一：Kerberos简介

在大数据领域，安全永远是一个绕不开的话题。

对于一个简单安装上线的 `hadoop` 集群，我们可以认为有如下安全隐患：

如，可以人为的添加一个客户端节点，并以此假冒的客户端来获取集群数据。对于一个假冒的客户端节点，成功加入集群就能够伪装 `datanode` 让得到 `namenode` 指派的任务和数据。创建一个HDFS账户，就可以得到 `hadoop` 文件系统的最高权限。

`Kerberos` 主要用来做网络通讯中的身份认证，帮助我们高效、安全的识别访问者。

那么 `Kerberos` 是如何做身份认证的呢？

我们来看一个现实中的例子：

> 小明要去电影院观看一场电影
>
> 那么对于这样一个流程来说就有：
>
> 1. 前期需求，确定了自己想要看什么电影，位于哪个影院，什么时间后使用自己的账户密码登录票务中心
> 2. 购票机制，通过付费（发送请求）来让小明从未授权的影院访问者变成被授权访问的状态
> 3. 验票机制，验证票据持有者的身份，和票务中心核对验证票据的合法性、时间、以及访问的位置
> 4. 观看电影，一切验证通过后得到想要的内容。
> 5. 再次观看，需要重新购票走流程

那么对于这样一个例子，相信大家应该都很好理解。

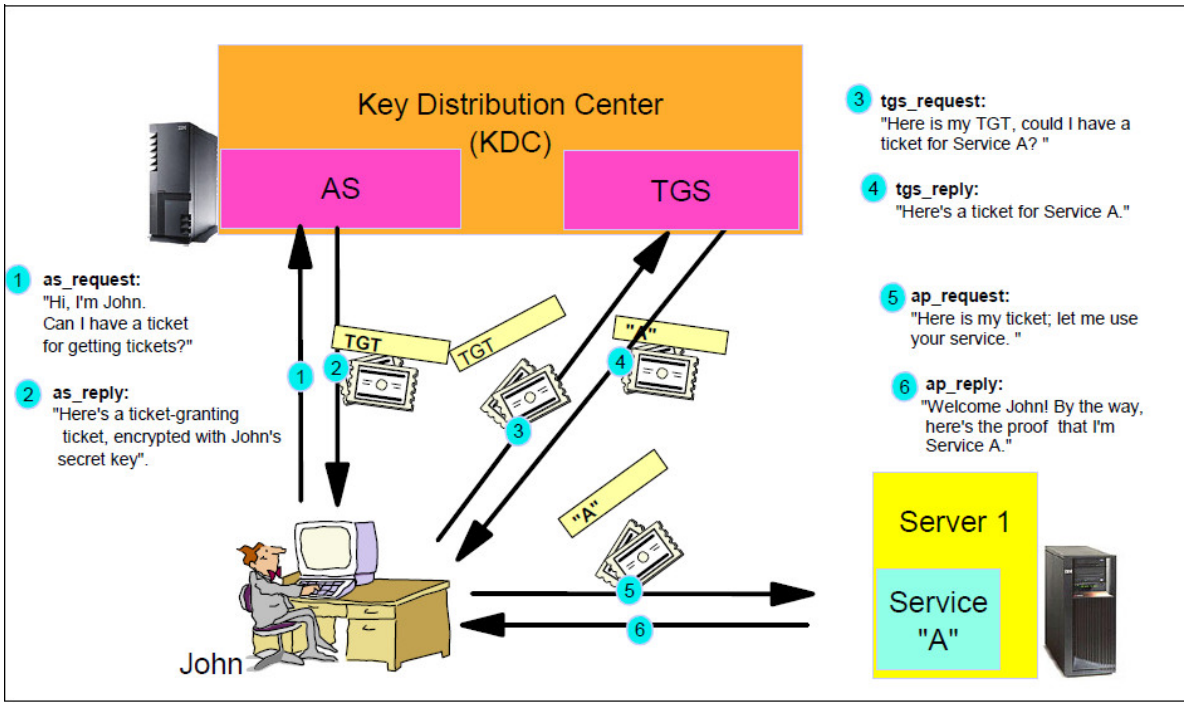
Kerberos的认证流程基本上和上述的例子差不多，我们来对上述例子进行一个转换一一对应：

> 1. 发送请求，表明要访问什么服务，使用自己的密码来对请求进行加密
> 2. 验证身份后，得到一个ticket（票据）
> 3. 服务提供者和Kerberos进行通讯验证ticket的合法性、有效期
> 4. 验证通过提供服务
> 5. 超出ticket的有效期后再次访问需要重新申请

基于这样一个转换，我们可以得到一个关键信息： `Kerberos` 的身份认证其实是基于 `ticket` 来完成的，就像看电影是基于电影票来进行验证的一样。

那么我们客户端，想要访问某些服务，最主要的是得到一张 `ticket`

理解了上述的概念之后，来看一下具体的 `Kerberos` 的执行流程：



在如上的流程里有如下关键字：

1. `KDC`（秘钥分发中心），`KDC`（也就是`Kerberos Server`）提供提供`AS`和`TGS`两个服务
    1. AS：`authorization server`,授权服务，对于上面流程1，提供初始授权认证，用户表明需求并使用密码对请求进行加密，AS用提供的密码对请求进行解密后得到请求内容，返回给用户一个TGT（ticket granting tickets)(用一个秘钥加密)
    2. 用户得到TGT后使用TGT去访问TGS(`Ticket Granting Server`),TGS验证TGT后（使用秘钥解密）返回一个Ticket给用户
2. 用户（图中John），得到ticket后去访问server，server收到ticket和KDC进行验证，通过后提供服务

如上是一个典型的Kerberos运行流程，对于`hadoop`的授权认证来说，就是把server换为具体的服务，如`namenode` `resourcemanager`等

想要访问`namenode`（也就是`hdfs`）需要拿到对应`hdfs`的`ticket`才可以访问

对`Kerberos`有了基本了解后，我们开始进入安装部署的步骤，首先是先准备基本的服务器环境

二：环境准备

1. 各软件版本信息

本次课程基于使用的软件环境如下：

- CDH 5.14.4
    - hadoop2.6.0-cdh5.14.4
    - hive1.1.1-cdh5.14.4
- CentOS 6.9 x64 操作系统
- JDK 1.8 and jdk1.7(for maven)

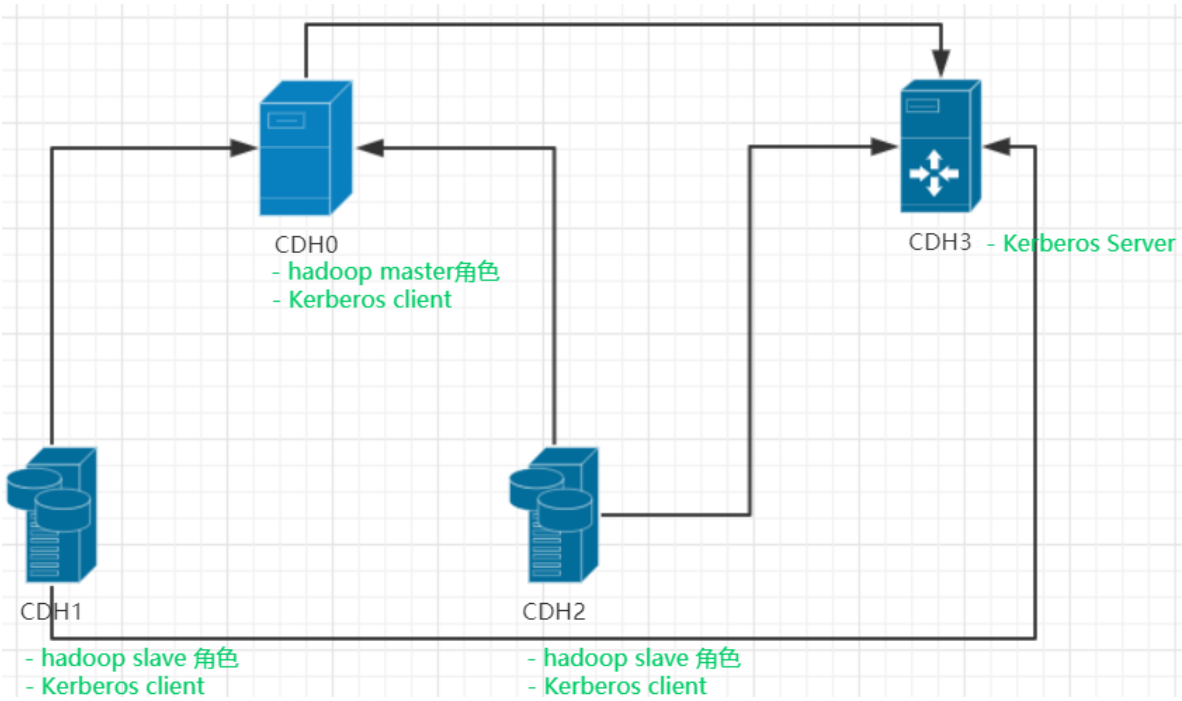
- apache-maven-3.0.5
- Kerberos 使用在线yum库提供的最新版本

