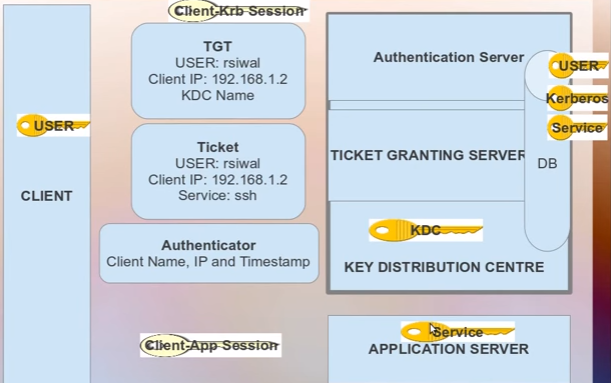
Quick Ref Kerberos

**Kerberos vs Sentry**

[Kerberos](http://en.wikipedia.org/wiki/Kerberos_%28protocol%29) enables you to authenticate users in your Hadoop cluster. For example, it guarantees that it is really the user ‘bob’ and not ‘joe’ that is submitting a job, listing files or doing a search. Next step is configuring what the user can access, this is called [authorization](http://en.wikipedia.org/wiki/Authorization).

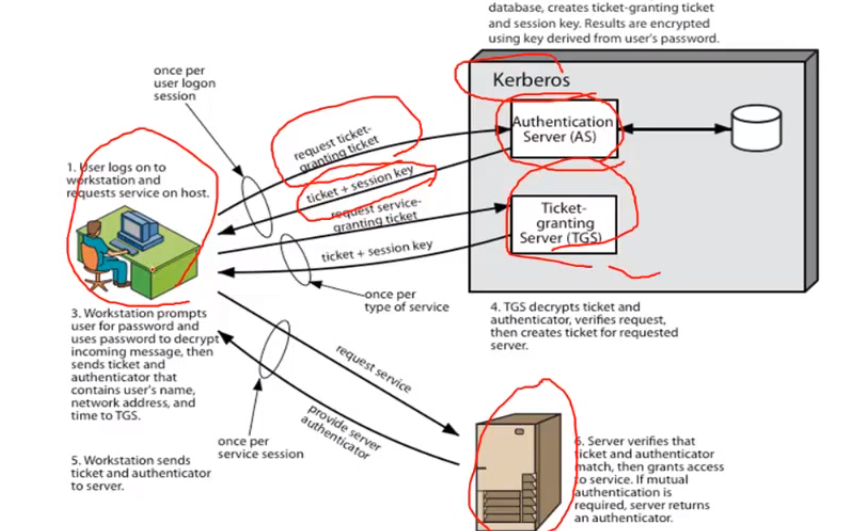
[Sentry](https://sentry.incubator.apache.org/) is the secure way to define who can see, query, add data in the Solr collections/indexes. This is only possible as we guarantee the usernames performing the actions with Kerberos.

http://gethue.com/hadoop-tutorial-kerberos-security-and-sentry-authorization-for-solr-search-app/

