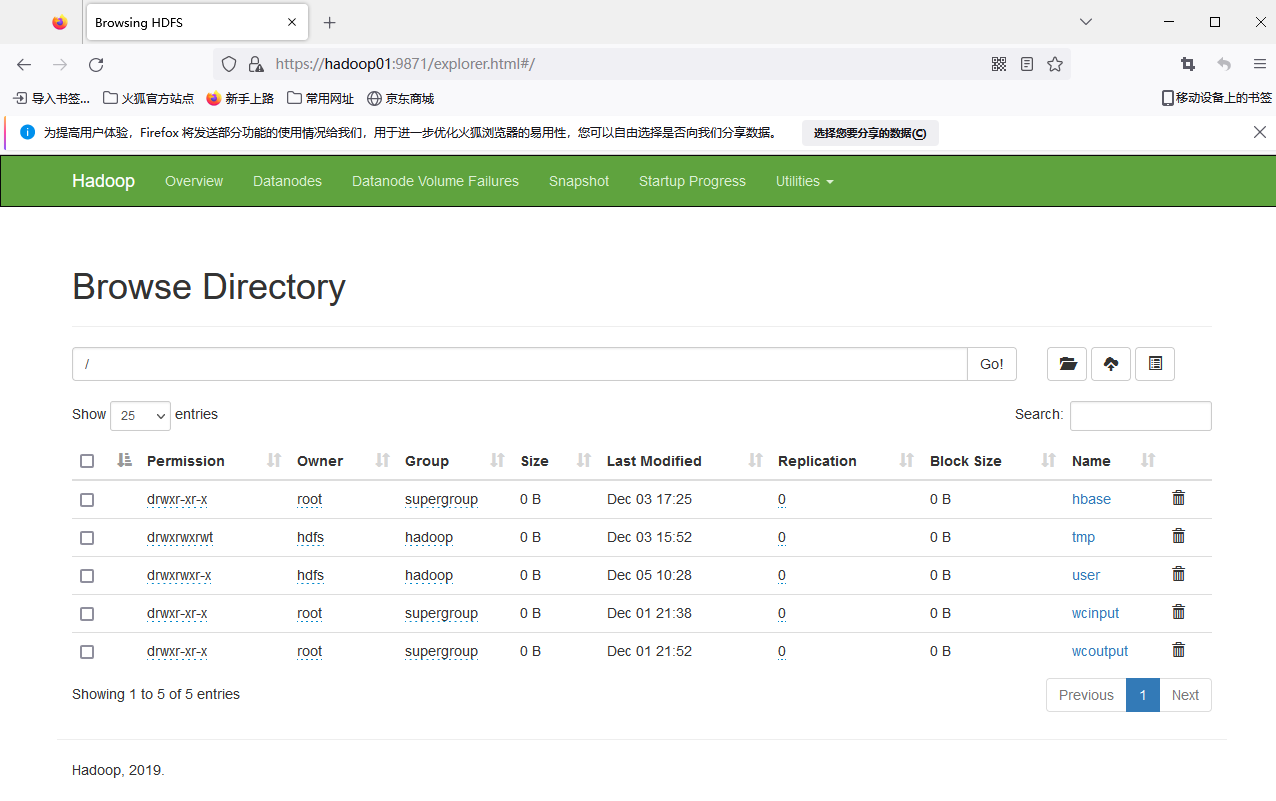
**２）输入主体名和密码，点击OK**



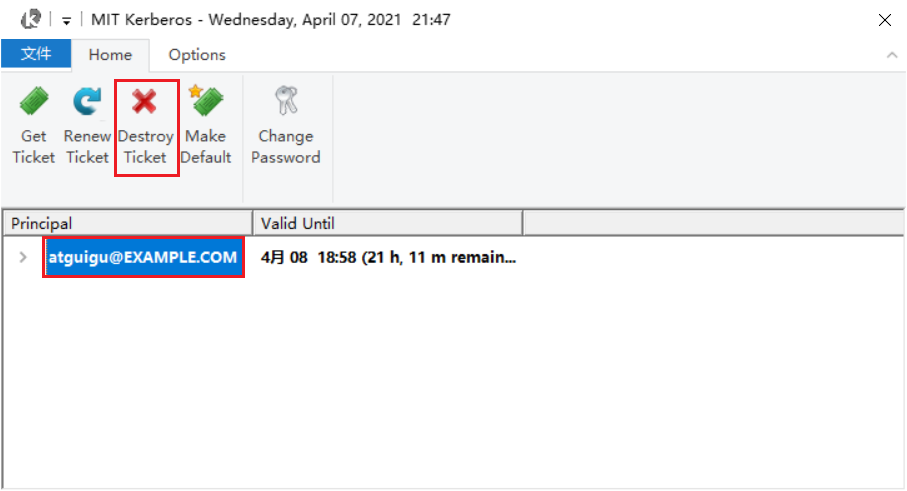
**３）认证成功**



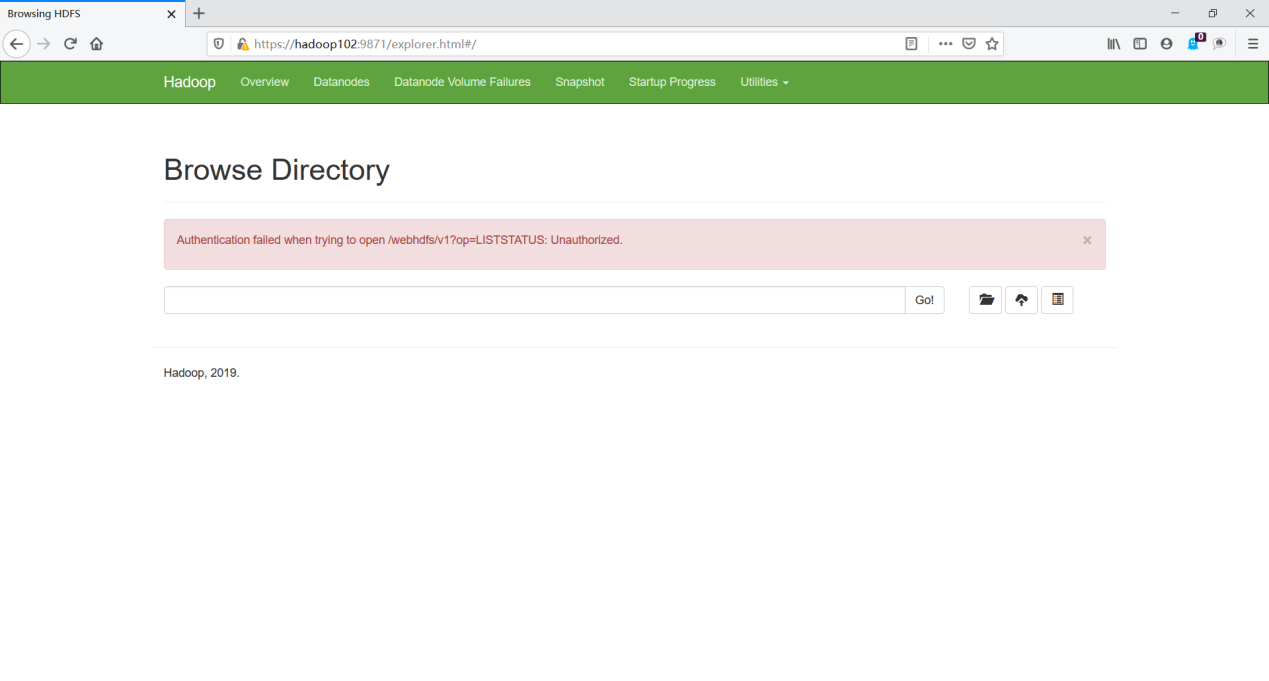
**4.访问HDFS**



**5.注销认证**



**6.重启浏览器，再次访问HDFS**



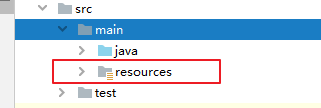
## HDFS HA Kerberos示例代码

### 1 导入pom.xml依赖

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 </dependency>  
 <!--打印日志-->  
 <dependency>  
 <groupId>org.apache.logging.log4j</groupId>  
 <artifactId>log4j-slf4j-impl</artifactId>  
 <version>2.12.0</version>  
 </dependency>  
  
<!-- hadoop连接 -->  
 <dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-client</artifactId>  
 <version>3.1.3</version>  
 </dependency>