> 除操作系统外，所有需要用到的安装包，在课件内都有提供，同学们可以直接使用

2. 部署节点信息

本次课程使用4台服务器来进行演示，分别是：

- 主机名cdh0（FQDN: cdh0.itcast.cn）：hadoop的master节点，部署 `namenode`，`resourcemanager`，`hive server2`，`historyserver` 等master角色 以及 `Kerberos 的 客户端`
- 主机名cdh1（FQDN: cdh1.itcast.cn）：hadoop的slave节点，部署 `datanode`，`nodemanager` 等slave角色 以及 `Kerberos 的 客户端`
- 主机名CDH2（FQDN: cdh2.itcast.cn）：同CDH1
- 主机名CDH3（FQDN: cdh3.itcast.cn）：Kerberos的server端，部署 `Kerberos server`



硬件配置：

CDH0、CDH1、CDH2均为2核心3GB内存配置，CDH3为1核心1GB内存

同学们可以根据自己主机的配置情况做相应增删

其中slave节点可以删除一个，比如CDH2可以删除不要，以节省内存开销

如果内存比较大的同学可以给slave节点配置三个，来模拟更加真实的环境

3. 基础系统环境准备

a. 关闭防火墙服务

`chkconfig iptables off` 关闭防火墙自启

`service iptables stop` 关闭当前防火墙的运行

b. 配置主机名

修改 `/etc/sysconfig/network` 文件，设置HOSTNAME为需要的主机名

这里设置FQDN形式，保存即可

> FQDN形式的主机名就是包含完整的主机所在域（所在组织）的主机名
>
> 如 cdh0.itcast.cn 其中cdh0是简写的主机名 .itcast.cn 是这台主机的所在域 也就是domain
>
> hadoop中配置建议使用FQDN形式的主机名

c. 配置各个主机IP地址

> 老师演示的网段是 192.168.66.0 网段
>
> 下面演示的IP地址根据你虚拟机的网段来设置，自行修改

1. 关闭 `NetworkManager` 服务和 `selinux`

```
# NetworkManager
service NetworkManager stop # 关闭
chkconfig NetworkManager off # 关闭开机自启

# Selinux
vim /etc/sysconfig/selinux
设置SELINUX=disabled
然后重启生效
```

2. 配置 `ip` : `vim /etc/sysconfig/network-scripts/ifcfg-eth0`

   在其中确保有如下:

```
DEVIDE="eth0" # 网卡名字
BOOTPROTO="static" # 静态IP
ONBOOT="yes" # 开机启动
TYPE="Ethernet" # 类型
IPADDR=192.168.66.200 # IP地址
NETMAST=255.255.255.0 # 子网掩码
GATEWAY=192.168.66.2 # 网关地址
DNS1=192.168.66.2 # DNS1的地址
```

3. 执行 `service network restart` 重启网卡

   如果遇到 `Device eth0 does not seem to be present, delaying initialization.` 错误，可以执行: `rm /etc/udev/rules.d/70-persistent-net.rules` 然后重启机器即可

   重启可以执行 `init 6`

4. 执行 `curl www.baidu.com` 来判断自己是否可以上网

d. 配置 `/etc/hosts` 文件

> 执行到这一步网络就配置好了，可以用secureCRT等工具连接服务器了

`vim /etc/hosts`

添加:

```
192.168.66.200 cdh0.itcast.cn cdh0
192.168.66.201 cdh1.itcast.cn cdh1
192.168.66.202 cdh2.itcast.cn cdh2
192.168.66.203 cdh3.itcast.cn cdh3
```

在每个机器都执行， 不要动原本的localhost的配置，可能会有bug

e. 安装必要软件（通过 `yum` 在线安装）

执行:

```
yum -y install epel-release
yum install -y autoconf automake libtool cmake ncurses-devel openssl-devel lzo-devel
zlib-devel gcc gcc-c++ bzip2-devel cmake3 lrzsz ntp
```

所有机器上都执行

f. 配置ntp和时间

1. 修改时区: `cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime`
2. 校对当前时间: `ntpdate -u 0.cn.pool.ntp.org`
3. 设置ntp服务并启动和设置自启

```
vim /etc/ntp.conf
在server那里最上面添加一行
server 0.cn.pool.ntp.org iburst

然后执行service ntpd start
执行 chkconfig ntpd on
```

所有机器上都执行

g. 创建用户和组以及配置各自的ssh免密登录

添加用户

```
# 添加一个hadoop组
groupadd hadoop
# 添加用户
useradd hadoop -g hadoop -p hadoop
useradd hive -g hadoop -p hive
useradd yarn -g hadoop -p yarn
useradd mapred -g hadoop -p mapred
或者一条命令搞定：
groupadd hadoop;useradd hadoop -g hadoop -p hadoop;useradd hive -g hadoop -p
hive;useradd yarn -g hadoop -p yarn;useradd mapred -g hadoop -p mapred
```

在cdh0 cdh1 cdh2三台机器上执行

设置ssh免密登录（可选）

> 可选，方便操作，课程演示做root账户的互相免密

执行 `ssh-keygen -t rsa` 一路回车就可创建当前账户的 `ssh key`

执行 `ssh-copy-id 目标主机` 即可让当前机器的当前用户免密登录目标主机的当前用户

h. 上传Hadoop软件包

上传包

创建目录 `mkdir /bigdata` 在 cdh0执行

上传:

- `apache-maven-3.0.5-bin.tar.gz`
- `hadoop-2.6.0-cdh5.14.4.tar.gz`
- `hive-1.1.0-cdh5.14.4.tar.gz`

并解压到/bigdata 目录下, 参考命令 `tar -zxvf hive-1.1.0-cdh5.14.4.tar.gz -C /bigdata/`

使用 `scp` 命令将 /bigdata 目录复制到cdh1 和 cdh2

`scp -r /bigdata cdh1:/`

`scp -r /bigdata cdh2:/`

> 目录/bigdata 用来安装hadoop等组件，同学可以自行修改

i. 安装jdk

上传jdk1.7 和 jdk1.8到服务器并解压

设置java home 为jdk1.8

在每个hadoop节点（cdh0,1,2)机器都执行

j. 配置环境变量

```
export HADOOP_HOME=/bigdata/hadoop-2.6.0-cdh5.14.4
export MAVEN_HOME=/bigdata/apache-maven-3.0.5
export HIVE_HOME=/bigdata/hive-1.1.0-cdh5.14.4
export JAVA_HOME=/usr/local/jdk1.8.0_221
export
PATH=$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$MAVEN_HOME/bin:$HIVE_HOME/bin:
$PATH
# export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
# export HADOOP_OPTS="-Djava.library.path=${HADOOP_HOME}/lib/native"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/bigdata/hadoop-2.6.0-cdh5.14.4/lib/native
```

`source /etc/profile`

在每个hadoop节点（cdh0,1,2)机器都执行

三： Kerberos搭建

> 按照环境准备中的设定，cdh0 cdh1 cdh2 均作为Kerberos 的客户端， cdh3作为Kerberos的服务端

搭建Kerberos Server

> 以下操作运行在cdh3

1. 使用 `yum` 安装Kerberos Server的套件

   `yum install -y krb5-libs krb5-server krb5-workstation`

2. 配置 `/etc/krb5.conf`

   `vim /etc/krb5.conf`

   填入以下内容:

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = ITCAST.CN
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true

[realms]
 ITCAST.CN = {
  kdc = cdh3.itcast.cn
  admin_server = cdh3.itcast.cn
 }

[domain_realm]
```

```
  .itcast.cn = ITCAST.CN
  itcast.cn = ITCAST.CN
```

名词讲解：

- realm 域：表示一个公司或者一个组织。逻辑上的授权认证范围

  比如，某个认证账户是属于某个域下的，跨域账户不通用

  域和FQDN的配置很像，使用大写，本次演示使用ITCAST.CN 来标记域

其中：

- logging 块配置日志相关
- libdefaults块配置默认的设置，包括ticket的生存周期等
- realms 是域的配置，可以配置多个realm，本次演示只配置一个 即是 ITCAST.CN
- domain_realm 是 Kerberos内的域 和 主机名的域的一个对应关系
  - `.itcast.cn` 类似 `*.itcast.cn` 表示如cdh0.itcast.cn cdh1.itcast.cn 等均是 ITCAST.CN 这个realm
  - `itcast.cn` 表示 `itcast.cn` 这个主机名也是 `ITCAST.CN` 这个realm的一部分

3. 配置 `/var/kerberos/krb5kdc/kdc.conf`

填入：