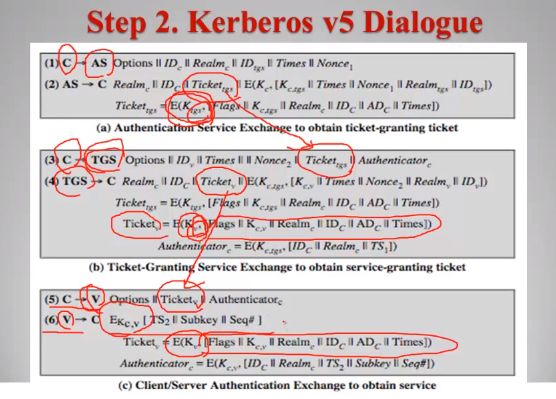


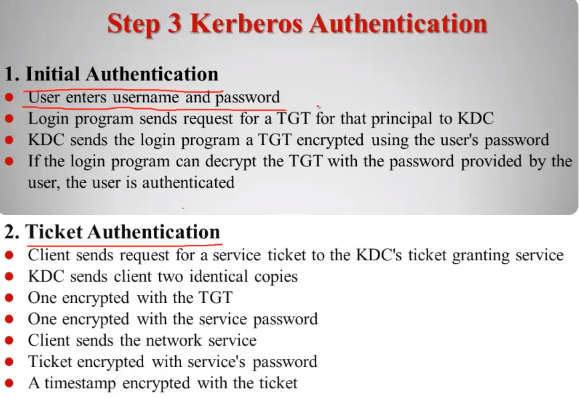
**Kerberos Version 4 and Version 5**

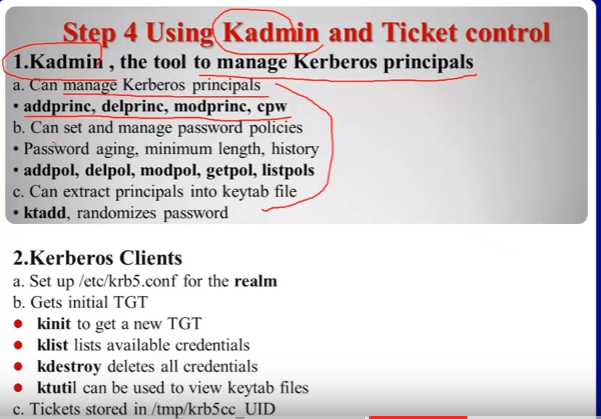
**Version4**

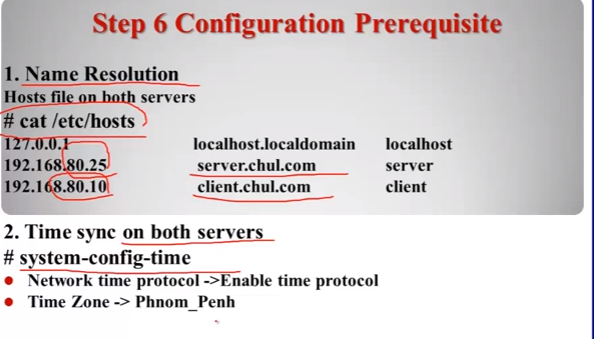


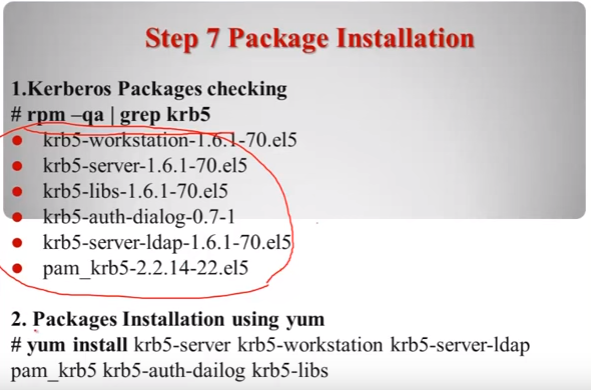
**Version 5**

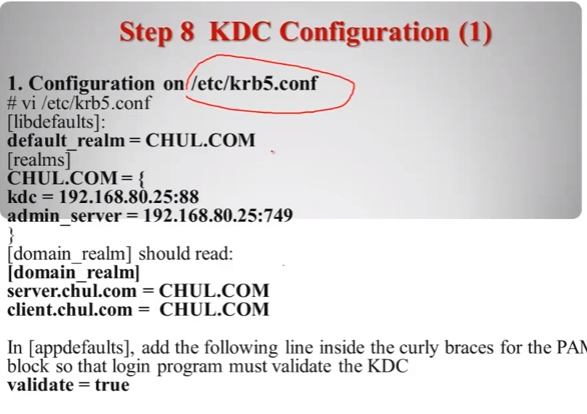


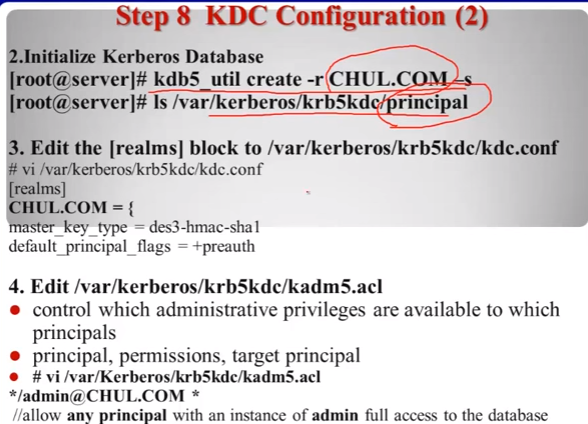


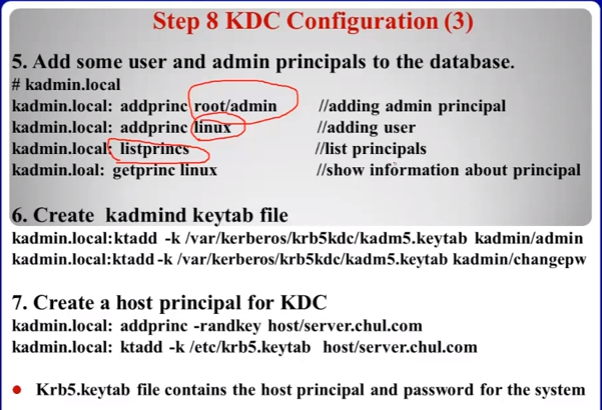


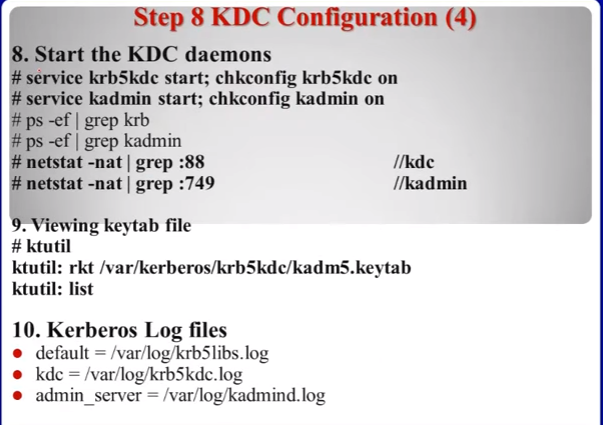




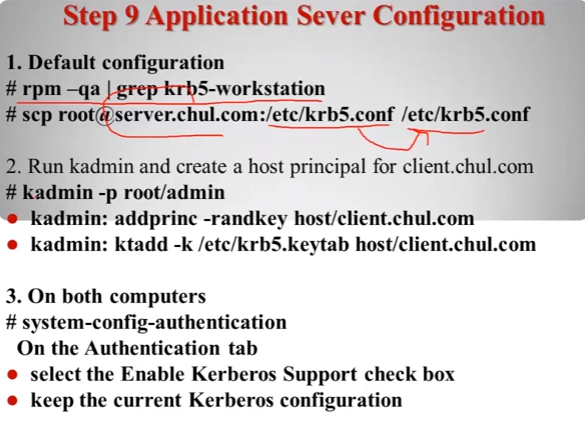


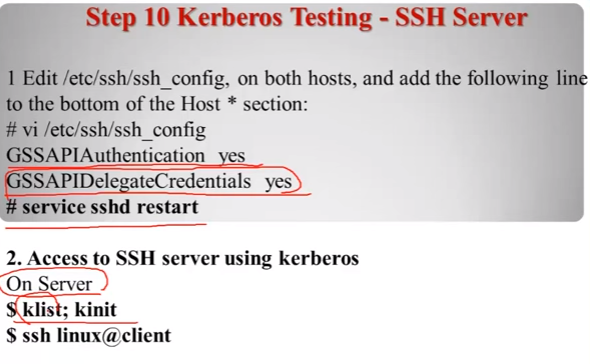


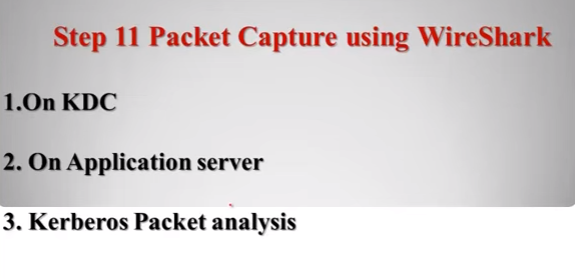




Note: Step 9 to copy the krb5.conf file from server to client







|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Desc** |
|  | **Web Reference** |  |
| \* | Understand the usage of Kerberos | <https://www.youtube.com/watch?v=S20e5ZrU2-o> |
| \* | How does Kerberos works in public network | <https://www.youtube.com/watch?v=KD2Q-2ToloE> |
| \* | Kerberos in detail | <https://www.youtube.com/watch?v=XLBAhz735mg> |
| \* | Theory & Lab – Part 1 | <https://www.youtube.com/watch?v=Vo-huwPFqz8> |
| \* | Theory & Lab – Part 2 | <https://www.youtube.com/watch?v=38k5ZLqVbI0> |
|  |  |  |
|  |  |  |
|  | **Kerberos Installation** |  |
| 1 | **In Master Node** | Login as root |
| 1.1 | Mount to cdrom | >mount /dev/cdrom /media/ |
| 1.2 | Search Kerberos | >yum search krb5-server  Ex: it may return some info as follows   1. Krb5-server.x86\_64: The KDC and related prg for Kerberos 2. Krb5-server-ldap.i686: The LDAP storage plugin 3. Krb5-server-ldap.x86\_64: The LDAP storage plugin |
| 1.3 | Install Kerberos | >yum install krb5-server.x86\_64 krb5-workstation |
| 2 | **Define REALM** | In master node,   1. configure REALM 2. client configuration 3. configure KDC |
| 2.1 | Client Configuration | >hostname -- To get the hostname. REALM will part of hostname in capital letter |
| 2.2 | Client Configuration | >vim /etc/krb5.conf  # krb5.conf has 4 parts   1. logging 2. libdefaults 3. realms 4. domain\_realms |
| 2.3 | Client Configuration | **# logging**  Default=FILE:/var/log/krb5libs.log  Kdc=FILE:/var/log/krb5kdc.log  Admin\_server=FILE:/var/log/kadmind.log  **#libdefaults**  Default\_realm=part of HOSTNAME(In CAPS)  Dns\_lookup\_realm=false  Dns\_lookup\_kdc=false  Ticket\_lifetime=24h  Renew\_lifetime=7d  Forwardable=true  **#realms**  Part of HOSTNAME(in CAPS)={  Kdc=full hostname (in small)  Admin\_server=full hostname (in small)  **#domain\_realms**  .Part of hostname(starts with . and small case) = part of HOSTNAME(upper)  Part of hostname(small case) = part of HOSTNAME(upper) |