### 2 将配置文件放到项目的resources目录下

用到的配置文件有krb5.conf core-site.xml hdfs-site.xml zhtx.keytab



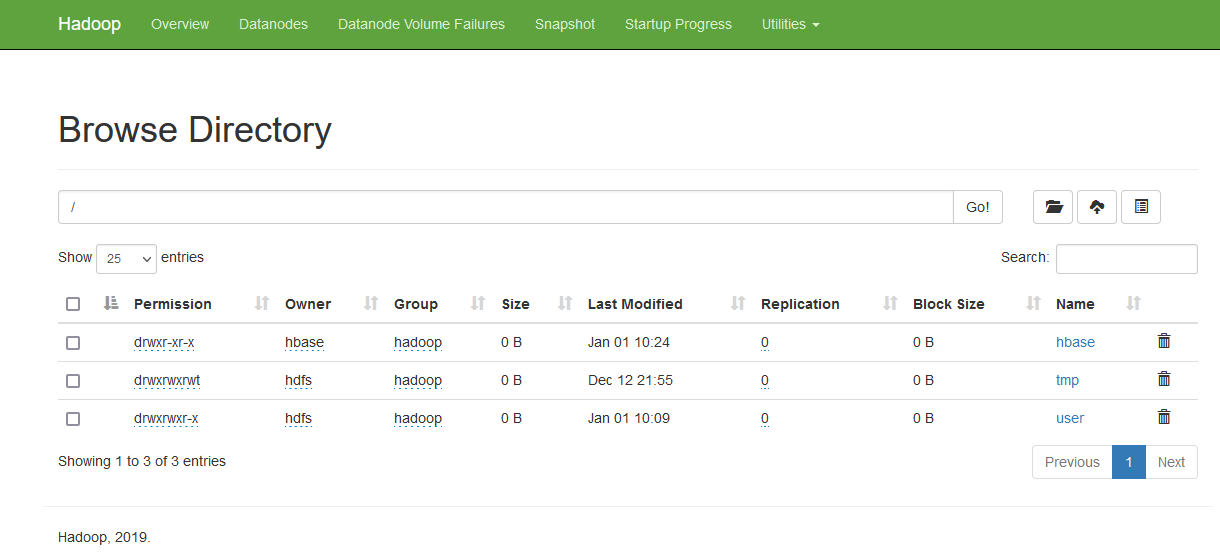
### 3 测试代码如下

判断hdfs根目录下的所有文件属性（是目录，还是文件）

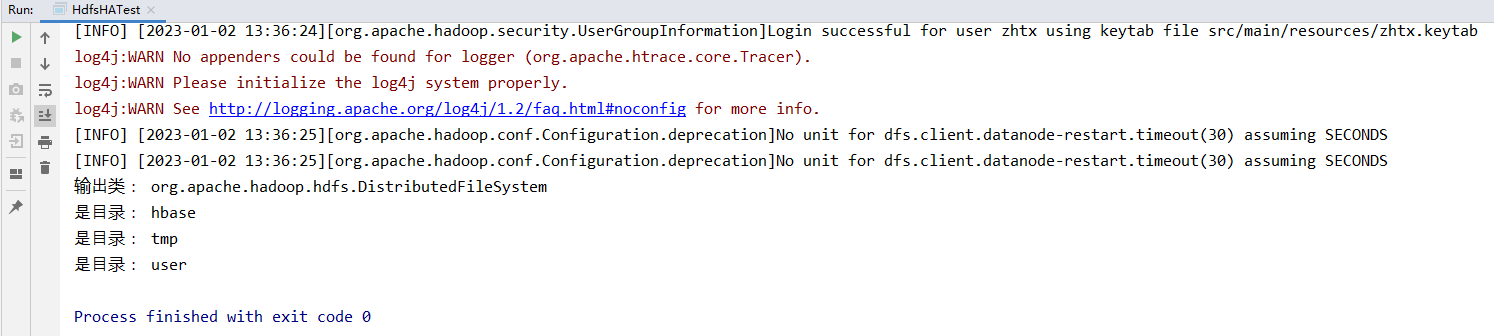
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.FileStatus;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.security.UserGroupInformation;  
  
import java.io.IOException;  
import java.net.URI;  
import java.net.URISyntaxException;  
  
public class HdfsHATest {  
 public static void main(String[] args) throws IOException, InterruptedException, URISyntaxException {  
 //如果不是在hadoop集群内部需要加如下参数  
 System.setProperty("java.security.krb5.conf", "src/main/resources/krb5.conf");  
  
 //声明配置  
 Configuration conf = new Configuration();  
  
 //添加hadoop配置  
 conf.addResource(new Path("src/main/resources/core-site.xml"));  
 conf.addResource(new Path("src/main/resources/hdfs-site.xml"));  
  
 //kerberos认证  
 UserGroupInformation.setConfiguration(conf);  
 UserGroupInformation.loginUserFromKeytab("zhtx","src/main/resources/zhtx.keytab");  
  
 //获取hadoop namenode地址  
 URI uri = new URI("hdfs://mycluster");  
 //获取hdfs  
 FileSystem fs = FileSystem.get(uri,conf);  
 System.out.println("输出类： "+fs.getClass().getName());  
  
 //判断根目录下的所有目录或文件的性质  
 FileStatus[] listStatus = fs.listStatus(new Path("/"));  
 for (FileStatus fileStatus : listStatus) {  
  
 if(fileStatus.isFile()){  
 System.out.println("是文件： "+fileStatus.getPath().getName());  
 }else {  
 System.out.println("是目录： "+fileStatus.getPath().getName());  
 }  
  
 }  
  
 fs.close();  
  
 }  
}

### 4 测试结果

hdfs根目录



代码测试结果



## Hive开启kerberos认证

### 1 创建Hive系统用户和Kerberos主体

1.创建系统用户

[root@hadoop01 ~]# useradd hive -g hadoop

[root@hadoop01 ~]# echo hive | passwd --stdin hive

[root@hadoop02 ~]# useradd hive -g hadoop

[root@hadoop02 ~]# echo hive | passwd --stdin hive

[root@hadoop03 ~]# useradd hive -g hadoop

[root@hadoop03 ~]# echo hive | passwd --stdin hive

2.创建Kerberos主体并生成keytab文件

创建hive用户的Kerberos主体

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey hive/hadoop01"

在Hive所部署的节点生成keytab文件

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/hive.service.keytab hive/hadoop01"

3.修改keytab文件所有者和访问权限

[root@hadoop01 ~]# chown -R root:hadoop /etc/security/keytab/hive.service.keytab

[root@hadoop01 ~]# chmod 660 /etc/security/keytab/hive.service.keytab

### 2 配置认证

#### 1)修改$HIVE\_HOME/conf/hive-site.xml文件，增加如下属性

[root@hadoop01 ~]# vim $HIVE\_HOME/conf/hive-site.xml

#增加如下内容

<!-- HiveServer2启用Kerberos认证 -->

<property>

<name>hive.server2.authentication</name>

<value>kerberos</value>

</property>

<!-- HiveServer2服务的Kerberos主体 -->

<property>

<name>hive.server2.authentication.kerberos.principal</name>

<value>hive/hadoop01@HADOOP.COM</value>

</property>

<!-- HiveServer2服务的Kerberos密钥文件 -->

<property>

<name>hive.server2.authentication.kerberos.keytab</name>

<value>/etc/security/keytab/hive.service.keytab</value>

</property>

<!-- Metastore 启动认证 -->

<property>

<name>hive.metastore.sasl.enabled</name>

<value>true</value>

</property>

<!-- Metastore服务的Kerberos主体 -->

<property>

<name>hive.metastore.kerberos.principal</name>

<value>hive/hadoop01@HADOOP.COM</value>

</property>

<!-- Metastore的Kerberos密钥文件 -->

<property>

<name>hive.metastore.kerberos.keytab.file</name>

<value>/etc/security/keytab/hive.service.keytab</value>

</property>

#### 2)修改$HADOOP\_HOME/etc/hadoop/core-site.xml文件，具体修改如下

[root@hadoop01 ~]# vim $HADOOP\_HOME/etc/hadoop/core-site.xml

**删除**以下参数

<property>

<name>hadoop.http.staticuser.user</name>

<value>root</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.root.users</name>

<value>\*</value>

</property>

**增加**以下参数

<property>

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

<value>\*</value>

</property>

<property>

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

<value>\*</value>

</property>

<property>

<name>hadoop.proxyuser.hive.users</name>

<value>\*</value>

</property>

#### 3)分发配置core-site.xml文件

[root@hadoop01 ~]# xsync $HADOOP\_HOME/etc/hadoop/core-site.xml

#### 4)重启Hadoop集群

[root@hadoop01 ~]# stop-dfs.sh

[root@hadoop02 ~]# stop-yarn.sh

[root@hadoop01 ~]# start-dfs.sh

[root@hadoop02 ~]# start-yarn.sh

#### 5)修改权限

[root@hadoop02 ~]# kinit hdfs/hadoop

[root@hadoop01 ~]# hadoop fs -chown -R hdfs:hadoop /user/hive/warehouse

[root@hadoop01 ~]# hadoop fs -chmod -R 775 /user/hive/warehouse

[root@hadoop02 ~]# chown -R hive:hadoop /opt/module2/hive-3.1.2

[root@hadoop01 ~]# chmod -R 775 /opt/module2/hive-3.1.2

**注：只有hdfs/hadoop账号能够修改hdfs目录的权限**

### 3 启动hive metastore和hiveserver2

[root@hadoop01 ~]# cd /opt/module/hive-3.1.2/bin

[root@hadoop01 bin]# my\_hiveservices.sh start

### 4 hiveserver2的jdbc连接方式

#第一种 未开启hiveServer2负载均衡

[root@hadoop01 ~]# beeline -u "jdbc:hive2://hadoop01:10000/;principal=hive/hadoop01@HADOOP.COM" -n root

#第二种 开启hiveServer2负载均衡

[root@hadoop01 bin]# beeline -u "jdbc:hive2://hadoop01:2181,hadoop02:2181,hadoop03:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2\_zk;principal=hive/hadoop01@HADOOP.COM" -n root

## Hive Kerberos认证使用说明

以下说明均基于普通用户

### 1 beeline客户端

1.认证，执行以下命令，并按照提示输入密码

[root@hadoop01 bin]# su zhtx

[zhtx@hadoop01 ~]$ kinit zhtx

**注：无论hive --service metastore 还是 hive --service hiveserver2 服务，想要提交mr任务都必须通过 kinit zhtx【用户所属组为hadoop】 认证**

2.使用beeline客户端连接hiveserver2

[zhtx@hadoop01 ~]$ beeline

使用如下url进行连接

> !connect **jdbc:hive2://hadoop01:10000/;principal=hive/hadoop01@HADOOP.COM**

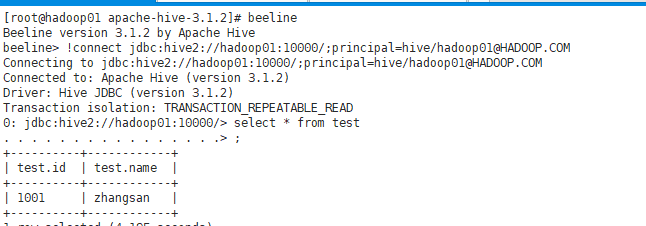
或者使用如下连接方式

[root@hadoop01 ~]# beeline -u "jdbc:hive2://hadoop01:10000/;principal=hive/hadoop01@HADOOP.COM"

