[Hadoop的kerberos的实践部署](http://blog.csdn.net/xiao_jun_0820/article/details/39375819)

原文链接：https://github.com/zouhc/MyHadoop/blob/master/doc/[**Hadoop**](http://lib.csdn.net/base/hadoop)%E7%9A%84kerberos%E7%9A%84%E5%AE%9E%E8%B7%B5%E9%83%A8%E7%BD%B2.md

本文重点描述实际操作和实践部分。理论部分和介绍将一笔带过。[**测试**](http://lib.csdn.net/base/softwaretest)结果：在CM下的kerberos，遇到严重的bug不能顺畅跑通。在自己的[**hadoop**](http://lib.csdn.net/base/hadoop)下，能够顺利跑通。

**Hadoop的认证机制**

详细介绍请参考[Hadoop安全机制研究](http://dongxicheng.org/mapreduce/hadoop-security/)

[hadoop-kerberos介绍](http://dongxicheng.org/mapreduce/hadoop-kerberos-introduction/)

简单来说,没有做kerberos认证的Hadoop，只要有client端就能够连接上。而且，通过一个有root的权限的内网机器，通过创建对应的[**Linux**](http://lib.csdn.net/base/linux)用户，就能够得到Hadoop集群上对应的权限。

而实行Kerberos后，任意机器的任意用户都必须现在Kerberos的KDC中有记录，才允许和集群中其它的模块进行通信。

**Java的安全机制**

详细介绍请参考[JAAS:灵活的Java安全机制](http://docs.huihoo.com/java/j2ee/jaas.html)

简单来说,用户首先使用LoginContext的接口进行登录验证。LoginContext可以配置使用不同的验证协议。验证通过后，用户得到一个subject，里面包含凭证，公私钥等。之后，在涉及到需要进行权限认证的地方（例如，资源访问，外部链接校验，协议访问等），使用doAs函数()代替直接执行。

这样，[**Java**](http://lib.csdn.net/base/java)的权限认证就和用户的业务逻辑分离了。

//一段典型的代码如下

LoginContext lc = new LoginContext("MyExample");

try {

lc.login();

} catch (LoginException) {

// Authentication failed.

}

// Authentication successful, we can now continue.

// We can use the returned Subject if we like.

Subject sub = lc.getSubject();

Subject.doAs(sub, new MyPrivilegedAction());

**Kerberos认证协议**

Kerberos是一种网络认证协议，其设计目标是通过密钥系统为客户机 / 服务器应用程序提供强大的认证服务。

**简单介绍**

使用Kerberos时，一个客户端需要经过三个步骤来获取服务:

1. 认证：客户端向认证服务器发送一条报文，并获取一个含时间戳的Ticket-Granting Ticket（TGT）。
2. 授权：客户端使用TGT向Ticket-Granting Server（TGS）请求一个服务Ticket。
3. 服务请求：客户端向服务器出示服务Ticket，以证实自己的合法性。该服务器提供客户端所需服务，在Hadoop应用中，服务器可以是namenode或jobtracker。

为此，Kerberos需要The Key Distribution Centers（KDC）来进行认证。KDC只有一个Master，可以带多个slaves机器。slaves机器仅进行普通验证。Mater上做的修改需要自动同步到slaves。

另外，KDC需要一个admin，来进行日常的管理操作。这个admin可以通过远程或者本地方式登录。

**搭建Kerberos**

环境：假设我们有5个机器，分别是hadoop1~hadoop5。选择hadoop1,hadoop2,hadoop3组成分布式的KDC。hadoop1作为Master机器。

1.安装：通过yum安装即可，组成KDC。

yum install -y krb5-server krb5-lib krb5-workstation

2.配置：Kerberos的配置文件只有两个。在Hadoop1中创建以下两个文件,并同步/etc/krb5.conf到所有机器。

1. /var/kerberos/krb5kdc/kdc.conf:包括KDC的配置信息。默认放在 /usr/local/var/krb5kdc。或者通过覆盖KRB5\_KDC\_PROFILE环境变量修改配置文件位置。