| 2.4 | Verify | 1. Sudo yum install bind-utils |
| 3 | Package Installation | Refer step 7 – screen shot from above *-- TBD*  <https://www.youtube.com/watch?v=38k5ZLqVbI0&spfreload=1> |
| 4 | Package Configuration | Refer step 8 – screen shot from above – TBD (same youtube video)  >kadmin.local  Kadmin.local : addprinc hduser  Kadmin.local : listprinc  Kadmin.local : addprinc -randkey host/server.chul.com  Note: chul.com is a REALM name  Kadmin.local : ktadd -k /etc/krb5.keytab host/server.chul.com  Kadmin.local : exit |
| 5 | Application Server configuration | Refer step 9 – screen shot from above – *TBD* (same youtube video)  Note: Step 9 to copy the krb5.conf file from server to client  >kadmin -p root/admin *-- To login as admin* |
| 6 | Configuration validation | # Login as root. To check the configuration (it will open a dialog box with all configuration)  > system-config-authentication |
| 7 | Testing Kerberos | Kerberos testing – ssh server  Refer step 10 from above screen shot – TBD |
| 8 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **General Information** | Version 4 and 5 (two important versions) |
|  | About Kerberos | 1. Kerberos is an authentication protocol for trusted hosts on untrusted network. 2. Based around credentials called tickets 3. Tickets secured by secret key encryption |
|  | Diff: Kerberos vs Telnet vs LDAP | 1. **Telnet:** Credential flow through the network 2. **Tls, ssl** 3. **LDAP:** credential flow through the network but encrypted. 4. Ssha algorithm -- To be verified 5. Sha algorithm -- To be verified 6. Kript/script algoritm -- To be verified 7. **Kerberos:** credential (especially) password will not flow in network |
|  | Time configuration in network nodes  *<Pre-request>* | # login as root   1. >system-config-time 2. Step1: It will show a dialog box, go to ‘Network Time Protocol’ tab and click ‘Enable Network Time Protocol’ 3. Step2: Go to ‘Timezone’ tab and choose the time zone |
| 1 | Components/ Participant of Kerberos | Kerberos has 3 principles/components (Kerberos means 3 headed dogs)   1. Kerberos Client (User) : User/Client 2. Kerberos Key Distribution Center (KDC) : Server (KDC knows all passwords)   KDC has 2 sub-components   1. AS: Authentication Server 2. TGS: Ticket Granting Server 3. Kerberized Service (Application services, network services, etc Ex: Mail server, ssh) : Service |