```
  kdc_ports = 88
  kdc_tcp_ports = 88

 [realms]
  ITCAST.CN = {
   #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
   supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal
arcfour-hmac:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal
  }
```

名词解释：

`acl_file` : `Kerberos acl` 的一些配置对应的文件

`kerberos` : admin账户的 `keytab` 本地路径

`keytab` : 登录凭证，有了这个相当于直接有了 `ticket` ， 可以免密直接登录某个账户，所以这个文件很重要

> 在一些企业的配置中，对于用户的配置，就是直接发放给用户对应的 `keytab` 文件，用户有了这个文件就能访问对应的资源
>
> 就类似于 `ssh` 中的 登录私钥一样 有了私钥就能直接登录机器
>
> 这个文件就是一把钥匙，能开门的

这个配置文件内：设置了对于server的一些重要配置

比如：

- `kdc` 的端口
- 以及ITCAST.CN这个域中的 `acl file` 文件路径(下一步设置它)

- admin账户的 `keytab`
- 支持的加密方法等

4. 配置 `/var/kerberos/krb5kdc/kadm5.acl`

   填入 `*/admin@ITCAST.CN  *`

   其中 */admin 是Kerberos中的账户形式

   如账户 `rm/cdh0.itcast.cn@ITCAST.CN` 表示 在 cdh0机器上的 resourcemanager账户

   这个账户属于ITCAST.CN这个域

   最后的 * 表示 符合 */admin的账户拥有所有权限

   > 这一步也就是配置了admin的规则 如 admin/ **admin@ITCAST.CN** 就拥有ITCAST.CN 域内的全部权限

5. 初始化Kerberos的数据库

   输入: `kdb5_util create -s -r ITCAST.CN`

   其中 `ITCAST.CN` 是对应的域，如果你的不同请修改

   然后命令要求设置数据库master的密码，要求输入两次，输入 `krb5kdc` 即可

   这样得到 数据库master账户： `K/M@ITCAST.CN` , 密码： `krb5kdc`

6. 创建ITCAST.CN 域内的管理员

   执行：kadmin.local 进入 kerberos 的 admin 命令行界面

   ```
   # 输入如下内容，添加一个用户
   addprinc root/admin@ITCAST.CN
   # 要求输入密码，输入root作为密码（可自行设置）

   # 上面的账户就作为ITCAST.CN的管理员账户存在 （满足 */admin@ITCAST.CN 的规则 拥有全部权限）
   # 再创建一个 测试的管理员用户
   addprinc krbtest/admin@ITCAST.CN # 同样满足 */admin@ITCAST.CN 密码设置为krbtest

   # 查看当前拥有的所有用户
   listprincs
   ```

   > 名词解释: principal
   >
   > 可以当作是用户的意思 一个principal由3个部分组成，如下:
   >
   > `nn/cdh0.itcast.cn@ITCAST.CN`
   >
   > 也就是 `account/instance@realm`
   >
   > 其中account 表示账户名 或者服务类型
   >
   > instance表示实例，一般为主机名表示 属于这个主机名下的某个账户
   >
   > realm 就是域 如 ITCAST.CN

   > `nn/cdh0.itcast.cn@ITCAST.CN` 就表示
   >
   > nn 这个账户 只能在cdh0机器登录 ITCAST.CN 这个域

7. 重启Kerberos server的组件并设置开机自启

   `service krb5kdc restart`

   `service kadmin restart`

`chkconfig krb5kdc on`

`chkconfig kadmin on`

搭建Kerberos 客户端

1. 在cdh0 cdh1 chd2 均执行：

   `yum -y install krb5-libs krb5-workstation`

   使用 `yum` 来安装客户端组件
2. 从server机器将 `/etc/krb5.conf` 复制到各个客户端同样的位置
3. 测试登录admin

   执行 `kinit krbtest/admin@ITCAST.CN` 输入密码

   正确的结果就是没有任何反应

   然后输入 `klist`

   正确应该可以得到如下输出:

   ```
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: krbtest/admin@ITCAST.CN

   Valid starting     Expires            Service principal
   09/25/19 11:41:52  09/26/19 11:41:52  krbtgt/ITCAST.CN@ITCAST.CN
           renew until 09/25/19 11:41:52
   ```
4. 此时，客户端配置完成，使用kinit 即可得到对应账户的ticket

规划为Hadoop中各个服务分配Kerberos的principal

先在每个节点执行： `mkdir /etc/security/keytabs`

我们预计将要使用Kerberos的服务有：

`namenode` 、 `secondarynamenode` 、 `datanode` 、 `resourcemanager` 、 `nodemanager` 、 `job historyserver` 、 `https` 、 `hive`

那么应给这些服务根据其所运行的主机分配对应的账户

如 `namenode` 会运行在cdh0上，所以可以分配账户 `nn/cdh0.itcast.cn@ITCAST.CN`

其中nn表示namenode， cdh0.itcast.cn是principal中的instance 我们以主机名来代替，最后是域 ITCAST.CN

那么其他服务可以有 `sn: 对应secondarynamenode`

`dn : datanode`

`rm: resourcemanager`

`nm: nodemanager`

`jhs: job history server`

`HTTP: https相关服务`

`hive: hive服务`

到这里我们先了解上述的规则，然后先不创建具体用户，在用到的地方在创建，避免提前创建的和预期不符合

四：配置HDFS

> 至此，Kerberos已经安装完成，现在需要配置Hadoop集群，针对Kerberos进行设置

1. 添加用户

```
groupadd hadoop;useradd hdfs -g hadoop -p hdfs;useradd hive -g hadoop -p hive;useradd yarn -g hadoop -p yarn;useradd mapred -g hadoop -p mapred
```

cdh0 cdh1 cdh2 均执行

2. 配置HDFS相关的Kerberos账户

> 说明： Hadoop需要Kerberos来进行认证，以启动服务来说，在后面配置 `hadoop` 的时候我们会给对应服务指定一个Kerberos的账户，比如 `namenode` 运行在cdh0机器上，我们可能将 `namenode` 指定给了 `nn/cdh0.itcast.cn@ITCAST.CN` 这个账户， 那么 想要启动 `namenode` 就必须认证这个账户才可以。

1. 在每个节点执行 `mkdir /etc/security/keytabs`
2. 配置cdh0上面运行的服务对应的Kerberos账户

```
执行 kadmin
输入密码，进入Kerberos的admin后台

创建namenode的账户
addprinc -randkey nn/cdh0.itcast.cn@ITCAST.CN
创建secondarynamenode的账户
addprinc -randkey sn/cdh0.itcast.cn@ITCAST.CN
创建用于https服务的相关账户
addprinc -randkey HTTP/cdh0.itcast.cn@ITCAST.CN

# 防止启动或者操作的过程中需要输入密码，创建免密登录的keytab文件
# 创建nn账户的keytab
ktadd -k /etc/security/keytabs/nn.service.keytab nn/cdh0.itcast.cn@ITCAST.CN
```

```
 # 创建sn账户的keytab
 ktadd -k /etc/security/keytabs/sn.service.keytab sn/cdh0.itcast.cn@ITCAST.CN
 # 创建HTTP账户的keytab
 ktadd -k /etc/security/keytabs/spnego.service.keytab
HTTP/cdh0.itcast.cn@ITCAST.CN


 最终得到:
 -r-------- 1 hdfs hadoop 406 Sep 26 11:11 nn.service.keytab
 -r-------- 1 hdfs hadoop 406 Sep 26 11:13 sn.service.keytab
 -r-------- 1 hdfs hadoop 418 Sep 26 12:20 spnego.service.keytab
```

3. 配置cdh1 和 cdh2 上面运行的服务对应的Kerberos账户

   分别在cdh1 和 cdh2上执行以下操作

```
 kadmin 进入admin后台

 # 创建datanode的账户
 addprinc -randkey dn/cdh1.itcast.cn@ITCAST.CN   # 如果是cdh2机器 使用
cdh2.itcast.cn
 # 创建datanode需要使用的https相关服务账户
 addprinc -randkey HTTP/cdh1.itcast.cn@ITCAST.CN # 如果是cdh2机器 使用
cdh2.itcast.cn

 # 添加keytab
 ktadd -k /etc/security/keytabs/dn.service.keytab dn/cdh1(或者
2).itcast.cn@ITCAST.CN
 ktadd -k /etc/security/keytabs/spnego.service.keytab HTTP/cdh1(或者
2).itcast.cn@ITCAST.CN

 最终得到:
 -r-------- 1 hdfs hadoop 406 Sep 26 11:18 dn.service.keytab
 -r-------- 1 hdfs hadoop 418 Sep 26 12:22 spnego.service.keytab
```

4. 在cdh0 cdh1 cdh2 上将刚刚得到的 keytab文件全部设置:

   `chown hdfs:hadoop`

   以及

   `chmod 400`

3. 创建数据目录

```
mkdir -p /data/nn;mkdir /data/dn; mkdir /data/nm-local;mkdir /data/nm-log;mkdir
/data/mr-history
```

/data/nn: namenode数据目录

/data/dn: datanode 数据目录

/data/nm-local;/data/nm-log : nodemanager相关目录

/data/mr-history: mr history目录