配置示例：

[kdcdefaults]

kdc\_ports = 88

kdc\_tcp\_ports = 88

[realms]

HADOOP.COM = {

master\_key\_type = aes128-cts

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

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

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

max\_renewable\_life = 7d

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

}

说明：

HADOOP.COM:是设定的realms。名字随意。Kerberos可以支持多个realms，会增加复杂度。本文不探讨。大小写敏感，一般为了识别使用全部大写。这个realms跟机器的host没有大关系。

max\_renewable\_life = 7d 涉及到是否能进行ticket的renwe必须配置。

master\_key\_type:和supported\_enctypes默认使用aes256-cts。由于，JAVA使用aes256-cts验证方式需要安装额外的jar包。推荐不使用。

acl\_file:标注了admin的用户权限，需要用户自己创建。文件格式是

Kerberos\_principal permissions [target\_principal] [restrictions]

支持通配符等。最简单的写法是

\*/admin@HADOOP.COM \*

代表名称匹配\*/admin@HADOOP.COM 都认为是admin，权限是 \*。代表全部权限。

admin\_keytab:KDC进行校验的keytab。后文会提及如何创建。

supported\_enctypes:支持的校验方式。注意把aes256-cts去掉。

1. /etc/krb5.conf:包含Kerberos的配置信息。例如，KDC的位置，Kerberos的admin的realms 等。需要所有使用的Kerberos的机器上的配置文件都同步。这里仅列举需要的基本配置。详细介绍参考：[krb5conf](http://web.mit.edu/~kerberos/krb5-devel/doc/admin/conf_files/krb5_conf.html)

配置示例：

[logging]

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

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

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

[libdefaults]

default\_realm = HADOOP.COM

dns\_lookup\_realm = false

dns\_lookup\_kdc = false

ticket\_lifetime = 24h

renew\_lifetime = 7d

max\_life = 12h 0m 0s

forwardable = true

udp\_preference\_limit = 1

[realms]

HADOOP.COM = {

kdc = hadoop1:88

admin\_server = hadoop1:749

default\_domain = HADOOP.COM

}

[appdefaults]

说明：

[logging]：表示server端的日志的打印位置

[libdefaults]：每种连接的默认配置，需要注意以下几个关键的小配置

default\_realm = HADOOP.COM 默认的realm，必须跟要配置的realm的名称一致。

udp\_preference\_limit = 1 禁止使用udp可以防止一个Hadoop中的错误

[realms]:列举使用的realm。

kdc：代表要kdc的位置。格式是 机器:端口

admin\_server:代表admin的位置。格式是 机器:端口

default\_domain：代表默认的域名

[appdefaults]:可以设定一些针对特定应用的配置，覆盖默认配置。

1. 初始化并启动：完成上面两个配置文件后，就可以进行初始化并启动了。

A.初始化[**数据库**](http://lib.csdn.net/base/mysql):在hadoop1上运行命令。其中-r指定对应realm。

kdb5\_util create -r HADOOP.COM -s

如果遇到数据库已经存在的提示，可以把/var/kerberos/krb5kdc/目录下的principal的相关文件都删除掉。默认的数据库名字都是principal。可以使用-d指定数据库名字。(尚未测试多数据库的情况)。

