Instalación de kerberos

Después de preparar nuestro cluster pasamos a la instalación de kerberos
yum install krb5-server krb5-workstation

[*]: Hemos instalado el servicio de tiempo en la máquina y lo hemos activado, en un cluster deberíamos hacerlo en todas las máquinas. Kerberos es sensible al tiempo, necesita que los servidores tengan los relojes ajustados, la diferencia por defecto no debe superar los 300 segundos.
https://web.mit.edu/kerberos/krb5-1.4/krb5-1.4.4/doc/krb5-admin/Clock-Skew.html

Ficheros de configuración
/etc/krb5.conf
/var/kerberos/krb5kdc/kadm5.acl
/var/kerberos/krb5kdc/kdc.conf

"Casualmente" hemos utilizado kerberos como nombre de máquina y example.com como dominio, por lo que los ficheros de configuración por defecto _casi_ nos valdrían ;-)

[*]: Hemos instalado la versión de encriptación fuerte de java por lo que podemos usar AES256, en caso de no desplegar este tipo de cifrado tendríamos que deshabilitarlo en kerberos ya que los procesos en Java no serían capaces de autenticar.

Modificamos /var/kerberos/krb5kdc/kdc.conf para añadir unos valores por defecto para nuestro realm.

```
# cat << EOF >  /var/kerberos/krb5kdc/kdc.conf
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
EXAMPLE.COM = {
  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
  default_principal_flags = +preauth +forwardable +proxiable +renewable
  max_renewable_life = 7d
  max_life = 24h
}
EOF
```

Ya podemos inicializar nuestra base de datos de kerberos
kdb5_util create -s

Creamos un usuario administrador
kadmin.local -q "addprinc root/admin"

Creamos identidades en kerberos
kadmin.local -q "addprinc bofh"
kadmin.local -q "addprinc foouser"
kadmin.local -q "addprinc baruser"

Habilitamos los servicios
chkconfig krb5kdc on
chkconfig kadmin on

Arrancamos kerberos
service krb5kdc start
service kadmin start

Verificamos que nuestro sistemas de tickets funciona creando un ticket para el usuario administrador
kinit root/admin
Listamos los ticket y comprobamos que en los falgs aparece el flag R (renovable)
klist -ef
Probamos a conectar a la consola de administración de kerberos
kadmin
> listprincs
> quit
Por último destruimos el ticket

Realizamos la prueba de solicitar un ticket, listar el ticket disponible y destruirlo con los usuarios foouser, baruser y bofh. Podemos realizar las pruebas como root, no es necesario cambiar al usuario de correspondiente del sistema.

Kerberizado de Hadoop

Securizamos permisos en el filesystem de la máquina
chmod 700 /var/lib/hadoop-hdfs/cache/hdfs/dfs/name

Preparamos HDFS para la securizacion
sudo -u hdfs hadoop fs -chown hdfs:hadoop /
sudo -u hdfs hadoop fs -chmod 755 /
sudo -u hdfs hadoop fs -mkdir -p /user/history
sudo -u hdfs hadoop fs -chmod -R 1777 /user/history
sudo -u hdfs hadoop fs -chown mapred:hadoop /user/history

Creamos las identidades en kerberos
kadmin.local -q "addprinc -randkey hdfs/kerberos.example.com@EXAMPLE.COM"
kadmin.local -q "addprinc -randkey mapred/kerberos.example.com@EXAMPLE.COM"
kadmin.local -q "addprinc -randkey yarn/kerberos.example.com@EXAMPLE.COM"
kadmin.local -q "addprinc -randkey HTTP/kerberos.example.com@EXAMPLE.COM"

Creamos los keytabs
cd /etc/hadoop/conf
kadmin.local -q "xst -norandkey -k hdfs.keytab
hdfs/kerberos.example.com@EXAMPLE.COM
HTTP/kerberos.example.com@EXAMPLE.COM”
kadmin.local -q "xst -norandkey -k mapred.keytab
mapred/kerberos.example.com@EXAMPLE.COM
HTTP/kerberos.example.com@EXAMPLE.COM”
kadmin.local -q "xst -norandkey -k yarn.keytab
yarn/kerberos.example.com@EXAMPLE.COM
HTTP/kerberos.example.com@EXAMPLE.COM”

chown hdfs:hadoop /etc/hadoop/conf/hdfs.keytab
chown mapred:hadoop /etc/hadoop/conf/mapred.keytab
chmod 400 /etc/hadoop/conf/*.keytab
chown yarn:hadoop /etc/hadoop/conf/yarn.keytab
chmod 400 /etc/hadoop/conf/yarn.keytab

Apagamos el cluster
for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x stop ; done

Editamos el fichero /etc/hadoop/conf/core-site.xml y nos aseguramos de que tenga estos valores.

```
...
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value> <!-- A value of "simple" would disable security. -->
</property>

<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
</property>
…
```