或者

[root@hadoop01 ~]# beeline -u "jdbc:hive2://hadoop01:10000/;principal=hive/hadoop01@HADOOP.COM" -n root

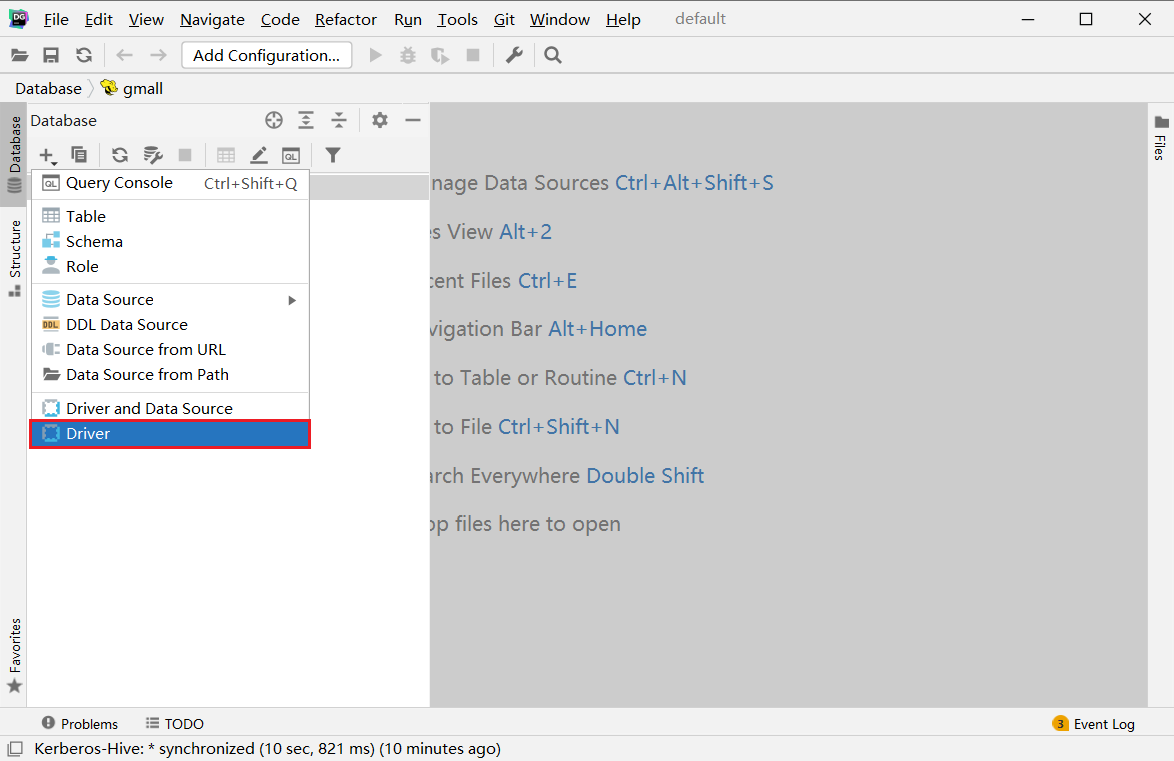
3.测试查询



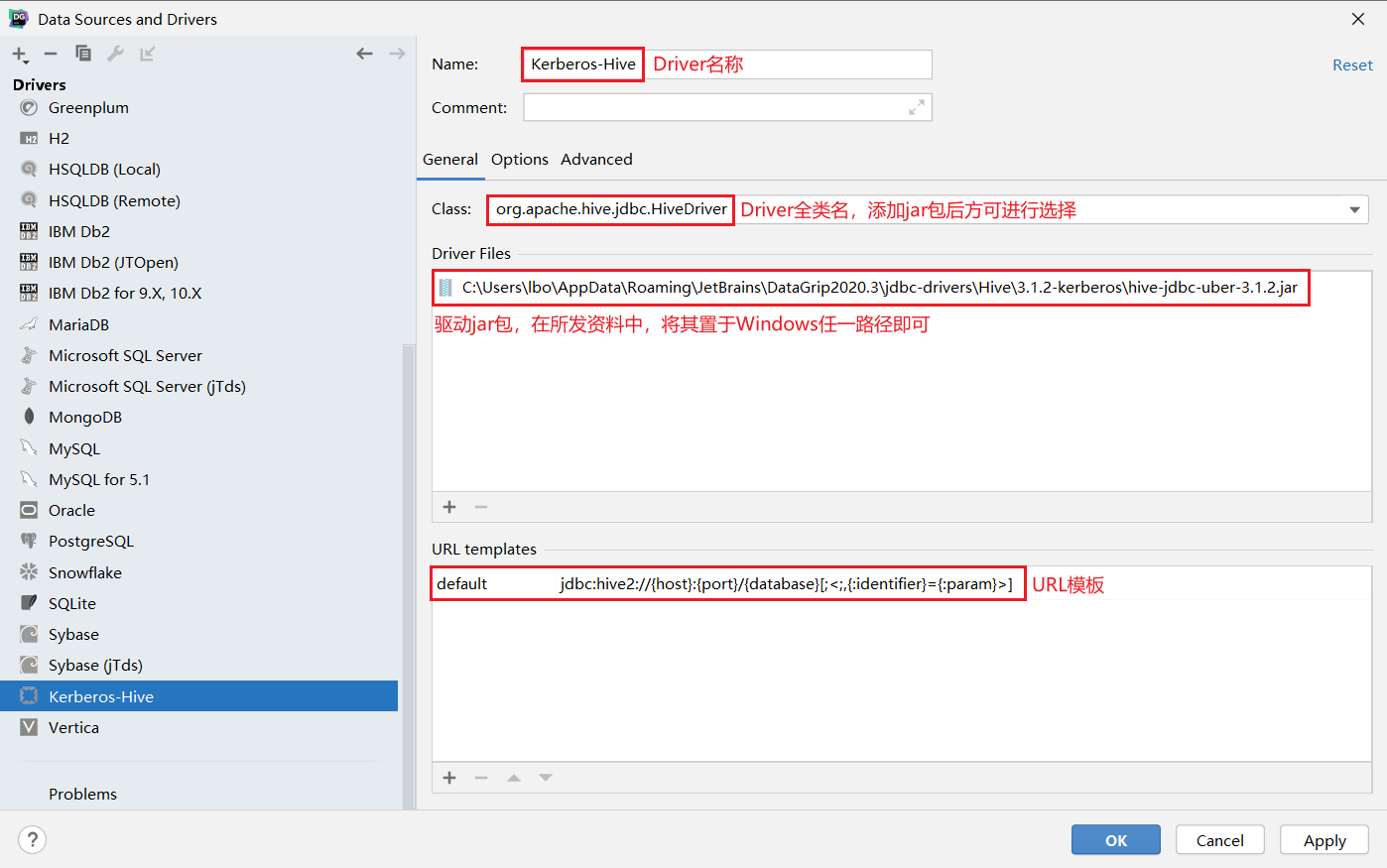
### 2 DataGrip客户端

#### 2.1 新建Driver

1.创建Driver



2.配置Driver

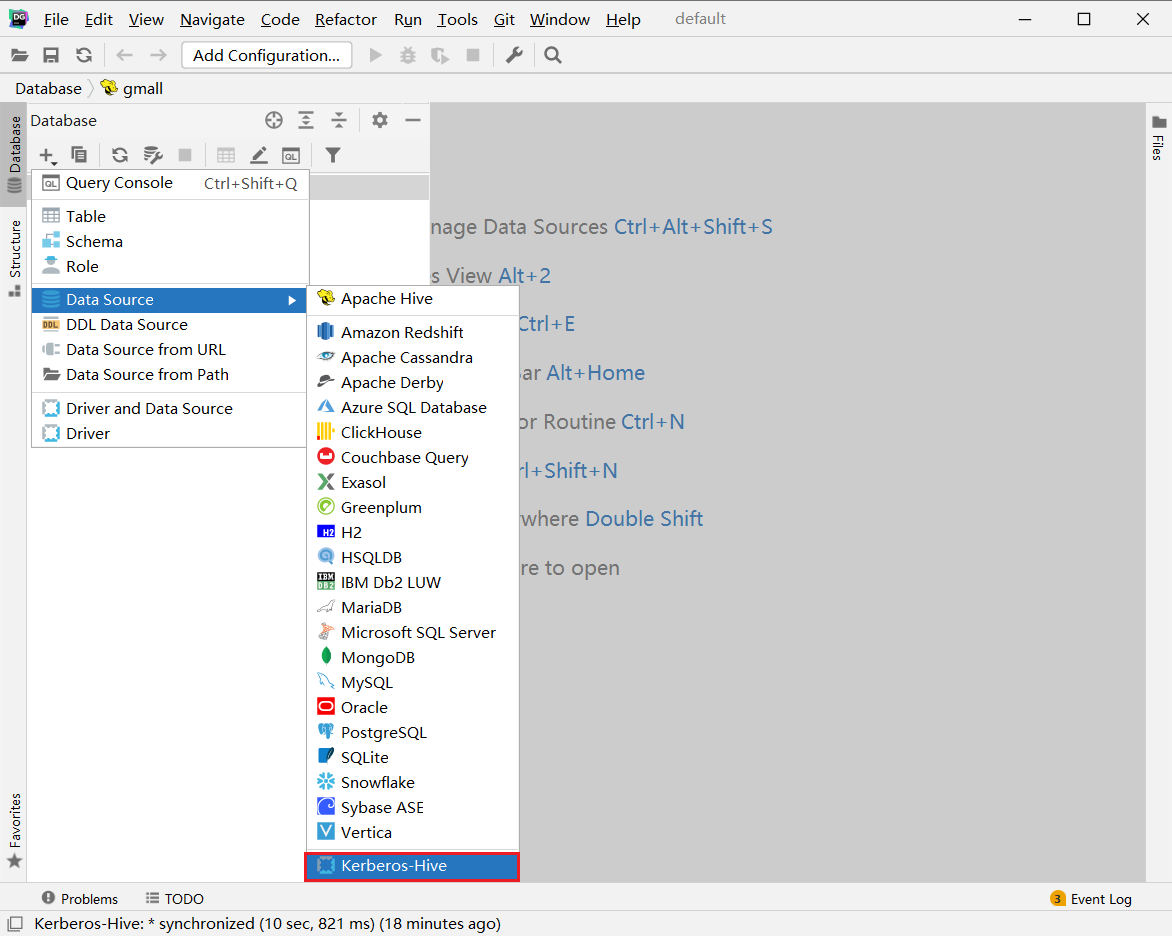


**注：**

url模板：jdbc:hive2://{host}:{port}/{database}[;<;,{:identifier}={:param}>]