B.启动kerberos。如果想开机自启动，需要stash文件。

/usr/local/sbin/krb5kdc

/usr/local/sbin/kadmind

至此kerberos，搭建完毕。

1. 搭建Slave KDCs

为了在生产环境中获得高可用的KDC。还需要搭建Slave KDCs。 TODO 经过各种努力还是不能成功同步，先放下。

1. 测试kerberos，搭建完毕后，进行以下步骤测试Kerberos是否可用。

A. 进入kadmin在kadmin上添加一个超级管理员账户，需要输入passwd

kadmin.local

addprinc admin/admin

B. 在其它机器尝试通过kadmin连接,需要输入密码

kinit admin/admin

kadmin

如果能成功进入，则搭建成功。

**kerberos日常操作**

* 管理员操作
  1. 登录到管理员账户: 如果在本机上，可以通过kadmin.local直接登录。其它机器的，先使用kinit进行验证。
  2. kadmin.local
  3. kinit admin/admin
  4. kadmin
  5. 增删改查账户:在管理员的状态下使用addprinc,delprinc,modprinc,listprincs命令。使用?可以列出所有的命令。
  6. kamdin:addprinc -randkey hdfs/hadoop1
  7. kamdin:delprinc hdfs/hadoop1
  8. kamdin:listprincs命令
  9. 生成keytab:使用xst命令或者ktadd命令
  10. kadmin:xst -k /xxx/xxx/kerberos.keytab hdfs/hadoop1
* 用户操作
  1. 查看当前的认证用户:klist
  2. 认证用户:kinit -kt /xx/xx/kerberos.keytab hdfs/hadoop1
  3. 删除当前的认证的缓存: kdestroy

**在CM上使用Kerberos认证**

在CM上使用Kerberos认证，它会帮我们创建所有的需要的Kerberos账户，并且在启动的时候自动生成keytab存放到对应的启动目录，在配置文件中添加对应的keytab文件配置和用户名。

所以，只需要给CM创建一个拥有管理员权限的账户。CM就能够完成大部分的初始化工作。

**初始化部署**

1. 为CM添加一个账户，并生成keytab文件

kadmin kadmin:addprinc -randkey cloudera-scm/admin@HADOOP.COM kadmin:xst -k cmf.keytab cloudera-scm/admin@HADOOP.COM

1. 将上文产生的keytab文件移到cloudera-scm的配置目录，添加cmf.principal文件并写入账户的名称，最后修改文件权限。
2. mv cmf.keytab /etc/cloudera-scm-server/
3. echo "cloudera-scm/admin@HADOOP.COM" >> /etc/cloudera-scm-server/cmf.principal
4. chown cloudera-scm:cloudera-scm cmf.keytab
5. chmod 600 cmf.keytab
6. chown cloudera-scm:cloudera-scm cmf.principal
7. chmod 600 cmf.principal

默认配置目录在/etc/cloudera-scm-server/,但是我们修改为/home/cloudera-manager/cm-4.6.3/etc/cloudera-scm-server/

1. 设置CM的default Realm ：在界面上顶部的Administrator-setting-security-Kerberos Security Realm 填入 HADOOP.COM
2. 针对所有服务开启security选项
   * Zookeeper:
     + 勾选 Zookeeper Service > Configuration > Enable Zookeeper Security
   * HDFS:
     + 勾选 HDFS Service > Configuration > Authentication
     + 勾选 HDFS Service > Configuration > Authorization
     + 修改Datanode Transceiver Port 到1004
     + 修改Datanode HTTP Web UI Port 到1006
   * HBASE：
     + 勾选HBase Service > Configuration > Authentication
3. 启动即可

重大的bug: 当我在测试机上成功跑通之后，重新删除了kerberos的数据库后。关闭掉所有服务的安全选项。重新启动后，Generate Credentials不能成功创建账户。而且也连接不到已经存在的账户的内容。第二天回来，发现创建了少量的账户YARN和mapred的账户。但是其它的账户都没有。猜测：可能是因为增加了两个账户分别是 krbtgt/HADOOP.COM@HADOOP.COM krbtgt/hadoop1@HADOOP.COM 根据数据库的结构分析，怀疑cloudera manager把keytab都保存了。所以，不再重新产生keytab。 keytab不能重新生成是一个大问题。