```
mkdir -p /bigdata/hadoop-2.6.0-cdh5.14.4/logs/hadoop
```

```
mkdir /bigdata/hadoop-2.6.0-cdh5.14.4/logs/yarn
```

在cdh0 cdh1 cdh2 执行

> 上述目录可以根据需要修改

4. 安装 protobuf

上传安装包内的 `protobuf-2.5.0.tar.gz` ，解压进入目录执行：

```
./configure
```

```
make
```

```
make install
```

在cdh0执行

编译源码构建Linux-Container-executor

> Kerberos需要使用基于cgroup工作的一个名为Linux-container-executer的容器来运行YARN任务，这个容器需要我们自己编译源码来构建出来
>
> 如果编译不出来可以使用课件内提供的（基于centos6.9 cdh5.14.4版本构建）但是不100%确保能够正确运行
>
> 最好自己构建出来这样100%能运行

- `cd $HADOOP_HOME/src/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-nodemanager`
- 执行 `mvn package -Pdist,native -DskipTests -Dtar -Dcontainer-executor.conf.dir=/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop/`

> 正常情况下 执行这一句命令，经过一段时间的联网下载依赖之后会报错，说语法错误
>
> 这是因为源码使用jdk1.7开发，我们也只能用1.7来编译
>
> 但是因为1.7的在SSL上无法连接cloudera的库，所以先使用1.8下载好依赖
>
> 然后接着用1.7来编译

- 使用jdk1.7再次编译

  `修改 mvn程序 增加 export JAVA_HOME=jdk1.7的路径` 后 再次执行编译命令
- 编译成功后，进入 `target/native/target/usr/local/bin` 将里面两个文件copy到 `$HADOOP_HOME/bin/` 下

在cdh0一台机器编译即可，编译完成后复制到其他机器

- 将这两个文件远程复制到cdh1 cdh2同样的目录下

5. 执行如下脚本，设置 hadoop 需要使用的各个目录的权限

```
# 将如下内容保存到.sh内，然后执行 sh xxx.sh 以root执行

# 其中内部的配置根据自己的目录设置修改
HADOOP_HOME=/bigdata/hadoop-2.6.0-cdh5.14.4
DFS_NAMENODE_NAME_DIR=/data/nn
DFS_DATANODE_DATA_DIR=/data/dn
NODEMANAGER_LOCAL_DIR=/data/nm-local
NODEMANAGER_LOG_DIR=/data/nm-log
MR_HISTORY=/data/mr-history


if [ ! -n "$HADOOP_HOME" ];then
        echo "请填入hadoop home 路径"
        exit
fi


chgrp -R hadoop $HADOOP_HOME
chown -R hdfs:hadoop $HADOOP_HOME
chown root:hadoop $HADOOP_HOME
chown hdfs:hadoop $HADOOP_HOME/sbin/distribute-exclude.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/hadoop-daemon.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/hadoop-daemons.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/hdfs-config.cmd
chown hdfs:hadoop $HADOOP_HOME/sbin/hdfs-config.sh
chown mapred:hadoop $HADOOP_HOME/sbin/mr-jobhistory-daemon.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/refresh-namenodes.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/slaves.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/start-all.cmd
chown hdfs:hadoop $HADOOP_HOME/sbin/start-all.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/start-balancer.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/start-dfs.cmd
chown hdfs:hadoop $HADOOP_HOME/sbin/start-dfs.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/start-secure-dns.sh
chown yarn:hadoop $HADOOP_HOME/sbin/start-yarn.cmd
chown yarn:hadoop $HADOOP_HOME/sbin/start-yarn.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/stop-all.cmd
chown hdfs:hadoop $HADOOP_HOME/sbin/stop-all.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/stop-balancer.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/stop-dfs.cmd
chown hdfs:hadoop $HADOOP_HOME/sbin/stop-dfs.sh
chown hdfs:hadoop $HADOOP_HOME/sbin/stop-secure-dns.sh
chown yarn:hadoop $HADOOP_HOME/sbin/stop-yarn.cmd
chown yarn:hadoop $HADOOP_HOME/sbin/stop-yarn.sh
chown yarn:hadoop $HADOOP_HOME/sbin/yarn-daemon.sh
chown yarn:hadoop $HADOOP_HOME/sbin/yarn-daemons.sh
chown mapred:hadoop $HADOOP_HOME/bin/mapred*
chown yarn:hadoop $HADOOP_HOME/bin/yarn*
chown hdfs:hadoop $HADOOP_HOME/bin/hdfs*

chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/capacity-scheduler.xml
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/configuration.xsl
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/core-site.xml
```

```
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/hadoop-*
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/hdfs-*
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/httpfs-*
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/kms-*
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/log4j.properties
chown mapred:hadoop $HADOOP_HOME/etc/hadoop/mapred-*
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/slaves
chown hdfs:hadoop $HADOOP_HOME/etc/hadoop/ssl-*
chown yarn:hadoop $HADOOP_HOME/etc/hadoop/yarn-*
chmod 755 -R $HADOOP_HOME/etc/hadoop/*
chown root:hadoop $HADOOP_HOME/etc
chown root:hadoop $HADOOP_HOME/etc/hadoop
chown root:hadoop $HADOOP_HOME/etc/hadoop/container-executor.cfg
chown root:hadoop $HADOOP_HOME/bin/container-executor
chown root:hadoop $HADOOP_HOME/bin/test-container-executor
chmod 6050 $HADOOP_HOME/bin/container-executor
chown 6050 $HADOOP_HOME/bin/test-container-executor

mkdir $HADOOP_HOME/logs
mkdir $HADOOP_HOME/logs/hdfs
mkdir $HADOOP_HOME/logs/yarn
chown root:hadoop $HADOOP_HOME/logs
chmod 775 $HADOOP_HOME/logs
chown hdfs:hadoop $HADOOP_HOME/logs/hdfs
chmod 755 -R $HADOOP_HOME/logs/hdfs
chown yarn:hadoop $HADOOP_HOME/logs/yarn
chmod 755 -R $HADOOP_HOME/logs/yarn

chown -R hdfs:hadoop $DFS_DATANODE_DATA_DIR
chown -R hdfs:hadoop $DFS_NAMENODE_NAME_DIR
chmod 700 $DFS_DATANODE_DATA_DIR
chmod 700 $DFS_NAMENODE_NAME_DIR

chown -R yarn:hadoop $NODEMANAGER_LOCAL_DIR
chown -R yarn:hadoop $NODEMANAGER_LOG_DIR
chmod 770 $NODEMANAGER_LOCAL_DIR
chmod 770 $NODEMANAGER_LOG_DIR

chown -R mapred:hadoop $MR_HISTORY
chmod 770 $MR_HISTORY
```

6. 配置hadoop的 lib/native(本地运行库)

> Hadoop是使用Java语言开发的，但是有一些需求和操作并不适合使用java，所以就引入了本地库（Native Libraries）的概念，通过本地库，Hadoop可以更加高效地执行某一些操作。

- 上传提供的 hadoop-2.6.0+cdh5.14.4+2785-1.cdh5.14.4.p0.4.el6.x86_64.rpm
- 执行 `rpm2cpio hadoop-2.6.0+cdh5.14.4+2785-1.cdh5.14.4.p0.4.el6.x86_64.rpm | cpio -div` 解压
- 进入解压后的路径 `usr/lib/hadoop/lib/native`
- 复制里面的内容到 `$HADOOP_HOME/lib/native` 下，因为被复制的有一些软连接，所以使用 `cp -d` 以及 `scp -d` 命令来复制
- 将 `$HADOOP_HOME/lib/native` 内的内容使用 `scp -d` 命令复制到其它机器上（cdh1 和 cdh2）

> 对于其它版本的CDH，请自行根据版本去CDH库中下载对应版本的hadoop rpm 包解压得到本地库文件

7. 设置 HDFS 的配置文件

1. `hadoop-env.sh` 增加

```
export JAVA_HOME=/usr/local/jdk1.8.0_221
export HADOOP_HOME=/bigdata/hadoop-2.6.0-cdh5.14.4
export HADOOP_CONF_DIR=/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop
export HADOOP_LOG_DIR=/bigdata/hadoop-2.6.0-cdh5.14.4/logs/hdfs
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=${HADOOP_HOME}/lib/native"

# 路径请根据具体自己配置的路径修改
```

2. `yarn-env.sh` 增加

```
export JAVA_HOME=/usr/local/jdk1.8.0_221
export YARN_CONF_DIR=/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop
export YARN_LOG_DIR=/bigdata/hadoop-2.6.0-cdh5.14.4/logs/yarn
```

3. `mapred-env.sh` 增加

```
export JAVA_HOME=/usr/local/jdk1.8.0_221
```

4. `core-site.xml` 配置如下

> 配置项见注释

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://cdh0.itcast.cn:8020</value>
    <description></description>
  </property>

  <property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
    <description></description>
  </property>

<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
  <description>是否开启hadoop的安全认证</description>
</property>

<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
  <description>使用kerberos作为hadoop的安全认证方案</description>
</property>
```