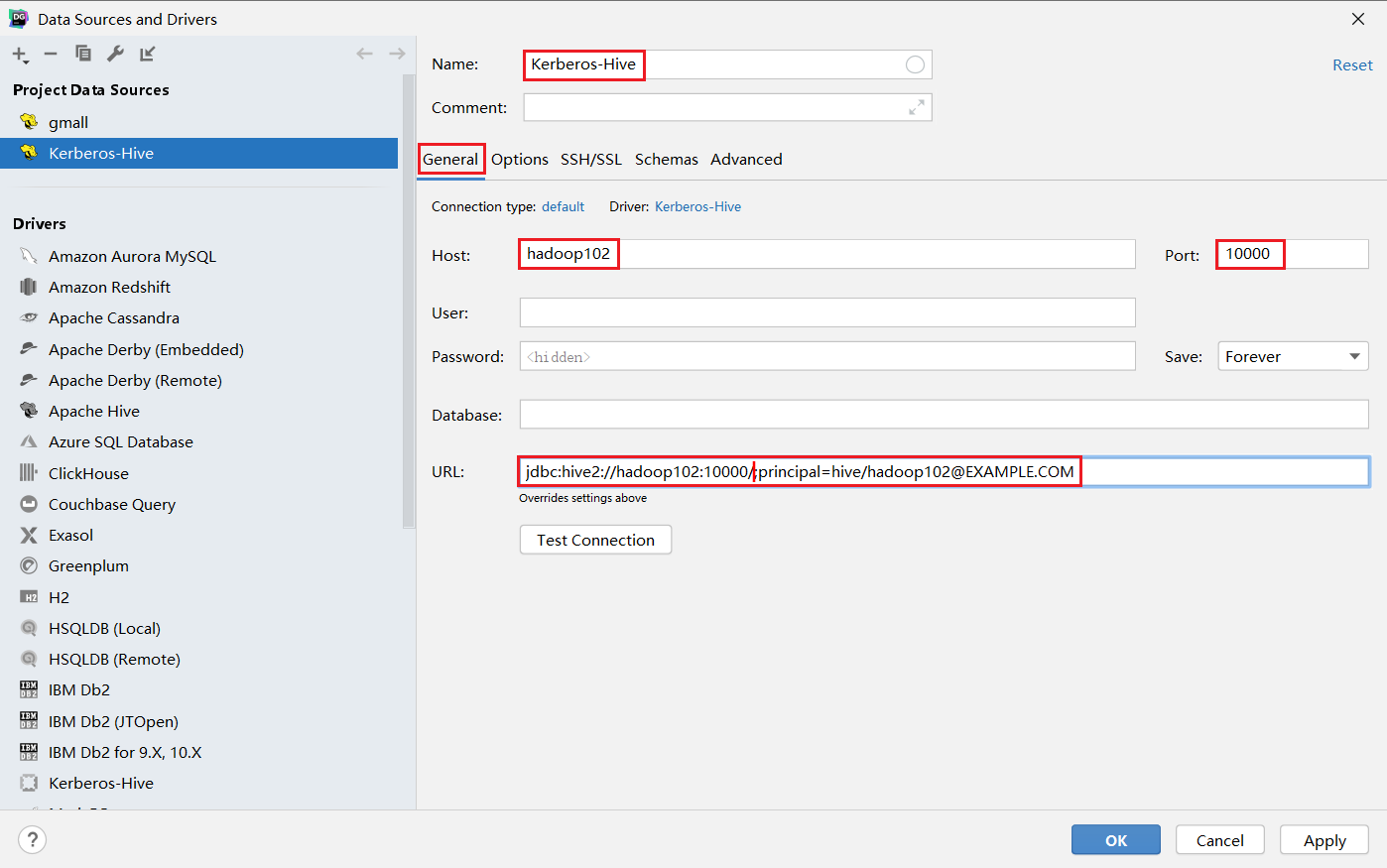
#### 2.2 新建连接

1.创建连接



2.配置连接

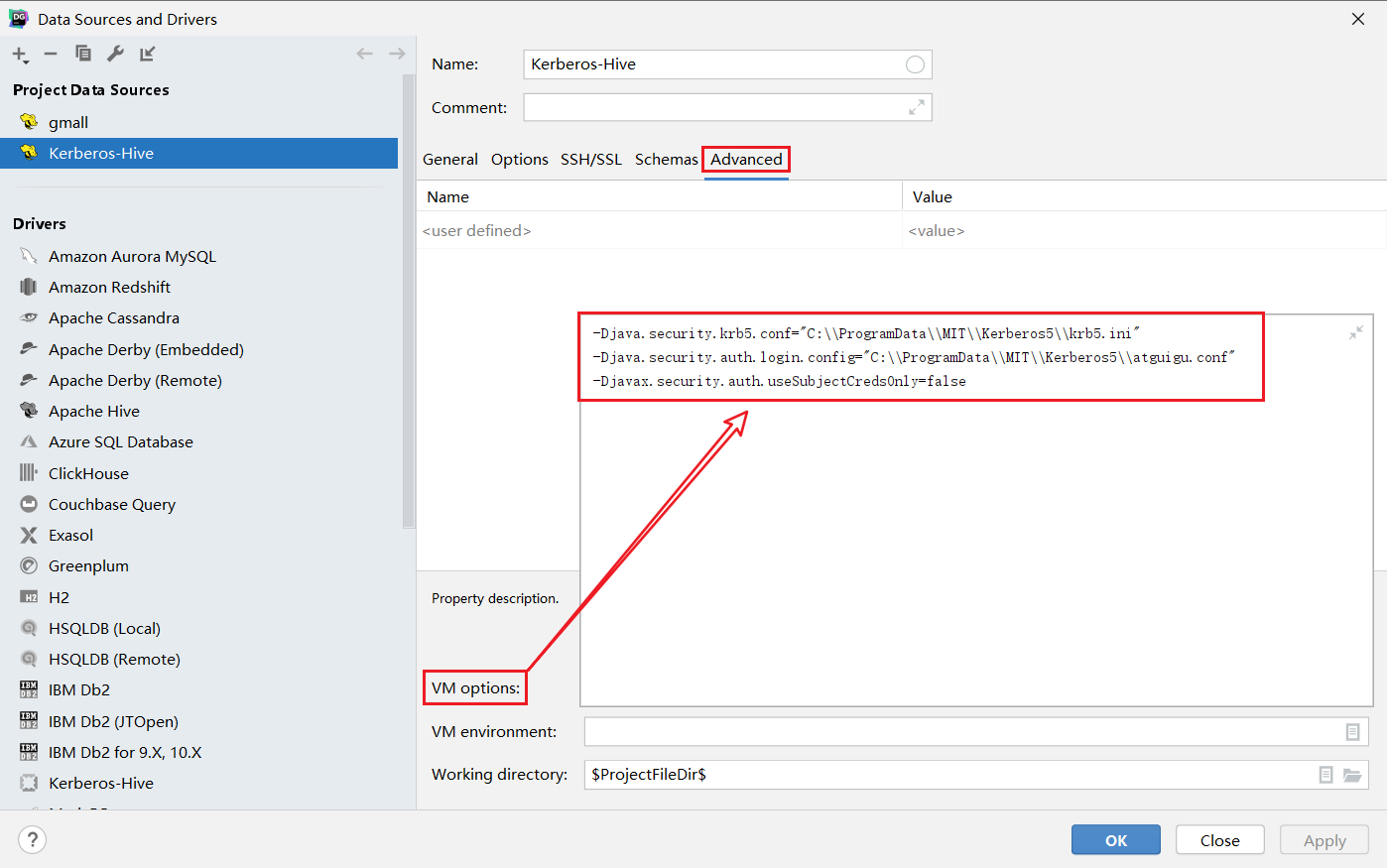
1）基础配置



**注：**

url：jdbc:hive2://hadoop01:10000/;principal=hive/hadoop01@HADOOP.COM

2）高级配置



**注：**

配置参数：

-Djava.security.krb5.conf="C:\\ProgramData\\MIT\\Kerberos5\\krb5.ini"

-Djava.security.auth.login.config="C:\\ProgramData\\MIT\\Kerberos5\\zhtx.conf"

-Djavax.security.auth.useSubjectCredsOnly=false

3）编写JAAS（Java认证授权服务）配置文件，内容如下，文件名和路径须和上图中java.security.auth.login.config参数的值保持一致。

在windows目录C:\ProgramData\MIT\Kerberos5下添加配置文件zhtx.conf，添加如下内容

com.sun.security.jgss.initiate{

com.sun.security.auth.module.Krb5LoginModule required

useKeyTab=true

useTicketCache=false

keyTab="C:\\ProgramData\\MIT\\Kerberos5\\zhtx.keytab"

principal="zhtx@HADOOP.COM";