| 1.1 | Principles of Kerberos | # Principle identifies each participant in Kerberos authentication   1. Users and Networks services are identified by Primary, Instance and REALM 2. Ex: [root/admin@CHUL.COM](mailto:root/admin@CHUL.COM) 3. In the above example, root is Primary 4. Admin is Instance 5. CHUL.COM is REALM (REALM is domain name) |
| 2 | Aims of Kerberos | 1. The User’s password will never travel over network 2. The User’s password will never be stored in any form on the client machine. It must be discarded immediately after use 3. The User’s password will never be stored in an unencrypted form even in authentication server database 4. Single sign on: **Kerberos provides SSO.** The user is asked to enter a password only once per working session. Because Kerberos generates TGT and **TGT will be re-used every time you access the server. So no need to enter password again and again for the ‘same session’** |
| 3 | **TGT** | Ticket Granting Ticket (TGT)   1. Client/User presents TGT to Application Service for authentication 2. TGT contains: UserID, ClientIP, KDC Name 3. Client Cannot read TGT, TGT can be read by Application Server |
| 4 | Connection between KDC and TGT | 1. Consider a user wants to access his server. He enters his user id and password in login screen. It will generate a packet and it will contain ONLY user name (no password) and the packet will send to KDC to generate Ticket Granting Ticket (TGT). Also the packet will contain 2. Access raised Time stamp for the session 3. Access expired Time stamp for the session 4. First KDC will check whether the user is a valid user, if so it will generate TGT (KDC will derive the encrypted key- temporary or session specific password based on actual password. Note: The actual password will not travel in network, KDC generates session specific password) 5. TGT is session specific and it was generated by KDC and it contains 6. UserName 7. KDC Name 8. Client’s IP 9. Session Key (Session specific generated by KDC. It changes periodically means session specific, so secured) 10. Then KDC sends back TGT along with Session Key (Session key is encrypted with user’s private key) 11. Note: At this point, access is not granted to user. KDC just sent the TGT will session key. TGT is still unopened by User 12. Now, Again User will send packet (it contains username, IP and time stamp for the session) + TGT to KDC 13. Now, KDC will compare the packet with unopened TGT. If matches, then KDC