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
          RULE:[2:$1@$0](nn@.*ITCAST.CN)s/.*/hdfs/
          RULE:[2:$1@$0](sn@.*ITCAST.CN)s/.*/hdfs/
          RULE:[2:$1@$0](dn@.*ITCAST.CN)s/.*/hdfs/
          RULE:[2:$1@$0](nm@.*ITCAST.CN)s/.*/yarn/
          RULE:[2:$1@$0](rm@.*ITCAST.CN)s/.*/yarn/
          RULE:[2:$1@$0](tl@.*ITCAST.CN)s/.*/yarn/
          RULE:[2:$1@$0](jhs@.*ITCAST.CN)s/.*/mapred/
          RULE:[2:$1@$0](HTTP@.*ITCAST.CN)s/.*/hdfs/
          DEFAULT
  </value>
  <description>匹配规则，比如第一行就是表示将nn/*@ITCAST.CN的principal 绑定到hdfs账户上，也就是想要得到一个认证后的hdfs账户，请使用Kerberos的nn/*@ITCAST.CN账户来认证
     同理，下面的HTTP开头的Kerberos账户，其实也是绑定到了hdfs本地账户上，也就是如果想要操作hdfs，用nn和http都是可以的，只是我们在逻辑上多创建几个kerberos账户好分配，比如namenode分配nn， datanode分配给dn  其实 nn 和 dn 都是对应的hdfs</description>
</property>

<!-- HIVE KERBEROS -->
<property>
        <name>hadoop.proxyuser.hive.hosts</name>
        <value>*</value>
</property>

<property>
        <name>hadoop.proxyuser.hive.groups</name>
        <value>*</value>
</property>

<property>
        <name>hadoop.proxyuser.hdfs.hosts</name>
        <value>*</value>
</property>

<property>
        <name>hadoop.proxyuser.hdfs.groups</name>
        <value>*</value>
</property>

<property>
        <name>hadoop.proxyuser.HTTP.hosts</name>
        <value>*</value>
</property>

<property>
        <name>hadoop.proxyuser.HTTP.groups</name>
        <value>*</value>
</property>

<property>
        <name></name>
        <value></value>
</property>

</configuration>
```

5. 配置 `hdfs-site.xml`

```
 <configuration>
   <property>
     <name>dfs.namenode.name.dir</name>
     <value>/data/nn</value>
     <description>Path on the local filesystem where the NameNode stores the
namespace and transactions logs persistently.</description>
   </property>

   <property>
     <name>dfs.namenode.hosts</name>
     <value>cdh1.itcast.cn,cdh2.itcast.cn</value>
     <description>List of permitted DataNodes.</description>
   </property>

   <property>
     <name>dfs.blocksize</name>
     <value>268435456</value>
     <description></description>
   </property>


   <property>
     <name>dfs.namenode.handler.count</name>
     <value>100</value>
     <description></description>
   </property>

   <property>
     <name>dfs.datanode.data.dir</name>
     <value>/data/dn</value>
   </property>

   <property>
     <name>dfs.block.access.token.enable</name>
     <value>true</value>
   </property>

     <!-- NameNode security config -->
     <property>
         <name>dfs.namenode.kerberos.principal</name>
         <value>nn/_HOST@ITCAST.CN</value>
         <description>namenode对应的kerberos账户为 nn/主机名@ITCAST.CN    _HOST会自动
转换为主机名</description>
     </property>
     <property>
         <name>dfs.namenode.keytab.file</name>
         <!-- path to the HDFS keytab -->
         <value>/etc/security/keytabs/nn.service.keytab</value>
         <description>因为使用-randkey 创建的用户 密码随机不知道，所以需要用免密登录的
keytab文件 指定namenode需要用的keytab文件在哪里</description>
     </property>

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

```
        <value>HTTP/_HOST@ITCAST.CN</value>
        <description>https 相关（如开启namenodeUI）使用的账户</description>
    </property>
    <!--Secondary NameNode security config -->
    <property>
        <name>dfs.secondary.namenode.kerberos.principal</name>
        <value>sn/_HOST@ITCAST.CN</value>
        <description>secondarynamenode使用的账户</description>
    </property>
    <property>
        <name>dfs.secondary.namenode.keytab.file</name>
        <!-- path to the HDFS keytab -->
        <value>/etc/security/keytabs/sn.service.keytab</value>
        <description>sn对应的keytab文件</description>
    </property>
    <property>
        <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
        <value>HTTP/_HOST@ITCAST.CN</value>
        <description>sn需要开启http页面用到的账户</description>
    </property>
    <!-- DataNode security config -->
    <property>
        <name>dfs.datanode.kerberos.principal</name>
        <value>dn/_HOST@ITCAST.CN</value>
        <description>datanode用到的账户</description>
    </property>
    <property>
        <name>dfs.datanode.keytab.file</name>
        <!-- path to the HDFS keytab -->
        <value>/etc/security/keytabs/dn.service.keytab</value>
        <description>datanode用到的keytab文件路径</description>
    </property>

    <property>
        <name>dfs.datanode.data.dir.perm</name>
        <value>700</value>
    </property>

    <property>
        <name>dfs.web.authentication.kerberos.principal</name>
        <value>HTTP/_HOST@ITCAST.CN</value>
        <description>web hdfs 使用的账户</description>
    </property>
    <property>
        <name>dfs.web.authentication.kerberos.keytab</name>
        <value>/etc/security/keytabs/spnego.service.keytab</value>
        <description>对应的keytab文件</description>
    </property>
    <property>
        <name>dfs.permissions.supergroup</name>
        <value>hdfs</value>
    </property>

    <property>
        <name>dfs.http.policy</name>
        <value>HTTPS_ONLY</value>
        <description>所有开启的web页面均使用https, 细节在ssl server 和client那个配置文
件内配置</description>
```

```
    </property>

    <property>
        <name>dfs.data.transfer.protection</name>
        <value>integrity</value>
    </property>
    <property>
        <name>dfs.https.port</name>
        <value>50470</value>
    </property>
</configuration>
```

8. 创建HTTPS证书

1. 在cdh0上

```
# 1
mkdir /etc/security/cdh.https
cd /etc/security/cdh.https

# 2
openssl req -new -x509 -keyout bd_ca_key -out bd_ca_cert -days 9999 -subj
'/C=CN/ST=beijing/L=beijing/O=test/OU=test/CN=test'（输入密码和确认密码是123456，此命
令成功后输出
bd_ca_key和bd_ca_cert两个文件）
>>>
-rw-r--r-- 1 root root 1294 Sep 26 11:31 bd_ca_cert
-rw-r--r-- 1 root root 1834 Sep 26 11:31 bd_ca_key

# 3
将得到的两个文件复制到cdh1 cdh2上
scp -r /etc/security/cdh.https cdh1:/etc/security/
scp -r /etc/security/cdh.https cdh2:/etc/security/
```

2. 在cdh0 cdh1 cdh2 每个节点上都依次执行以下命令

```
cd /etc/security/cdh.https

# 所有需要输入密码的地方全部输入123456（方便起见，如果你对密码有要求请自行修改）

# 1  输入密码和确认密码：123456，此命令成功后输出keystore文件
keytool -keystore keystore -alias localhost -validity 9999 -genkey -keyalg RSA -
keysize 2048 -dname "CN=test, OU=test, O=test, L=beijing, ST=beijing, C=CN"

# 2 输入密码和确认密码：123456，提示是否信任证书：输入yes，此命令成功后输出truststore文件
keytool -keystore truststore -alias CARoot -import -file bd_ca_cert

# 3 输入密码和确认密码：123456，此命令成功后输出cert文件
keytool -certreq -alias localhost -keystore keystore -file cert

# 4 此命令成功后输出cert_signed文件
openssl x509 -req -CA bd_ca_cert -CAkey bd_ca_key -in cert -out cert_signed -
days 9999 -CAcreateserial -passin pass:123456

# 5 输入密码和确认密码：123456，是否信任证书，输入yes，此命令成功后更新keystore文件
keytool -keystore keystore -alias CARoot -import -file bd_ca_cert
```

```
# 6 输入密码和确认密码：123456
keytool -keystore keystore -alias localhost -import -file cert_signed


最终得到：
-rw-r--r-- 1 root root 1294 Sep 26 11:31 bd_ca_cert
-rw-r--r-- 1 root root   17 Sep 26 11:36 bd_ca_cert.srl
-rw-r--r-- 1 root root 1834 Sep 26 11:31 bd_ca_key
-rw-r--r-- 1 root root 1081 Sep 26 11:36 cert
-rw-r--r-- 1 root root 1176 Sep 26 11:36 cert_signed
-rw-r--r-- 1 root root 4055 Sep 26 11:37 keystore
-rw-r--r-- 1 root root  978 Sep 26 11:35 truststore
```