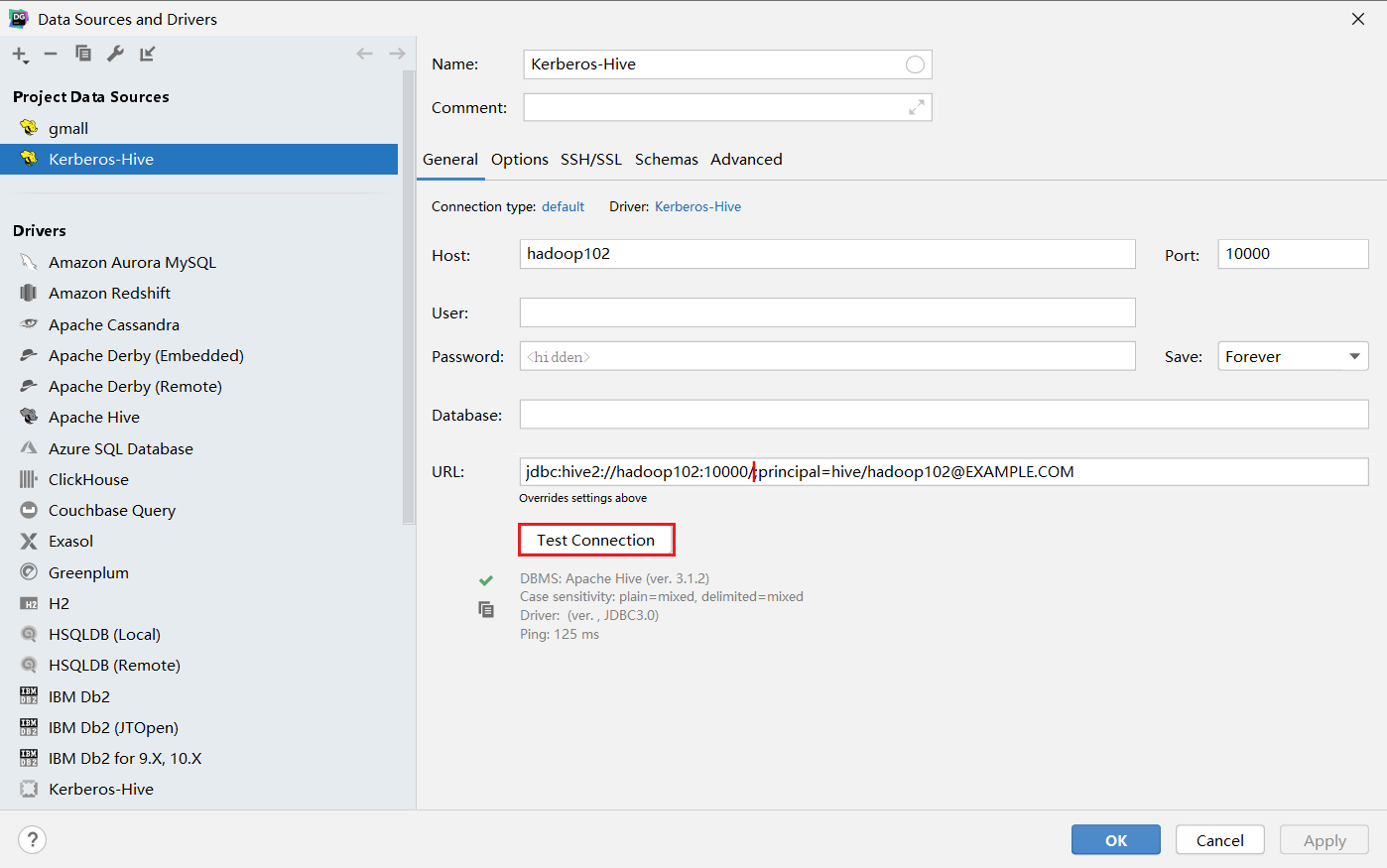
};

4）为用户生成keytab文件，在krb5kdc所在节点（hadoop02）执行以下命令

[root@hadoop02]# kadmin.local -q"xst -norandkey -k /opt/software/test/zhtx.keytab zhtx"

5）将上一步生成的zhtx.keytab文件，置于Windows中的特定路径，该路径须与**3）**中的keyTab属性的值保持一致。

6）测试连接



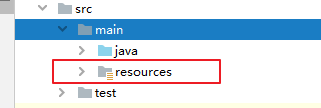
## HiveServer2 HA Kerberos示例代码

### 1 导入pom.xml依赖

<!-- hive连接 -->  
<dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-common</artifactId>  
 <version>3.1.3</version>  
</dependency>  
<dependency>  
 <groupId>org.apache.hive</groupId>  
 <artifactId>hive-jdbc</artifactId>  
 <version>3.1.2</version>  
</dependency>

### 2 将配置文件放到项目的resources目录下

用到的配置文件有krb5.conf core-site.xml hdfs-site.xml zhtx.keytab



### 3 测试代码如下

查询一张hive表的内容

import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.security.UserGroupInformation;  
  
import java.io.File;  
import java.io.IOException;  
import java.sql.\*;  
  
public class HiveHATest {  
 public static void main(String[] args) throws ClassNotFoundException, SQLException, IOException {  
 //如果不是在hadoop集群内部需要加如下参数  
 System.setProperty("java.security.krb5.conf", "src/main/resources/krb5.conf");  
  
 //声明配置  
 Configuration conf = new Configuration();  
  
 //添加hadoop配置  
 conf.addResource(new Path("src/main/resources/core-site.xml"));  
 conf.addResource(new Path("src/main/resources/hdfs-site.xml"));  
  
 //kerberos认证  
 UserGroupInformation.setConfiguration(conf);  
  
 //System.getProperty("user.dir")可以返回当前工程根目录的绝对路径  
 //在文件操作中，不用/或者\最好，推荐使用File.separator  
 UserGroupInformation.loginUserFromKeytab("zhtx",System.getProperty("user.dir")+File.separator+"src"+ File.separator+"main"+ File.separator+"resources"+ File.separator+"zhtx.keytab");  
  
 //hive 连接  
 Class.forName("org.apache.hive.jdbc.HiveDriver");

//todo 方式一  
//Connection conn = DriverManager.getConnection("jdbc:hive2://hadoop01:10000/;principal=hive/hadoop01@HADOOP.COM");  
  
//todo 方式二

Connection conn = DriverManager.getConnection("jdbc:hive2://hadoop01:2181,hadoop02:2181,hadoop03:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2\_zk;principal=hive/hadoop01@HADOOP.COM");  
 //查询hive表  
// PreparedStatement ps = conn.prepareStatement("use default ");  
 PreparedStatement ps = conn.prepareStatement("select \* from work");  
 ResultSet resultSet = ps.executeQuery();  
 while (resultSet.next()){  
 System.out.print("id: "+resultSet.getString("id"));  
 System.out.println("; name: "+resultSet.getString("name"));  
 }  
  
  
 conn.close();  
  
  
 }  
}

### 4 测试结果

代码测试结果



## HBase开启Kerberos认证

### 1 用户准备

1）在各节点创建hbase系统用户

[root@hadoop01 ~]# useradd -g hadoop hbase

[root@hadoop01 ~]# echo hbase | passwd --stdin hbase

[root@hadoop02 ~]# useradd -g hadoop hbase

[root@hadoop02 ~]# echo hbase | passwd --stdin hbase

[root@hadoop03 ~]# useradd -g hadoop hbase

[root@hadoop03 ~]# echo hbase | passwd --stdin hbase

### 2 创建hbase Kerberos主体

（1）在hadoop01节点创建主体，生成密钥文件，并修改所有者

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey hbase/hadoop01"

[root@hadoop01 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/hbase.service.keytab hbase/hadoop01"

[root@hadoop01 ~]# chown hbase:hadoop /etc/security/keytab/hbase.service.keytab

（2）在hadoop02节点创建主体，生成密钥文件，并修改所有者

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey hbase/hadoop02"

[root@hadoop02 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/hbase.service.keytab hbase/hadoop02"

[root@hadoop02 ~]# chown hbase:hadoop /etc/security/keytab/hbase.service.keytab

（3）在hadoop03节点创建主体，生成密钥文件，并修改所有者

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"addprinc -randkey hbase/hadoop03"

[root@hadoop03 ~]# kadmin -padmin/admin -w123456 -q"xst -k /etc/security/keytab/hbase.service.keytab hbase/hadoop03"