**非CM下的keytab配置**

检查：如果JAVA的版本在1.6.21或以前的,会遇到客户端需要renew ticket,才能通过认证。而renwe ticket必须保证kdc的配置文件包含max\_renewable\_life = 7d项。

**创建账户**

创建所有账户,生成keytab(我们使用hadoop账户启动所有的服务，所以，只生成hadoop和HTTP账户就足够了)

kadmin:addprinc -randkey hadoop/hadoop1@HADOOP.COM

...

kadmin:addprinc -randkey hadoop/hadoop5@HADOOP.COM

kadmin:addprinc -randkey HTTP/hadoop1@HADOOP.COM

...

kadmin:addprinc -randkey HTTP/hadoop5@HADOOP.COM

kadmin:xst -k /xxx/hadoop.keytab hadoop/hadoop1 HTTP/hadoop1

...

kadmin:xst -k /xxx/hadoop.keytab hadoop/hadoop5 HTTP/hadoop5

说明：一共添加了10个账户分别是hadoop的hadoop1到hadoop5的账户和HTTP的hadoop1到hadoop5的账户。导出账户的时候，把hadoop1机器的hadoop账户和HTTP账户导入到同一个keytab文件中。

在标准的情况中，依据不同服务的启动者的不同，会创建不同的账户，导出不同的keytab文件。由于我们使用的是hadoop用户启动所有服务的状况，所以一个hadoop.keytab就足够使用了。如果像ClouderaManager那样的一个用户启动一种服务，就要创建不同的用户，导出不同的keytab。例如:hadoop1的zookeeper配置文件中需要zookeeper.keytab，当中含有zookeeper/hadoop1这个账户

下文提到的配置文件中添加keytab文件，都要求不同机器含有对应的机器名和启动用户的keytab文件。要测试这个机器的keytab文件是否可用，可使用以下命令进行测试:

kinit -kt /xx/xx/hadoop.keytab hadoop/hadoop1

klist

**为ZK添加认证**

* 修改zoo.cfg添加配置
  + authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
  + jaasLoginRenew=3600000
* 在配置目录中添加对应账户的keytab文件且创建jaas.conf配置文件,内容如下：

Server {

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

useKeyTab=true

keyTab="/XX/XX/hadoop.keytab"

storeKey=true

useTicketCache=true

principal="hadoop/hadoop3@HADOOP.COM";

};

其中keytab填写真实的keytab的绝对路径，principal填写对应的认证的用户和机器名称。

* 在配置目录中添加java.env的配置文件，内容如下：

export JVMFLAGS="-Djava.security.auth.login.config=/xx/xx/jaas.conf"

* 每个zookeeper的机器都进行以上的修改
* 启动方式和平常无异，如成功使用安全方式启动，日志中看到如下日志：

2013-11-18 10:23:30,067 ... - successfully logged in.