trust the identity of the user. So KDC will send ‘service ticket’ to user (with new session key) 14. Service ticket: It is specific for requested service . (NOT for all the services in the single sign on. Ex: only mail service) and ONLY works for the particular user and it will work for only specific time 15. Now, User will send the new session key to Kerberized Service (it can be Mail server, etc) 16. Kerberized Service will check the service was authorized by KDC. Check User name, ip , service from session matches to KDC, then check for authentication and provide access |
| 5 | REALM (reem)  *<REALM is Kerberos domain name>*  **Note**: REALM is case sensitive. So Always use REALM in capital letter | 1. REALM is an Authentication Administrative Domain 2. REALM is a collection of principles that belong to the same domain 3. Principle means: Any entry in Kerberos DB, it could be a user, server, service. It uses the following format: Name[/Instance]@REALM. 4. To be verified. Ex: hostname/ipAddress@REALM 5. Key Distribution Center (KDC): It has 3 components 6. Authentication Server (AS) 7. Ticket Granting Server (TGS) 8. Application Server 9. sdf |
| 6 | Default time for session | Usually 24 hours… we can change it during KDC Configuration |
| 7 | Kerberos **disabled** | /user/kumar  /user/abc  # Login with kumar’s login  >ssh kumar@server  # list all the file that ‘kumar’ owns  >Hadoop dfs –ls /user/kumar/  # list all the file that ‘abc’ owns. So kumar can see other users data  >Hadoop dfs –ls /user/abc/ |
| 8 | Kerberos **enabled** | /user/kumar  /user/abc  # Login with kumar’s login  >ssh kumar@server  # list all the file that ‘kumar’ owns  >Hadoop dfs –ls /user/kumar/  # list all the file that ‘abc’ owns  >Hadoop dfs –ls /user/abc/  Kerbero through errors tht kumar doesn’t have permission to see user ‘abc’’s data |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **General Commands** |  |
| 1 | KAdmin | # KAdmin is the tool to manage Kerberos Principle  Can Manage Kerberos Principles. Ex:   1. >addprinc -- Add principle 2. >delprinc -- Delete principle 3. >modprinc -- Modify principle 4. >cpw -- Change password   Can set and manage password policies. Password aging, minimum length, history:   1. >addpol 2. >delpol 3. >modpol 4. >getpol 5. >listpol   Can extract principles into keytab (kt) files:   1. >ktadd 2. >randomized password |