[root@hadoop03 ~]# chown hbase:hadoop /etc/security/keytab/hbase.service.keytab

### 3 修改HBase配置文件

修改$HBASE\_HOME/conf/hbase-site.xml配置文件，增加以下参数

[root@hadoop01 ~]# vim $HBASE\_HOME/conf/hbase-site.xml

#增加如下内容

<property>

<name>hbase.security.authentication</name>

<value>kerberos</value>

</property>

<property>

<name>hbase.master.kerberos.principal</name>

<value>hbase/\_HOST@HADOOP.COM</value>

</property>

<property>

<name>hbase.master.keytab.file</name>

<value>/etc/security/keytab/hbase.service.keytab</value>

</property>

<property>

<name>hbase.regionserver.kerberos.principal</name>

<value>hbase/\_HOST@HADOOP.COM</value>

</property>

<property>

<name>hbase.regionserver.keytab.file</name>

<value>/etc/security/keytab/hbase.service.keytab</value>

</property>

<property>

<name>hbase.coprocessor.region.classes</name>

<value>org.apache.hadoop.hbase.security.token.TokenProvider</value>

</property>

### 4 分发配置文件

[root@hadoop01 ~]# xsync $HBASE\_HOME/conf/hbase-site.xml

### 5 修改hbase.rootdir路径所有者

1）使用hdfs/hadoop用户进行认证

[root@hadoop01 ~]# kinit hdfs/hadoop

**注：只有hdfs/hadoop账号能够修改hdfs目录的权限**

2）修改所有者

[root@hadoop01 ~]# hadoop fs -chown -R hbase:hadoop /hbase

### 6 修改hbase进程pid存储路径

1）在hbase安装目录下，创建pids文件夹

[root@hadoop01 ~]# /opt/module/hbase-2.0.5

[root@hadoop01 hbase-2.0.5]# mkdir pids

2）修改hadoop-env.sh配置文件

[root@hadoop01 ~]# cd /opt/module/hbase-2.0.5/conf

[root@hadoop01 conf]# vim hbase-env.sh

#按如下内容修改

export HBASE\_PID\_DIR=/opt/module/hbase-2.0.5/pids

3）分发配置

[root@hadoop01 ~]# xrysrc.sh /opt/module/hbase-2.0.5/conf/hbase-env.sh

[root@hadoop01 ~]# xrysrc.sh /opt/module/hbase-2.0.5/pids

### 7 启停Hbase命令

#注:无论启停，都要保证kerberos用户身份在有效期内，否则①启动时登录访问有问题，②停止时无法关闭

[root @hadoop01 ~]$ /opt/module/hbase-2.0.5/bin/stop-hbase.sh

[root @hadoop01 ~]$ /opt/module/hbase-2.0.5/bin/start-hbase.sh

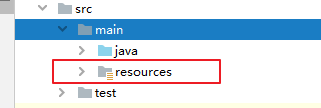
## Hbase Kerberos示例代码

### 1 导入pom.xml依赖

<!-- Hbase 连接 -->  
<dependency>  
 <groupId>org.apache.hbase</groupId>  
 <artifactId>hbase-server</artifactId>  
 <version>2.0.5</version>  
</dependency>  
  
<dependency>  
 <groupId>org.apache.hbase</groupId>  
 <artifactId>hbase-client</artifactId>  
 <version>2.0.5</version>  
</dependency>

### 2 将配置文件放到项目的resources目录下

用到的配置文件有krb5.conf core-site.xml hdfs-site.xml zhtx.keytab HbaseConfigker.properties



HbaseConfigker.properties内容

#hbase  
hbase.zookeeper.quorum=hadoop01,hadoop02,hadoop03  
hbase.zookeeper.property.clientPort=2181  
hadoop.security.authentication=kerberos  
hbase.security.authentication=kerberos  
zookeeper.znode.parent=/hbase  
hbase.site=src/main/resources/hbase-site.xml  
kerberos.keytab=src/main/resources/hbase.service.keytab  
kerberos.principal=hbase/hadoop01@HADOOP.COM  
#queryserver  
phoenix.queryserver.http.keytab.file=/etc/security/keytab/spnego.service.keytab  
phoenix.queryserver.http.kerberos.principal=HTTP/hadoop01@HADOOP.COM  
phoenix.queryserver.kerberos.allowed.realms=HADOOP.COM

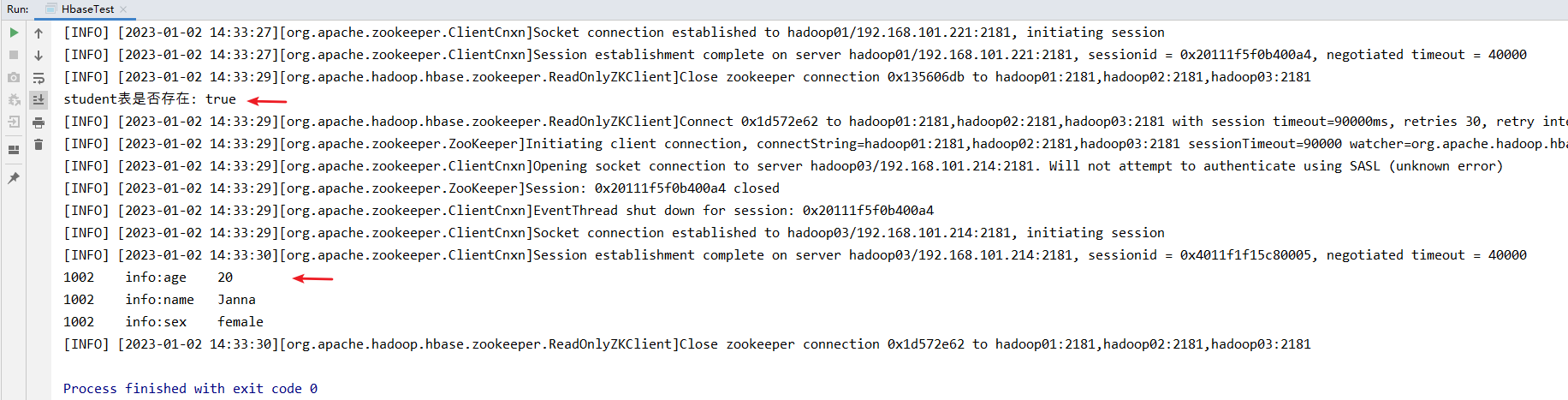
### 3 测试代码如下

判断hbase表是否存在，并扫描表数据

import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.hbase.Cell;  
import org.apache.hadoop.hbase.CellUtil;  
import org.apache.hadoop.hbase.HBaseConfiguration;  
import org.apache.hadoop.hbase.TableName;  
import org.apache.hadoop.hbase.client.\*;  
import org.apache.hadoop.hbase.util.Bytes;  
import org.apache.hadoop.security.UserGroupInformation;  
  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.Properties;  
  
  
public class HbaseTest {  
 private static String HBASE\_FILE = null;  
 private static String Authentication = null;  
 private static String zkQuorum = null;  
 private static String clientPort = null;  
 private static String principal = null;  
 private static String keytab = null;  
  
 private static Configuration configuration = null;  
  
 private static void init() throws IOException {  
 //如果不是在hadoop集群内部需要加如下参数  
 System.setProperty("java.security.krb5.conf", "src/main/resources/krb5.conf");  
  
  
 //1.创建配置信息并配置  
 configuration = HBaseConfiguration.create();  
  
 Properties clientInfo = null;  
 String userdir =  
 System.getProperty("user.dir")  
 + File.separator  
 + "src"  
 + File.separator  
 + "main"  
 + File.separator  
 + "resources"  
 + File.separator;  
 InputStream fileInputStream = null;  
 try {  
 clientInfo = new Properties();  
 String hbaseclientProp = userdir + "HbaseConfigker.properties";  
 fileInputStream = new FileInputStream(hbaseclientProp);  
 clientInfo.load(fileInputStream);  
 } catch (IOException e) {  
 throw new IOException(e);  
 } finally {  
 if (fileInputStream != null) {  
 fileInputStream.close();  
 }  
 }  
  
 HBASE\_FILE = clientInfo.getProperty("hbase.site");  
 Authentication = clientInfo.getProperty("hadoop.security.authentication");  
 zkQuorum = clientInfo.getProperty("hbase.zookeeper.quorum");  
 clientPort = clientInfo.getProperty("hbase.zookeeper.property.clientPort");  
 principal = clientInfo.getProperty("kerberos.principal");  
 keytab = clientInfo.getProperty("kerberos.keytab");  
  
 configuration.addResource(new Path(HBASE\_FILE));  
 configuration.set("hadoop.security.authentication", Authentication);  
 configuration.set("hbase.zookeeper.quorum", zkQuorum);  
 configuration.set("hbase.zookeeper.property.clientPort", clientPort);  
 configuration.set("kerberos.principal", principal);  
 configuration.set("keytab.file", keytab);  
  
 UserGroupInformation.setConfiguration(configuration);  
 UserGroupInformation.loginUserFromKeytab("zhtx", "src/main/resources/zhtx.keytab");  
  
 }  
  
 public static void main(String[] args) throws IOException {  
 init();  
  
 //判断表是否存在  
 String tableName="student";  
 boolean exist = isTableExist( tableName);  
 System.out.println(tableName + "表是否存在: " + exist);  
  
 //扫描表数据  
 scanTable(tableName);  
  
 }  
  
 //TODO 判断表是否存在  
 public static boolean isTableExist(String tableName) throws IOException {  
  
  
 //2.获取与HBase的连接  
 Connection connection = ConnectionFactory.createConnection(configuration);  
  
 //3.获取DDL操作对象  
 Admin admin = connection.getAdmin();  
  
 //4.判断表是否存在操作  
 boolean exists = admin.tableExists(TableName.valueOf(tableName));  
  
  
 //5.关闭连接  
 admin.close();  
 connection.close();  
  
 //6.返回结果  
 return exists;  
 }  
  
 //TODO 扫描数据(Scan)  
 public static void scanTable(String tableName) throws IOException {  
  
  
  
 //2.获取连接  
 Connection connection = ConnectionFactory.createConnection(configuration);  
  
 //3.获取表的连接  
 Table table = connection.getTable(TableName.valueOf(tableName));  
  
 //4.创建Scan对象  
 Scan scan = new Scan();  
  
 //5.扫描数据  
 ResultScanner results = table.getScanner(scan);  
  
 //6.解析results  
 for (Result result : results) {  
 for (Cell cell : result.rawCells()) {  
 System.out.println(  
 Bytes.toString(CellUtil.cloneRow(cell))+"\t"+  
 Bytes.toString(CellUtil.cloneFamily(cell))+":" +  
 Bytes.toString(CellUtil.cloneQualifier(cell)) +"\t" +  
 Bytes.toString(CellUtil.cloneValue(cell))  
 );  
 }  
 }  
  
 //7.关闭资源  
 table.close();  
 connection.close();  
  