**为HDFS添加认证**

* 增加基本配置包括各种的princal和keytab文件的配置(生成的hdfs的keytab和HTTP的keytab最好放一起，容易配置。下面的配置中keytab文件使用绝对路径，principal使用\_HOST，Hadoop会自动替换为对应的域名。）。
  + core-site.xml
    - hadoop.security.authorization：true
    - hadoop.security.authentication：kerberos
  + hdfs-site.xml
    - dfs.block.access.token.enable：true
    - dfs.namenode.keytab.file: /xx/xx/hadoop.keytab
    - dfs.namenode.kerberos.principal: hadoop/\_HOST@HADOOP.COM
    - dfs.namenode.kerberos.internal.spnego.principal: HTTP/\_HOST@HADOOP.COM
    - dfs.datanode.keytab.file: /xx/xx/hadoop.keytab
    - dfs.datanode.kerberos.principal: hadoop/\_HOST@HADOOP.COM
    - dfs.datanode.address: 1004 (小于1024）
    - dfs.datanode.http.address: 1006 (小于1024）
    - dfs.journalnode.keytab.file: /xx/xx/hadoop.keytab
    - dfs.journalnode.kerberos.principal: hadoop/\_HOST@HADOOP.COM
    - dfs.journalnode.kerberos.internal.spnego.principal: HTTP/\_HOST@HADOOP.COM
  + hadoop-env.sh
  + export HADOOP\_SECURE\_DN\_USER=hadoop
  + export HADOOP\_SECURE\_DN\_PID\_DIR=/home/hadoop/hadoop/pids
  + export HADOOP\_SECURE\_DN\_LOG\_DIR=/home/hadoop/hadoop/logs
  + export JSVC\_HOME=/usr/bin
  + #如果root下没有JAVA\_HOME配置，则需要指定JAVA\_HOME
  + export JAVA\_HOME=/home/hadoop/java/jdk
* 启动：设置了Security后，NameNode，QJM，ZKFC可以通过start-dfs.sh启动。DataNode需要使用root权限启动。设置了HADOOP\_SECURE\_DN\_USER的环境变量后，start-dfs.sh的启动脚本将会自动跳过DATANODE的启动。所以，整个启动过程分为以下两步：
  + 启动NameNode，QJM，ZKFC
  + start-dfs.sh

说明:查看QJM的日志和ZKFC的日志。检查有无exception。QJM的报错不会有明显的提示。如果启动不成功检查以下几点是否做好：

* + - QJM和NameNode对应的keytab文件是否包含hadoop账户和HTTP账户对应该机器的kerberos账户。
    - keytab使用绝对路径，可以避免一些问题。

疑惑：ZKFC中有日志，但是工作正常，大胆预测连接zookeeper不需要强制通过jaas验证。TODO：验证此猜想。

INFO org.apache.zookeeper.ClientCnxn: Opening socket connection to server hadoop3/10.1.74.46:59181. Will not attempt to authenticate using SASL (无法定位登录配置)

* + 启动DataNode：
    - 配置JSVC：DataNode需要JSVC启动。首先安装JSVC，然后配置的hadoop-env.sh的JSVC\_HOME变量。JSVC运行还需要一个commons-daemon-xxx.jar包。从[commons/daemon](http://archive.apache.org/dist/commons/daemon/binaries/)下载一个最新版本的jar包。当前，JSVC启动的时候遇到一个奇怪的bug，就是JSVC的classpath不支持\*匹配。详细修改如下：
    - #添加commons-daemon的jar包,并替换路径为绝对路径
    - export CLASSPATH=$CLASSPATH:/xxx/commons-daemon-1.0.15.jar
    - temp=${CLASSPATH//':'/' '}
    - t=`echo $temp`
    - export CLASSPATH=${t//' '/':'}
    - mv问题:由于权限问题，在移动日志文件启动的时候，会询问是否覆盖只读的日志文件。这个会导致使用start-secure-dns.sh启动的时候不顺畅。推荐修改hadoop-daemon.sh的74行:
    - mv "$log" "$log.$num"; -->修改为--> mv -f "$log" "$log.$num";
    - 启动：
      * 切换到root用户，需要配置这个root用户免密码登陆到其它的机器。
      * #自动登陆并启动datanode
      * sh /home/xx/hadoop/sbin/start-secure-dns.sh
      * 否则，需要单独登陆到所有机器启动datanode。
      * #如果单独登陆启动datanode
      * sh /home/xx/hadoop/sbin/hadoop-daemon.sh datanode start
* 测试：使用任意用户通过keytab文件进行认证，运行hdfs相关命令。
* kinit -kt /xx/xx/qiujw/keytab qiujw/hadoopN
* #对于java1.6\_26以下版本的需要renew ticket
* kinit -R
* klist
* hdfs dfs -ls /tmp