9. 配置ssl-server.xml和ssl-client.xml

> 刚刚配置到的https证书以及设置的密码等等在这里就用上场了

1. `ssl-sserver.xml` 如下：

```
<configuration>
<property>
  <name>ssl.server.truststore.location</name>
  <value>/etc/security/cdh.https/truststore</value>
  <description>Truststore to be used by NN and DN. Must be specified.
  </description>
</property>

<property>
  <name>ssl.server.truststore.password</name>
  <value>123456</value>
  <description>Optional. Default value is "".
  </description>
</property>

<property>
  <name>ssl.server.truststore.type</name>
  <value>jks</value>
  <description>Optional. The keystore file format, default value is "jks".
  </description>
</property>

<property>
  <name>ssl.server.truststore.reload.interval</name>
  <value>10000</value>
  <description>Truststore reload check interval, in milliseconds.
  Default value is 10000 (10 seconds).
  </description>
</property>

<property>
  <name>ssl.server.keystore.location</name>
  <value>/etc/security/cdh.https/keystore</value>
  <description>Keystore to be used by NN and DN. Must be specified.
  </description>
```

```
</property>

<property>
  <name>ssl.server.keystore.password</name>
  <value>123456</value>
  <description>Must be specified.
  </description>
</property>

<property>
  <name>ssl.server.keystore.keypassword</name>
  <value>123456</value>
  <description>Must be specified.
  </description>
</property>

<property>
  <name>ssl.server.keystore.type</name>
  <value>jks</value>
  <description>Optional. The keystore file format, default value is "jks".
  </description>
</property>

<property>
  <name>ssl.server.exclude.cipher.list</name>
  <value>TLS_ECDHE_RSA_WITH_RC4_128_SHA,SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA,
  SSL_RSA_WITH_DES_CBC_SHA,SSL_DHE_RSA_WITH_DES_CBC_SHA,
  SSL_RSA_EXPORT_WITH_RC4_40_MD5,SSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
  SSL_RSA_WITH_RC4_128_MD5</value>
  <description>Optional. The weak security cipher suites that you want excluded
  from SSL communication.</description>
</property>

</configuration>
```

2. `ssl-client.xml` 如下：

```
<configuration>
<property>
  <name>ssl.client.truststore.location</name>
  <value>/etc/security/cdh.https/truststore</value>
  <description>Truststore to be used by clients like distcp. Must be
  specified.
  </description>
</property>

<property>
  <name>ssl.client.truststore.password</name>
  <value>123456</value>
  <description>Optional. Default value is "".
  </description>
</property>

<property>
  <name>ssl.client.truststore.type</name>
  <value>jks</value>
  <description>Optional. The keystore file format, default value is "jks".
  </description>
```

```
  </property>

  <property>
    <name>ssl.client.truststore.reload.interval</name>
    <value>10000</value>
    <description>Truststore reload check interval, in milliseconds.
    Default value is 10000 (10 seconds).
    </description>
  </property>

  <property>
    <name>ssl.client.keystore.location</name>
    <value>/etc/security/cdh.https/keystore</value>
    <description>Keystore to be used by clients like distcp. Must be
    specified.
    </description>
  </property>

  <property>
    <name>ssl.client.keystore.password</name>
    <value>123456</value>
    <description>Optional. Default value is "".
    </description>
  </property>

  <property>
    <name>ssl.client.keystore.keypassword</name>
    <value>123456</value>
    <description>Optional. Default value is "".
    </description>
  </property>

  <property>
    <name>ssl.client.keystore.type</name>
    <value>jks</value>
    <description>Optional. The keystore file format, default value is "jks".
    </description>
  </property>

  </configuration>
```

10. 配置 `slaves`

```
cdh1.itcast.cn
cdh2.itcast.cn
```

11. 将HDFS相关配置文件复制到其他节点

```
scp hadoop-env.sh yarn-env.sh mapred-env.sh core-site.xml hdfs-site.xml ssl-client.xml
ssl-server.xml slaves cdh1:/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop/
scp hadoop-env.sh yarn-env.sh mapred-env.sh core-site.xml hdfs-site.xml ssl-client.xml
ssl-server.xml slaves cdh2:/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop/
```

12. 启动HDFS测试

12.1 启动Namenode

认证 `namenode` 对应的Kerberos账户

1. 向Kerberos认证 `namenode` 对应的账户: `nn/cdh0.itcast.cn@ITCAST.CN`

   得到认证才可以对 `namenode` 进行操作

   切换到 `hdfs` 账户: `su - hdfs`

   输入: `kinit -kt /etc/security/keytabs/nn.service.keytab nn/cdh0.itcast.cn`

   即可完成认证，输入 `klist` 查看认证结果，正常如下:

   ```
   [hdfs@cdh0 ~]$ klist
   Ticket cache: FILE:/tmp/krb5cc_505
   Default principal: nn/cdh0.itcast.cn@ITCAST.CN

   Valid starting     Expires            Service principal
   09/26/19 11:59:08  09/27/19 11:59:08  krbtgt/ITCAST.CN@ITCAST.CN
           renew until 09/26/19 11:59:08
   ```

2. 格式化namenode

   执行 `hadoop namenode -format`

启动 `namenode` 和 `secondarynamenode`

`hadoop-daemon.sh start namenode`

`hadoop-daemon.sh start secondarynamenode`

查看 `namenode` `webui`

浏览器打开: **https://cdh0:50470**

12.2 启动 `Datanode`

> 在cdh1 cdh2 上分别执行下面的流程

认证 `datanode` 对应的Kerberos账户

切换到 `hdfs` 账户

执行： `kinit -kt /etc/security/keytabs/dn.service.keytab dn/cdh1.itcast.cn@ITCAST.CN`

上面的cdh1 在cdh2执行的时候换成cdh2

执行 `klist` 查看是否认证成功

启动 datanode

执行 `hadoop-daemon.sh start datanode`

两个datanode都启动完成后，浏览器打开: https://cdh0:50470 查看datanode是否加入了集群

13. 上传文件测试 hdfs

```
在cdh0机器上

su - hdfs
kinit -kt /etc/security/keytabs/nn.service.keytab nn/cdh0.itcast.cn
cd /bigdata/hadoop-2.6.0-cdh5.14.4
hadoop fs -put ./README.txt /readme.txt

[hdfs@cdh0 hadoop-2.6.0-cdh5.14.4]$ hadoop fs -ls /
Found 1 items
-rw-r--r--   3 hdfs hdfs       1366 2019-09-26 13:57 /readme.txt


hadoop fs -cat /readme.txt
```

14. 创建相应的 hdfs 目录，并设置权限

> 其他服务会用到hdfs目录，预先创建好并设置好权限，避免其他服务没有权限运行
>
> 继续在cdh0机器的hdfs账户下执行以下命令

`hadoop fs -chown hdfs:hadoop /`

`hadoop fs -mkdir /tmp`

`hadoop fs -chown hdfs:hadoop /tmp`

`hadoop fs -chmod 777 /tmp`

`hadoop fs -mkdir /user`

`hadoop fs -chown hdfs:hadoop /user`

```
hadoop fs -chmod 775 /user
hadoop fs -mkdir /mr-data
hadoop fs -chown mapred:hadoop /mr-data
hadoop fs -mkdir /tmp/hadoop-yarn
hadoop fs -chmod 770 /tmp/hadoop-yarn
```

五： 配置YARN

1. 配置YARN相关的Kerberos账户

a. 配置cdh0

```
# 在root用户下执行
kadmin # 进入kerberos admin后台

# 添加resourcemanager的账户
addprinc -randkey rm/cdh0.itcast.cn@ITCAST.CN
# 添加job historyserver的账户
addprinc -randkey jhs/cdh0.itcast.cn@ITCAST.CN

# 添加rm账户的本地keytab
ktadd -k /etc/security/keytabs/rm.service.keytab rm/cdh0.itcast.cn@ITCAST.CN
ktadd -k /etc/security/keytabs/jhs.service.keytab jhs/cdh0.itcast.cn@ITCAST.CN
# 将得到的rm.service.keytab 改权限
chmod 400 rm.service.keytab
chown yarn:hadoop rm.service.keytab

# 将得到的jhs.service.keytab 改权限
chmod 400 jhs.service.keytab
chown mapred:hadoop jhs.service.keytab

# 得到
[root@cdh0 keytabs]# ll
total 20
-r-------- 1 mapred hadoop 412 Sep 26 14:20 jhs.service.keytab
-r-------- 1 hdfs   hadoop 406 Sep 26 11:11 nn.service.keytab
-r-------- 1 yarn   hadoop 406 Sep 26 14:09 rm.service.keytab
-r-------- 1 hdfs   hadoop 406 Sep 26 11:13 sn.service.keytab
-r-------- 1 hdfs   hadoop 418 Sep 26 12:20 spnego.service.keytab
```