 }  
  
  
}

### 4 测试结果

代码测试结果



## Phoenix开启kerberos认证

### 1修改Phoenix中的配置文件hbase-site.xml

[root@hadoop01 ~]# cd /opt/module2/phoenix-5.0.0/bin

[root@hadoop01 conf]# vim hbase-site.xml

#添加如下内容

<!-- Phoenix的kerberos认证 -->

<property>

<name>hbase.security.authentication</name>

<value>kerberos</value>

</property>

<property>

<name>hbase.master.kerberos.principal</name>

<value>hbase/\_HOST@HADOOP.COM</value>

</property>

<property>

<name>hbase.master.keytab.file</name>

<value>/etc/security/keytab/hbase.service.keytab</value>

</property>

<property>

<name>hbase.regionserver.kerberos.principal</name>

<value>hbase/\_HOST@HADOOP.COM</value>

</property>

<property>

<name>hbase.regionserver.keytab.file</name>

<value>/etc/security/keytab/hbase.service.keytab</value>

</property>

<property>

<name>hbase.coprocessor.region.classes</name>

<value>org.apache.hadoop.hbase.security.token.TokenProvider</value>

</property>

### 2 启动Phoenix

一定要保证hbase服务正常运行

[root @hadoop101 phoenix-5.0.0]$ /opt/module/phoenix-5.0.0/bin/sqlline.py hadoop01,openhadoop02,openhadoop03:2181

或者直接

[root@hadoop101 phoenix-5.0.0]$ sqlline.py

### 3为queryserver开启kerberos认证

官网部署 https://phoenix.apache.org/server.html

官方代码 <https://phoenix.apache.org/faq.html#What_is_the_Phoenix_JDBC_URL_syntax>

1）关闭hbase

[root@hadoop01 ~]$ /opt/module/hbase-2.0.5/bin/stop-hbase.sh

2）修改修改hbase中的hbase-site.xml配置文件

[root@hadoop01 ~]$ /opt/module/hbase-2.0.5/bin

[root@hadoop01 bin]# vim hbase-site.xml

#添加如下内容

<!-- kerberos之phoenix queryserver -->

<property>

<name>phoenix.queryserver.kerberos.principal</name>

<value>hbase/\_HOST@HADOOP.COM</value>

</property>

<property>

<name>phoenix.queryserver.keytab.file</name>

<value>/etc/security/keytab/hbase.service.keytab</value>

</property>

<property>

<name>phoenix.queryserver.http.keytab.file</name>

<value>/etc/security/keytab/spnego.service.keytab</value>

</property>

<!-- 这里必须写明主机地址HTTP/hadoop01，不可以使用HTTP/\_HOST -->

<property>

<name>phoenix.queryserver.http.kerberos.principal</name>

<value>HTTP/hadoop01@HADOOP.COM</value>

</property>

<!-- krb5.conf里的参数[realms] -->

<property>

<name>phoenix.queryserver.kerberos.allowed.realms</name>

<value>HADOOP.COM</value>

</property>

3）重启hbase

[root@hadoop01 ~]$ /opt/module/hbase-2.0.5/bin/start-hbase.sh

### 4修改Phoenix中的hbase-site.xml配置文件

[root@hadoop01 ~]# cd /opt/module2/phoenix-5.0.0/bin

[root@hadoop01 bin]# vim hbase-site.xml

#添加如下内容

<!-- kerberos之phoenix queryserver -->

<property>

<name>phoenix.queryserver.kerberos.principal</name>

<value>hbase/\_HOST@HADOOP.COM</value>

</property>

<property>

<name>phoenix.queryserver.keytab.file</name>

<value>/etc/security/keytab/hbase.service.keytab</value>

</property>

<property>

<name>phoenix.queryserver.http.keytab.file</name>

<value>/etc/security/keytab/spnego.service.keytab</value>

</property>

<!-- 这里必须写明主机地址HTTP/hadoop01，不可以使用HTTP/\_HOST -->

<property>

<name>phoenix.queryserver.http.kerberos.principal</name>

<value>HTTP/hadoop01@HADOOP.COM</value>

</property>

<!-- krb5.conf里的参数[realms] -->

<property>

<name>phoenix.queryserver.kerberos.allowed.realms</name>

<value>HADOOP.COM</value>

</property>

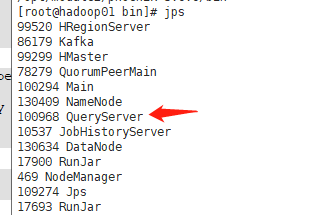
### 5 开启queryserver服务

[root@hadoop01 ~]# cd /opt/module/phoenix-5.0.0/bin

[root@hadoop01 bin]# ./queryserver.py start

查看服务已启动

[root@hadoop01 bin]# jps



Phoenix queryserver登录测试

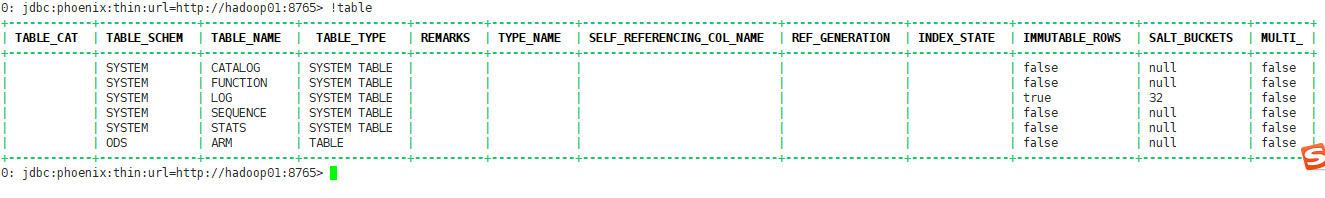
[root@hadoop01 ~]# kinit -kt /etc/security/keytab/hbase.service.keytab hbase/hadoop01

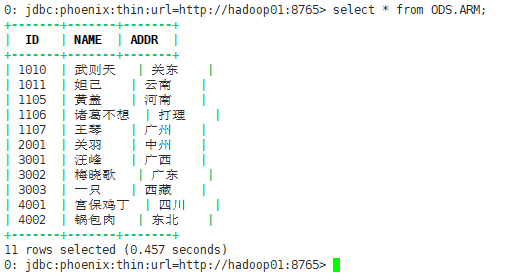
[root@hadoop01 bin]# sqlline-thin.py http://hadoop01:8765;principal="zhtx";keytab="/opt/software/zhtx.keytab"

登录成功如下



查询成功





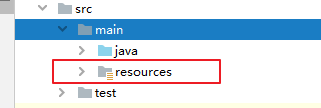
## Phoenix Thick Kerberos示例代码

### 1 导入pom.xml依赖

<!-- Phoenix Thick 连接 Phoenix Zookeeper -->  
 <dependency>  
 <groupId>org.apache.phoenix</groupId>  
 <artifactId>phoenix-core</artifactId>  
 <version>5.0.0-HBase-2.0</version>  
 <exclusions>  
 <exclusion>  
 <groupId>org.glassfish</groupId>  
 <artifactId>javax.el</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
  
 <dependency>  
 <groupId>org.glassfish</groupId>  
 <artifactId>javax.el</artifactId>  
 <version>3.0.1-b06</version>  
 </dependency>

### 2 将配置文件放到项目的resources目录下

用到的配置文件有krb5.conf hbase-site.xml zhtx.keytab HbaseConfigker.properties



HbaseConfigker.properties内容

#hbase  
hbase.zookeeper.quorum=hadoop01,hadoop02,hadoop03  
hbase.zookeeper.property.clientPort=2181  
hadoop.security.authentication=kerberos  
hbase.security.authentication=kerberos  
zookeeper.znode.parent=/hbase  
hbase.site=src/main/resources/hbase-site.xml  
kerberos.keytab=src/main/resources/hbase.service.keytab  
kerberos.principal=hbase/hadoop01@HADOOP.COM  
#queryserver  
phoenix.queryserver.http.keytab.file=/etc/security/keytab/spnego.service.keytab  
phoenix.queryserver.http.kerberos.principal=HTTP/hadoop01@HADOOP.COM  
phoenix.queryserver.kerberos.allowed.realms=HADOOP.COM

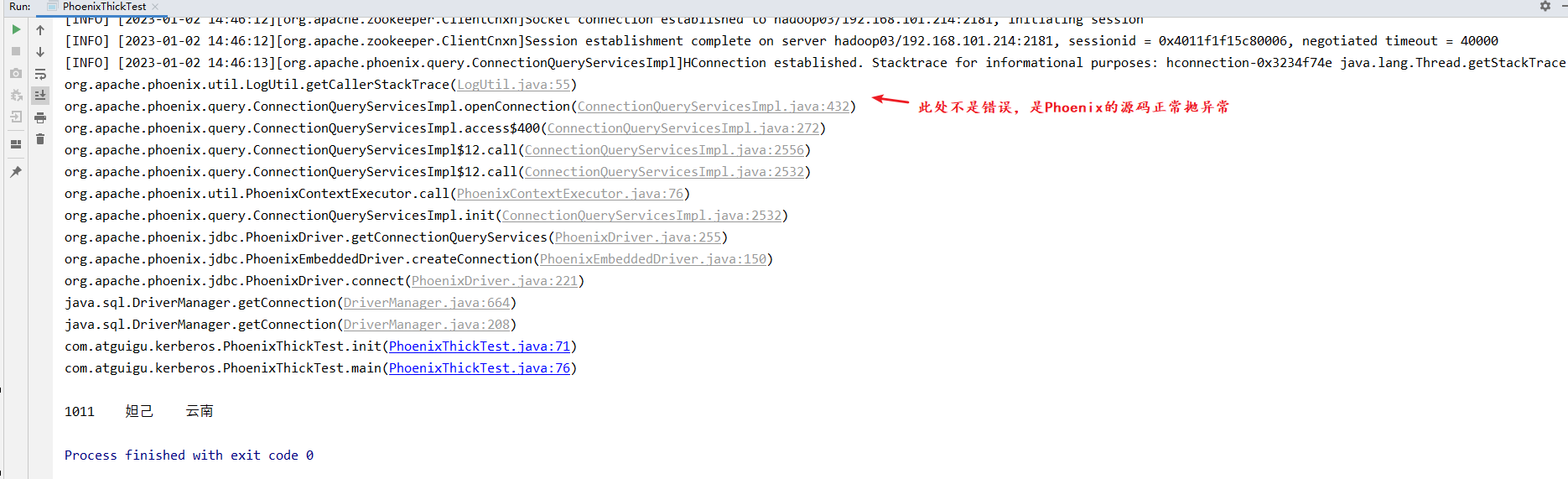
### 3 测试代码如下

查询一张Phoenix表的内容

import org.apache.phoenix.queryserver.client.ThinClientUtil;  
import org.apache.tools.ant.taskdefs.Java;  
  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.sql.\*;  
import java.util.Properties;  
  