Editamos /etc/hadoop/conf/hdfs-site.xml y añadimos la siguiente configuración

```
…
<!-- General HDFS security config -->
<property>
  <name>dfs.block.access.token.enable</name>
  <value>true</value>
</property>

<!-- NameNode security config -->
<property>
  <name>dfs.namenode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.namenode.kerberos.principal</name>
  <value>hdfs/_HOST@EXAMPLE.COM</value>
</property>
<property>
  <name>dfs.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@EXAMPLE.COM</value>
</property>

<!-- Secondary NameNode security config -->
<property>
  <name>dfs.secondary.namenode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.secondary.namenode.kerberos.principal</name>
  <value>hdfs/_HOST@EXAMPLE.COM</value>
</property>
<property>
```

```
  <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@EXAMPLE.COM</value>
</property>

<!-- DataNode security config -->
<property>
  <name>dfs.datanode.data.dir.perm</name>
  <value>700</value>
</property>
<property>
  <name>dfs.datanode.address</name>
  <value>0.0.0.0:1004</value>
</property>
<property>
  <name>dfs.datanode.http.address</name>
  <value>0.0.0.0:1006</value>
</property>
<property>
  <name>dfs.datanode.keytab.file</name>
  <value>/etc/hadoop/conf/hdfs.keytab</value> <!-- path to the HDFS keytab -->
</property>
<property>
  <name>dfs.datanode.kerberos.principal</name>
  <value>hdfs/_HOST@EXAMPLE.COM</value>
</property>

<!-- Web Authentication config -->
<property>
  <name>dfs.web.authentication.kerberos.principal</name>
  <value>HTTP/_HOST@EXAMPLE.COM</value>
 </property>
….


Add to /etc/default/hadoop-hdfs-datanode
cat << EOF >> /etc/default/hadoop-hdfs-datanode
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_PID_DIR=/var/lib/hadoop-hdfs
export HADOOP_SECURE_DN_LOG_DIR=/var/log/hadoop-hdfs
export JSVC_HOME=/usr/lib/bigtop-utils/
EOF

Arrancamos el Namenode
# service hadoop-hdfs-namenode start
```

Verificamos que se ve la estructura de directorios
hadoop fs -ls /user

Edit /etc/init.d/hadoop-hdfs-datanode.sh and change PIDFILE to:
PIDFILE="/var/lib/hadoop-hdfs/hadoop-hdfs-datanode.pid"

Arrancamos el Datanode
service hadoop-hdfs-datanode start

Arrancamos el SecondaryNamenode
service hadoop-hdfs-secondarynamenode start

Ahora debe funcionar todo HDFS, probamos a copiar un fichero y borrarlo para verificar el correcto funcionamiento.
kinit foouser
hadoop fs -put /etc/services
hadoop fs -ls
hadoop fs -cat services
hadoop fs -rm services
hadoop fs -ls

Editamos el fichero /etc/hadoop/conf/yarn-site.xml y añadimos estas configuraciones
```
<!-- ResourceManager security configs -->
<property>
  <name>yarn.resourcemanager.keytab</name>
  <value>/etc/hadoop/conf/yarn.keytab</value>
<!-- path to the YARN keytab -->
</property>
<property>
  <name>yarn.resourcemanager.principal</name>
  <value>yarn/_HOST@EXAMPLE.COM</value>
</property>
<!-- NodeManager security configs -->
<property>
  <name>yarn.nodemanager.keytab</name>
  <value>/etc/hadoop/conf/yarn.keytab</value>
<!-- path to the YARN keytab -->
</property>
<property>
  <name>yarn.nodemanager.principal</name>
  <value>yarn/_HOST@EXAMPLE.COM</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.group</name>
  <value>yarn</value>
</property>
```

Tras esto editamos el fichero /etc/hadoop/conf/mapred-site.xml y añadimos la configuración:

```
<!-- MapReduce JobHistory Server security configs -->
<property>
  <name>mapreduce.jobhistory.address</name>
  <value>kerberos.example.com:10020</value> <!-- Host and port of the MapReduce
JobHistory Server; default port is 10020  -->
</property>
<property>
  <name>mapreduce.jobhistory.keytab</name>
  <value>/etc/hadoop/conf/mapred.keytab</value>
<!-- path to the MAPRED keytab for the JobHistory Server -->
</property>
<property>
  <name>mapreduce.jobhistory.principal</name>
  <value>mapred/_HOST@EXAMPLE.COM</value>
</property>
```

Create /etc/hadoop/conf/container-executor.cfg

```
# cat << EOF > /etc/hadoop/conf/container-executor.cfg
yarn.nodemanager.local-dirs=/var/lib/hadoop-yarn/cache/yarn/nm-local-dir
yarn.nodemanager.linux-container-executor.group=yarn
yarn.nodemanager.log-dirs=/var/log/hadoop-yarn/containers
banned.users=hdfs,yarn,mapred,bin
min.user.id=500
EOF
```

Secure permissions for container executor binary

```
# chmod 2150 /usr/lib/hadoop-yarn/bin/container-executor
# chmod u+s /usr/lib/hadoop-yarn/bin/container-executor
```

Arrancamos el ResourceManager

```
# service hadoop-yarn-resourcemanager start
```

Arrancamos el NodeManager

```
# service hadoop-yarn-nodemanager start
```

Arrancamos el HistoryServer

```
# service hadoop-mapreduce-historyserver start
```

Para verificar que todo funciona correctamente ejecutamos un programa de ejemple
kinit foouser
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar pi 1 1

Ya tenemos funcionando un cluster kerberizado! :D

En tres cómodos pasos….