b. 配置cdh1

```
# 在root用户下执行
kadmin # 进入kerberos admin后台

# 添加nodemanager的账户
addprinc -randkey nm/cdh1.itcast.cn@ITCAST.CN

# 添加nm账户的本地keytab
```

```
ktadd -k /etc/security/keytabs/nm.service.keytab nm/cdh1.itcast.cn@ITCAST.CN

# 将得到的nm.service.keytab 改权限
chmod 400 nm.service.keytab
chown yarn:hadoop nm.service.keytab

# 得到
[root@cdh1 keytabs]# ll
total 12
-r-------- 1 hdfs hadoop 406 Sep 26 11:18 dn.service.keytab
-r-------- 1 yarn hadoop 406 Sep 26 14:14 nm.service.keytab
-r-------- 1 hdfs hadoop 418 Sep 26 12:22 spnego.service.keytab
```

c. 配置cdh2

```
# 在root用户下执行
kadmin # 进入kerberos admin后台

# 添加nodemanager的账户
addprinc -randkey nm/cdh2.itcast.cn@ITCAST.CN

# 添加nm账户的本地keytab
ktadd -k /etc/security/keytabs/nm.service.keytab nm/cdh2.itcast.cn@ITCAST.CN

# 将得到的nm.service.keytab 改权限
chmod 400 nm.service.keytab
chown yarn:hadoop nm.service.keytab

# 得到
[root@cdh1 keytabs]# ll
total 12
-r-------- 1 hdfs hadoop 406 Sep 26 11:18 dn.service.keytab
-r-------- 1 yarn hadoop 406 Sep 26 14:14 nm.service.keytab
-r-------- 1 hdfs hadoop 418 Sep 26 12:22 spnego.service.keytab
```

2. 配置yarn-site.xml

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
```

```
<configuration>

<property>
    <name>yarn.log.server.url</name>
    <value>https://cdh0.itcast.cn:19890/jobhistory/logs</value>
    <description></description>
</property>


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

  <property>
    <name>yarn.acl.enable</name>
    <value>false</value>
    <description>Enable ACLs? Defaults to false.</description>
  </property>


  <property>
    <name>yarn.admin.acl</name>
    <value>*</value>
    <description>ACL to set admins on the cluster. ACLs are of for comma-separated-
usersspacecomma-separated-groups. Defaults to special value of * which means anyone.
Special value of just space means no one has access.</description>
  </property>



  <property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
    <description>Configuration to enable or disable log aggregation</description>
  </property>

  <property>
    <name>yarn.nodemanager.remote-app-log-dir</name>
    <value>/tmp/logs</value>
    <description>Configuration to enable or disable log aggregation</description>
  </property>


  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>cdh0.itcast.cn</value>
    <description></description>
  </property>

<!--
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>cdh0.itcast.cn:8032</value>
    <description></description>
  </property>


  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>cdh0.itcast.cn:8030</value>
```

```
    <description></description>
  </property>


  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>cdh0.itcast.cn:8031</value>
    <description></description>
  </property>


  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>cdh0.itcast.cn:8033</value>
    <description></description>
  </property>

  <property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>0.0.0.0:8088</value>
    <description></description>
  </property>

-->




  <property>
    <name>yarn.resourcemanager.scheduler.class</name>

 <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</va
lue>
    <description></description>
  </property>

<!--
  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>64</value>
    <description>Minimum limit of memory to allocate to each container request at the
Resource Manager.</description>
  </property>


  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>2000</value>
    <description>Maximum limit of memory to allocate to each container request at the
Resource Manager.</description>
  </property>


  <property>
    <name>yarn.resourcemanager.nodes.include-path</name>
    <value>cdh1.itcast.cn,cdh2.itcast.cn</value>
    <description></description>
  </property>
```

```
  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>2000</value>
    <description>Resource i.e. available physical memory, in MB, for given
NodeManager</description>
  </property>


  <property>
    <name>yarn.nodemanager.vmem-pmem-ratio</name>
    <value>2.1</value>
    <description>Maximum ratio by which virtual memory usage of tasks may exceed
physical memory</description>
  </property>
-->

  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>/data/nm-local</value>
    <description>Comma-separated list of paths on the local filesystem where
intermediate data is written.</description>
  </property>


  <property>
    <name>yarn.nodemanager.log-dirs</name>
    <value>/data/nm-log</value>
    <description>Comma-separated list of paths on the local filesystem where logs are
written.</description>
  </property>


  <property>
    <name>yarn.nodemanager.log.retain-seconds</name>
    <value>10800</value>
    <description>Default time (in seconds) to retain log files on the NodeManager Only
applicable if log-aggregation is disabled.</description>
  </property>



  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
    <description>Shuffle service that needs to be set for Map Reduce applications.
</description>
  </property>


    <!-- ResourceManager security configs -->
    <property>
        <name>yarn.resourcemanager.principal</name>
        <value>rm/_HOST@ITCAST.CN</value>
    </property>
    <property>
        <name>yarn.resourcemanager.keytab</name>
```

```
        <value>/etc/security/keytabs/rm.service.keytab</value>
    </property>
    <property>
        <name>yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled</name>
        <value>true</value>
    </property>

    <!-- NodeManager security configs -->
    <property>
        <name>yarn.nodemanager.principal</name>
        <value>nm/_HOST@ITCAST.CN</value>
    </property>
    <property>
        <name>yarn.nodemanager.keytab</name>
        <value>/etc/security/keytabs/nm.service.keytab</value>
    </property>


    <!-- TimeLine security configs -->
    <property>
        <name>yarn.timeline-service.principal</name>
        <value>tl/_HOST@ITCAST.CN</value>
    </property>
    <property>
        <name>yarn.timeline-service.keytab</name>
        <value>/etc/security/keytabs/tl.service.keytab</value>
    </property>
    <property>
        <name>yarn.timeline-service.http-authentication.type</name>
        <value>kerberos</value>
    </property>
    <property>
        <name>yarn.timeline-service.http-authentication.kerberos.principal</name>
        <value>HTTP/_HOST@ITCAST.CN</value>
    </property>
    <property>
        <name>yarn.timeline-service.http-authentication.kerberos.keytab</name>
        <value>/etc/security/keytabs/spnego.service.keytab</value>
    </property>
    <!-- To enable SSL -->
    <property>
        <name>yarn.http.policy</name>
        <value>HTTPS_ONLY</value>
    </property>

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

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

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

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

```
    <value>/bigdata/hadoop-2.6.0-cdh5.14.4/bin/container-executor</value>
  </property>
</configuration>
```

3. 配置mapred.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
    <description></description>
  </property>

<!--

  <property>
    <name>mapreduce.map.memory.mb</name>
    <value>1024</value>
    <description>Larger resource limit for maps.</description>
  </property>



  <property>
    <name>mapreduce.map.java.opts</name>
    <value>-Xmx768M</value>
    <description></description>
  </property>



  <property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>1024</value>
    <description></description>
```

```
  </property>



  <property>
    <name>mapreduce.reduce.java.opts</name>
    <value>-Xmx2000M</value>
    <description></description>
  </property>



  <property>
    <name>mapreduce.task.io.sort.mb</name>
    <value>256</value>
    <description></description>
  </property>



  <property>
    <name>mapreduce.task.io.sort.factor</name>
    <value>50</value>
    <description></description>
  </property>


  <property>
    <name>mapreduce.reduce.shuffle.parallelcopies</name>
    <value>25</value>
    <description>Higher number of parallel copies run by reduces to fetch outputs from
very large number of maps.</description>
  </property>
-->

  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>cdh0.itcast.cn:10020</value>
    <description></description>
  </property>


  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>cdh0.itcast.cn:19888</value>
    <description></description>
  </property>


  <property>
    <name>mapreduce.jobhistory.intermediate-done-dir</name>
    <value>/mr-data/mr-history/tmp</value>
    <description></description>
  </property>


  <property>
    <name>mapreduce.jobhistory.done-dir</name>
```

```
    <value>/mr-data/mr-history/done</value>
    <description></description>
  </property>


    <property>
        <name>mapreduce.jobhistory.keytab</name>
        <value>/etc/security/keytabs/jhs.service.keytab</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.principal</name>
        <value>jhs/_HOST@ITCAST.CN</value>
    </property>


    <property>
        <name>mapreduce.jobhistory.webapp.spnego-principal</name>
        <value>HTTP/_HOST@ITCAST.CN</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.webapp.spnego-keytab-file</name>
        <value>/etc/security/keytabs/spnego.service.keytab</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.http.policy</name>
        <value>HTTPS_ONLY</value>
    </property>
</configuration>
```

4. 配置 container-executor.cfg

以root账户编辑