public class PhoenixThickTest {  
 private static String HBASE\_FILE = null;  
 private static String hbaseAuthentication = null;  
 private static String hadoopAuthentication = null;  
 private static String zkQuorum = null;  
 private static String clientPort = null;  
 private static String principal = null;  
 private static String keytab = null;  
 private static String parent = null;  
 private static Properties clientInfo = null;  
 private static Connection connection;  
  
  
 private static String userName = null;  
 private static String userTab = null;  
  
 private static void init() throws IOException, SQLException {  
 //如果不是在hadoop集群内部需要加如下参数  
 System.setProperty("java.security.krb5.conf", "src/main/resources/krb5.conf");  
  
  
 String userdir =  
 System.getProperty("user.dir")  
 + File.separator  
 + "src"  
 + File.separator  
 + "main"  
 + File.separator  
 + "resources"  
 + File.separator;  
 InputStream fileInputStream = null;  
 try {  
 clientInfo = new Properties();  
 String hbaseclientProp = userdir + "HbaseConfigker.properties";  
 fileInputStream = new FileInputStream(hbaseclientProp);  
 clientInfo.load(fileInputStream);  
 } catch (IOException e) {  
 throw new IOException(e);  
 } finally {  
 if (fileInputStream != null) {  
 fileInputStream.close();  
 }  
 }  
  
 clientInfo.getProperty("hbase.site");  
 clientInfo.getProperty("hbase.security.authentication");  
 clientInfo.getProperty("hadoop.security.authentication");  
 zkQuorum = clientInfo.getProperty("hbase.zookeeper.quorum");  
 clientPort = clientInfo.getProperty("hbase.zookeeper.property.clientPort");  
 clientInfo.getProperty("kerberos.principal");  
 clientInfo.getProperty("kerberos.keytab");  
 clientInfo.getProperty("zookeeper.znode.parent");  
  
 userName="zhtx";  
 userTab=userdir+"zhtx.keytab";  
 //String url = "jdbc:phoenix:"+zkQuorum+":"+clientPort+":"+parent+":"+userName+":"+userTab;  
 String url = "jdbc:phoenix:"+zkQuorum+":"+clientPort+":"+userName+":"+userTab;  
 System.out.println("$$$$$$$$$$$"+url);  
 connection = DriverManager.getConnection(url,clientInfo);  
 }  
  
 public static void main(String[] args) throws SQLException, IOException, ClassNotFoundException {  
  
 init();  
  
 //todo zhtx 查询数据  
 String tableName="ODS.ARM";  
// thickInsert(tableName);  
  
 //todo zhtx 查询表  
 thickSelect(tableName);  
  
 }  
  
 private static void thickSelect (String tableName) throws SQLException {  
  
 PreparedStatement ps = connection.prepareStatement("select \* from "+tableName);  
 ResultSet rs = ps.executeQuery();  
 while(rs.next()){  
 System.out.println(  
 rs.getString("ID")+"\t"  
 +rs.getString("NAME")+"\t"  
 +rs.getString("ADDR")  
 );  
 }  
 //关闭  
 ps.close();  
 connection.close();  
 }  
  
 private static void thickInsert (String tableName) throws SQLException {  
 PreparedStatement ps = connection.prepareStatement("upsert into "+tableName+" values ('4002','锅包肉','东北')");  
 ps.execute();  
 connection.commit();  
 //PreparedStatement ps = connection.prepareStatement("create table if not exists bigdata.dim\_test(id varchar primary key ,info.name varchar)");  
 //ps.executeUpdate();  
  
 }  
  
  
}

### 4 测试结果

代码测试结果



## Phoenix Thin Kerberos示例代码

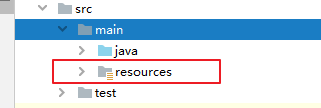
### 1 导入pom.xml依赖

**<!-- 注意：kerberos的大数据环境前提下，在同一项目下不能同时使用Phoenix Thick 和 Thin连接 -->**

<!-- Thin 连接 Phoenix queryserver -->  
<dependency>  
 <groupId>org.apache.phoenix</groupId>  
 <artifactId>phoenix-queryserver-client</artifactId>  
 <version>5.0.0-HBase-2.0</version>  
</dependency>  
  
<!-- phoenix thick连接和thin连接不兼容的部分；这个配置queryserver需要，但是与Thick连接冲突 -->  
<dependency>  
 <groupId>com.google.protobuf</groupId>  
 <artifactId>protobuf-java</artifactId>  
 <version>3.5.1</version>  
</dependency>

### 2 将配置文件放到项目的resources目录下

用到的配置文件有krb5.conf hbase-site.xml zhtx.keytab HbaseConfigThinker.properties



HbaseConfigThinker.properties内容

#hbase  
hadoop.security.authentication=kerberos  
hbase.security.authentication=kerberos  
hbase.site=src/main/resources/hbase-site.xml  
kerberos.keytab=src/main/resources/hbase.service.keytab  
kerberos.principal=hbase/hadoop01@HADOOP.COM  
#queryserver  
phoenix.queryserver.http.keytab.file=/etc/security/keytab/spnego.service.keytab  
phoenix.queryserver.http.kerberos.principal=HTTP/hadoop01@HADOOP.COM  
phoenix.queryserver.kerberos.allowed.realms=HADOOP.COM

### 3 测试代码如下

查询一张Phoenix表的内容

import org.apache.phoenix.queryserver.client.ThinClientUtil;  
  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.sql.\*;  
import java.util.Properties;  
  
  
public class PhoenixThinTest {  
 private static String HBASE\_FILE = null;  
 private static String hbaseAuthentication = null;  
 private static String hadoopAuthentication = null;  
 private static String zkQuorum = null;  
 private static String clientPort = null;  
 private static String principal = null;  
 private static String keytab = null;  
 private static String parent = null;  
 private static Properties clientInfo = null;  
 private static Connection connection;  
 private static String Authentication = null;  
  
 private static String userName = null;  
 private static String userTab = null;  
 private static String url = null;  
  
 private static void init() throws IOException, SQLException {  
 //如果不是在hadoop集群内部需要加如下参数  
 System.setProperty("java.security.krb5.conf", "src/main/resources/krb5.conf");  
  
 String userdir =  
 System.getProperty("user.dir")  
 + File.separator  
 + "src"  
 + File.separator  
 + "main"  
 + File.separator  
 + "resources"  
 + File.separator;  
 InputStream fileInputStream = null;  
 try {  
 clientInfo = new Properties();  
 String hbaseclientProp = userdir + "HbaseConfigThinker.properties";  
 fileInputStream = new FileInputStream("src/main/resources/HbaseConfigThinker.properties");  
 clientInfo.load(fileInputStream);  
 } catch (IOException e) {  
 throw new IOException(e);  
 } finally {  
 if (fileInputStream != null) {  
 fileInputStream.close();  
 }  
 }  
  
 clientInfo.getProperty("hbase.site");  
 clientInfo.getProperty("hbase.security.authentication");  
 clientInfo.getProperty("hadoop.security.authentication");  
 clientInfo.getProperty("kerberos.principal");  
 clientInfo.getProperty("kerberos.keytab");  
 clientInfo.getProperty("hadoop.security.authentication");  
  
// clientInfo.getProperty("hbase.zookeeper.quorum");  
// clientInfo.getProperty("zookeeper.znode.parent");  
// clientInfo.getProperty("hbase.zookeeper.property.clientPort");  
  
 userName = "hbase/hadoop01";  
 userTab = userdir + "hbase.service.keytab";  
 //String url = "jdbc:phoenix:"+zkQuorum+":"+clientPort+":"+parent+":"+userName+":"+userTab;  
 url = "jdbc:phoenix:thin:url=http://hadoop01:8765;serialization=PROTOBUF;authentication=SPNEGO" +";principal=" + userName + ";keytab=" + userTab+"";  
 System.out.println("$$$$$$$$$$$$$$$=="+url.replaceAll("\\\\","/"));  
  
 }  
  
 public static void main(String[] args) throws SQLException, IOException, ClassNotFoundException {  
  
 init();  
 //String connectionUrl = ThinClientUtil.getConnectionUrl("hadoop102", 8765);  
 //Class.forName("org.apache.phoenix.queryserver.client.Driver");  
   
 connection = DriverManager.getConnection(url,clientInfo);  
  
 PreparedStatement preparedStatement = connection.prepareStatement("select \* from ODS.ARM");  
  
 ResultSet resultSet = preparedStatement.executeQuery();  
  
 while (resultSet.next()) {  
 System.out.println(  
 resultSet.getString("ID") + "\t"  
 + resultSet.getString("NAME") + "\t"  
 + resultSet.getString("ADDR")  
 );  
 }  
  
 //关闭  
 connection.close();  
  
  
  
 }  
  
  
}

### 4 测试结果

代码测试结果