**为YARN添加认证配置**

* 添加配置
  + yarn.xml:
    - yarn.resourcemanager.keytab:/xx/xx/hadoop.keytab
    - yarn.resourcemanager.principal:hadoop/\_HOST@HADOOP.COM
    - yarn.nodemanager.keytab:/xx/xx/hadoop.keytab
    - yarn.nodemanager.principal:hadoop/\_HOST@HADOOP.COM
    - yarn.nodemanager.container-executor.class:org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor
    - yarn.nodemanager.linux-container-executor.group:hadoop
  + mapred.xml:
    - mapreduce.jobhistory.keytab:/xx/xx/hadoop.keytab
    - >mapreduce.jobhistory.principal:hadoop/\_HOST@HADOOP.COM
* 修改[**Container**](http://lib.csdn.net/base/docker)-executor.conf.dir，重新编译container-executor：

cp ~/hadoop/src

mvn package -Pdist,native -DskipTests -Dtar -Dcontainer-executor.conf.dir=/etc

cp ./hadoop-yarn-project/target/hadoop-yarn-project-2.0.0-cdh4.2.1/bin/container-executor ~/hadoop/bin

#以下命令查看编译是否成功

strings ~/hadoop/bin/container-executor|grep etc

#修改权限

sudo chown root:hadoop /xx/hadoop/bin/container-executor

sudo chmod 4750 /xx/hadoop/bin/container-executor

说明：为什么要编译container-executor?

答：因为container-executor要求container-executor.cfg这个文件及其所有父目录都属于root用户，且权限小于755。配置文件container-executor.cfg默认的路径在../etc/hadoop/container-executor.cfg。如果，按照默认的路径修改所有父目录都属于root，显然不现实。于是，把路径编译到/etc/container-executor.cfg中。

* 创建/etc/container-executor.cfg文件,文件内容如下：

#运行container的用户

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

#这个是允许运行应用的用户列表，默认是全部可以运行

#banned.users=

#这个是允许提交job的最小的userid的值。centos中一般用户的id在500以上。

min.user.id=500

修改/etc/container-executor.cfg的权限

sudo chown root:root /etc/container-executor.cfg

sudo chmod 600 /etc/container-executor.cfg

* 启动，使用hadoop用户直接启动即可

start-yarn.sh

* 检查Nodemanager和Resourcemanager的日志是否有异常。
  + 一般异常都是因为container-executor.cfg的权限和container-executor的权限问题。请仔细核对：
  + [hadoop@hadoop2 hadoop]$ ls ~/hadoop/bin/container-executor -l

-rwsr-x--- 1 root hadoop 89206 Nov 18 16:18 /home/hadoop/hadoop/bin/container-executor [hadoop@hadoop2 hadoop]$ ls /etc/container-executor.cfg -l -rw------- 1 root root 240 Nov 18 16:31 /etc/container-executor.cfg