```
yarn.nodemanager.local-dirs=/data/nm-local
yarn.nodemanager.log-dirs=/data/nm-log
yarn.nodemanager.linux-container-executor.group=hadoop
#configured value of yarn.nodemanager.linux-container-executor.group
banned.users=bin
#comma separated list of users who can not run applications
min.user.id=100
#Prevent other super-users
allowed.system.users=root,yarn,hdfs,mapred,hive,dev
##comma separated list of system users who CAN run applications
```

5. 将配置文件复制到其他机器

`scp container-executor.cfg mapred-site.xml yarn-site.xml cdh1:/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop/`

`scp container-executor.cfg mapred-site.xml yarn-site.xml cdh2:/bigdata/hadoop-2.6.0-cdh5.14.4/etc/hadoop/`

6. 启动 HistoryServer

4.1 在cdh0机器，切换到 `mapred` 账户

4.2 执行： `kinit -kt /etc/security/keytabs/jhs.service.keytab jhs/cdh0.itcast.cn@ITCAST.CN`

4.3 执行： `mr-jobhistory-daemon.sh start historyserver`

4.4 浏览器打开： https://cdh0:19890

7. 启动 resourcemanager

7.1 在cdh0机器，切换到yarn账户

7.2 执行： `kinit -kt /etc/security/keytabs/rm.service.keytab rm/cdh0.itcast.cn@ITCAST.CN`

7.3 执行： `yarn-daemon.sh start resourcemanager`

7.4 浏览器打开： https://cdh0:8090

8. 启动 nodemanager

> 在cdh1 和 cdh2 分别执行

```
su - yarn

kinit -kt /etc/security/keytabs/nm.service.keytab nm/cdh1.itcast.cn@ITCAST.CN (cdh1机器)
kinit -kt /etc/security/keytabs/nm.service.keytab nm/cdh2.itcast.cn@ITCAST.CN (cdh2机器)

yarn-daemon.sh start nodemanager
```

浏览器打开 https://cdh0:8090/cluster/nodes 查看两个nodemanager是否正常连接成功

9. 提交MR程序测试

在cdh0机器上：

1. `su - yarn`
2. `kinit -kt /etc/security/keytabs/rm.service.keytab rm/cdh0.itcast.cn@ITCAST.CN`
3. `hadoop jar /bigdata/hadoop-2.6.0-cdh5.14.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.14.4.jar wordcount /readme.txt /tmp/wd-output1`

六：配置HIVE

1. 创建hive用户 `useradd hive -g hadoop` 执行 `chown -R hive:hadoop /bigdata/hive-1.1.0-cdh5.14.4`
2. 创建hive对应的kerberos账户

```
在cdh0节点
kadmin    进入 kerberos的admin后台

addprinc -randkey hive/cdh0.itcast.cn@ITCAST.CN
ktadd -k /etc/security/keytabs/hive.keytab hive/cdh0.itcast.cn@ITCAST.CN

chown hive:hadoop hive.keytab
chmod 400 hive.keytab

得到
-r-------- 1 hive   hadoop 418 Sep 26 17:07 hive.keytab
```

3. 上传安装包中提供的 `mysql-connector-java-5.1.33.jar` 到hive 的lib目录下
4. 为hive安装mysql `yum install -y mysql-server mysql mysql-devel`
5. 启动和设置mysql

   启动： `service mysqld start`

   开机自动启动： `chkconfig mysqld on`

   设置初始密码为root： `/usr/bin/mysqladmin -u root password 'root'`
6. 验证登录 `mysql -uroot -p root`
7. 配置hive-env.sh

```
cp hive-env.sh.template hive-env.sh
# vim hive-env.sh 新增
export JAVA_HOME=/usr/local/jdk1.8.0_221
export HADOOP_HOME=/bigdata/hadoop-2.6.0-cdh5.14.4
export HIVE_HOME=/bigdata/hive-1.1.0-cdh5.14.4
export HIVE_CONF_DIR=/bigdata/hive-1.1.0-cdh5.14.4/conf
```

8. 配置 `hive-site.xml`

   `vim hive-site.xml`

```
<configuration>
<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost:3306/hive?
createDatabaseIfNotExist=true</value>
</property>

<property>
```

```
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
</property>


<property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>root</value>
</property>


<property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>root</value>
</property>


<property>
    <name>hive.server2.authentication</name>
    <value>KERBEROS</value>
</property>


<property>
    <name>hive.server2.authentication.kerberos.principal</name>
    <value>hive/_HOST@ITCAST.CN</value>
</property>


<property>
    <name>hive.server2.authentication.kerberos.keytab</name>
    <value>/etc/security/keytabs/hive.keytab</value>
</property>

<property>
        <name>hive.metastore.sasl.enabled</name>
        <value>true</value>
</property>

<property>
        <name>hive.metastore.kerberos.keytab.file</name>
        <value>/etc/security/keytabs/hive.keytab</value>
</property>


<property>
        <name>hive.metastore.kerberos.principal</name>
        <value>hive/_HOST@ITCAST.CN</value>
</property>


<property>
        <name></name>
        <value></value>
</property>
</configuration>
```

9. 初始化hive 源数据库：`bin/schematool -dbType mysql -initSchema -verbose`

10. 认证hive的账户：`kinit -kt /etc/security/keytabs/hive.keytab hive/cdh0.itcast.cn@ITCAST.CN`
11. 启动hive相关服务

    启动metastore：`nohup bin/hive --service metastore > metastore.log 2>&1 &`

    启动hiveserver2：`nohup bin/hive --service hiveserver2 > hs2.log 2>&1 &`
12. 测试hive

    执行bin/hive : 成功进入后，执行create database test;

    退出
13. 测试bin/beeline

    执行bin/beeline

    进入后执行：`!connect jdbc:hive2://cdh0.itcast.cn:10000/default;principal=hive/cdh0.itcast.cn@ITCAST.CN`

    成功连接后执行：`create table test_temp(id int, name string, address string)`

    然后执行：`insert into test_temp values(1, "hahaha", "beijing"), (2, "heiheihei", "shanghai"), (3, "hohoho", "shenzhen");`

    执行：`select * from test_temp;`
14. 没有报错的话，hive配置完成

七：使用代码集成测试

上传安装包提供的 `Kerberos-test.jar` 到/tmp/下

测试HDFS

切换到hdfs账户

cd /tmp

执行 `hadoop jar Kerberos-test.jar com.itheima.kerberos.test.FSTools`

正确输出：

```
===查询===
rwxr-xr-x       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/mr-data
rw-r--r--       3       1366    2019-09-26 16:15:50     hdfs://cdh0.itcast.cn:8020/readme.txt
rwxrwxrwx       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/tmp
rwxrwxr-x       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/user
===上传===
===查询===
rwxr-xr-x       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/mr-data
rw-r--r--       3       571815  2019-09-26 17:42:09     hdfs://cdh0.itcast.cn:8020/nn.log
rw-r--r--       3       1366    2019-09-26 16:15:50     hdfs://cdh0.itcast.cn:8020/readme.txt
rwxrwxrwx       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/tmp
rwxrwxr-x       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/user
===下载===
===删除===
true
===查询===
rwxr-xr-x       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/mr-data
rw-r--r--       3       1366    2019-09-26 16:15:50     hdfs://cdh0.itcast.cn:8020/readme.txt
rwxrwxrwx       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/tmp
rwxrwxr-x       0       0       1970-01-01 08:00:00     hdfs://cdh0.itcast.cn:8020/user
```

> 代码中使用的路径是基于本文档所配置的如/bigdata/hadoop-2.6.0-cdh5.14.4, 如果同学们的路径和这个jar包的不一致，可以使用附带的代码自行修改后打包出jar文件再执行

测试YARN

切换到yarn账户

cd /tmp

```
hadoop jar Kerberos-test.jar com.itheima.kerberos.test.WordCountApp /readme.txt /tmp/wd-output12 rm/cdh0.itcast.cn@ITCAST.CN /etc/security/keytabs/rm.service.keytab
```

> 上面命令接受4个参数 参数1 hdfs输入 参数2 wordcount输出 参数3 kerberos的账户 参数4 keytab文件路径
>
> 如果和你的不一致，请自行修改

测试HIVE

切换到hive用户

cd /tmp

执行：`hadoop jar Kerberos-test.jar com.itheima.kerberos.test.HS2Tools`

正确输出如下：

```
[hive@cdh0 tmp]$ hadoop jar Kerberos-test.jar com.itheima.kerberos.test.HS2Tools
19/09/26 17:52:04 INFO security.UserGroupInformation: Login successful for user hive/cdh0.itcast.cn@ITCA
ST.CN using keytab file /etc/security/keytabs/hive.keytab
19/09/26 17:52:04 INFO jdbc.Utils: Supplied authorities: cdh0.itcast.cn:10000
19/09/26 17:52:04 INFO jdbc.Utils: Resolved authority: cdh0.itcast.cn:10000
===显示表===
test_temp
===建表详情===
id      int
name    string
address string
===查询表===
1,hahaha,beijing
2,heiheihei,shanghai
3,hohoho,shenzhen
===删除表===
===显示表===
test_temp
```

参考资料：

https://blog.csdn.net/kwame211/article/details/78728989

https://www.jianshu.com/p/fc2d2dbd510b

https://blog.csdn.net/lovebomei/article/details/80004277