* 测试：使用任意用户通过keytab文件进行认证，运行yarn相关命令。

kinit -kt /xx/xx/qiujw/keytab qiujw/hadoopN

#对于java1.6\_26以下版本的需要renew ticket

kinit -R

klist

yarn jar /xx/xx/hadoop-mapreduce-examples-xx.jar pi 10 100

**为hbase添加认证**

* 添加配置：
  + hbase-site.xml:以下添加到client和server端
    - hbase.security.authentication：kerberos
    - hbase.rpc.engine: org.apache.hadoop.hbase.ipc.SecureRpcEngine
  + hbase-site.xml:以下添加到server端
    - hbase.regionserver.kerberos.principal：hadoop/\_HOST@HADOOP.COM
    - hbase.regionserver.keytab.file: /xx/xx/hadoop.keytab
    - hbase.master.kerberos.principal: hadoop/\_HOST@HADOOP.COM
    - hbase.master.keytab.file: /xx/xx/hadoop.keytab
* 添加[**Hbase**](http://lib.csdn.net/base/hbase)连接secure的zookeeper：
  + 创建zk-jaas.conf配置文件，内容如下：
  + Client {
  + com.sun.security.auth.module.Krb5LoginModule required
  + useKeyTab=true
  + useTicketCache=false
  + keyTab="/xx/hadoop.keytab"
  + principal="hadoop/hadoopN@HADOOP.COM";
  + };
  + 修改[**hbase**](http://lib.csdn.net/base/hbase)-env.sh:
  + export HBASE\_OPTS="$HBASE\_OPTS -Djava.security.auth.login.config=/xx/zk-jaas.conf"
  + export HBASE\_MANAGES\_ZK=false
  + 确保以下配置项是正确的:
    - hbase-site.xml:
      * hbase.zookeeper.quorum: hadoopN,...,hadoopN
      * hbase.cluster.distributed： true
  + 添加以下项目到zoo.cfg中：
    - kerberos.removeHostFromPrincipal: true
    - kerberos.removeRealmFromPrincipal: true
* 启动：如往常启动即可
* start-hbase.sh
* TroubleShooting

笔者在启动hbase后，在zookeeper的日志中大量发现这种信息：

Client failed to SASL authenticate: javax.security.sas l.SaslException: GSS initiate failed [Caused by GSSException: Failure unspecified at GSS-API level (Mechanism level: Specified version of key is not available (44 ))]

在多次调整无果后，怀疑是因为我的一些老旧的账户的renewmax属性还是0.于是，把所有相关账户都删除，生成后，再次启动。这个错误就消失了。

**Hbase的权限控制**

* 启动hbase的用户是超级用户拥有所有的权限。
* hbase支持4个权限
  + R ：读权限 Get, Scan, or Exists calls
  + W ：写权限 Put, Delete, LockRow, UnlockRow, IncrementColumnValue, CheckAndDelete, CheckAndPut, Flush, or Compact
  + C ：创建权限 Create, Alter, or Drop
  + A ：管理员权限 Enable, Disable, MajorCompact, Grant, Revoke, and Shutdown.
* 权限控制语句：
* grant <user> <permissions>[ <table>[ <column family>[ <column qualifier> ] ] ]
* revoke <user> <permissions> [ <table> [ <column family> [ <column qualifier> ]]]
* alter <table> {OWNER => <user>} # sets the table owner
* user\_permission <table> # displays existing permissions
* 创建表的用户拥有该表的所有权限
* 如果赋予权限的时候没有针对某个表或者CF进行赋予，就会对全局获得权限。请小心。

**Hive的权限**

* Hive的客户端的权限和普通的客户端的一致就可以了。

**客户端配置**

使用者要和实行了kerberos的集群进行通信。要kerberos的管理员创建对应的账户。并且生成keytab返回给使用者，使用者通过kinit命令认证后，就跟平常使用Hadoop的方式一致地使用即可。以下是一个例子：

kadmin:addprinc qiujw/hadoop1

kadmin:xst -k qiujw.keytab qiujw/hadoop1

#将qiujw.keytab交给用户

#在hadoop1机器上

kinit -kt qiujw.keytab qiujw/hadoop1

klist

Ticket cache: FILE:/tmp/krb5cc\_512

Default principal: qiujw/hadoop2@HADOOP.COM

Valid starting Expires Service principal

11/19/13 10:53:54 11/20/13 10:53:54 krbtgt/HADOOP.COM@HADOOP.COM

renew until 11/26/13 10:44:10

说明：Expires下面的是这个认证的过期的日志。renew until后面的是续约期。

意思是,如果这个缓存过了认证的过期时间，就会失效。在续约期期间通过使用kinit -R可以续约这个认证。但是，过了续约期。必须要使用keytab重新认证。

Hadoop等的服务中，都会使用keytab自动做续约不会存在过期的情况。如果客户端需要长久运行不过期，需要在程序中使用keytab做认证。

**协议控制**

Hadoop的框架中支持针对不同的协议开启权限控制。不再本次探究范围内。[服务协议控制](http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ServiceLevelAuth.html)