| 2 | Kerberos Clients | 1. Set up /etc/krb5.conf for the realm: 2. Gets Initial TGT: 3. >kinit --- to get the new TGT 4. >klist --- to list available credentials 5. >kdestroy --- to delete all credentials 6. >ktutil -- to view keytab(kt) utility files 7. Tickets stored in /tmp/krb5cc\_UID   <https://www.youtube.com/watch?v=S20e5ZrU2-o> |
| 3 | Start the service | >service krb5kdc start  >service kadmin start  >ps –ef | grep krb5  >ps –ef | grep kadmin  >netstat -nat | grep 88 *-- 88 is the port number*  >netstat -nat | grep 749 *-- 749 is the port number* |
| 4 | >kadmin.local | To provide local commands like  Kadmin.local : addprinc hduser  Kadmin.local : listprinc  Kadmin.local : addprinc -randkey host/server.chul.com  Note: chul.com is a REALM name  Kadmin.local : ktadd -k /etc/krb5.keytab host/server.chul.com  Kadmin.local : exit |
| 5 | >kutil | >kutil  # To list keytab details  Kutil : rkt /var/Kerberos/krb5kdc/kadm5.keytab  >quit |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

<https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/Superusers.html>

1. Which two features does Kerberos security add to a Hadoop cluster?
2. User authentication on all remote procedure calls (RPCs)
3. Authentication for user access to the cluster against a central server
4. sdf

**Code example**

In this example super's kerberos credentials are used for login and a proxy user ugi object is created for joe. The operations are performed within the doAs method of this proxy user ugi object.

...

//Create ugi for joe. The login user is 'super'.

UserGroupInformation ugi =

UserGroupInformation.createProxyUser("joe", UserGroupInformation.getLoginUser());

ugi.doAs(new PrivilegedExceptionAction<Void>() {

public Void run() throws Exception {

//Submit a job

JobClient jc = new JobClient(conf);

jc.submitJob(conf);

//OR access hdfs

FileSystem fs = FileSystem.get(conf);

fs.mkdir(someFilePath);

}

}

### Configurations

The superuser must be configured on namenode and jobtracker to be allowed to impersonate another user. Following configurations are required.

<property>

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

<value>group1,group2</value>

<description>Allow the superuser super to impersonate any members of the group group1 and group2</description>

</property>

<property>

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

<value>host1,host2</value>

<description>The superuser can connect only from host1 and host2 to impersonate a user</description>

</property>

If these configurations are not present, impersonation will not be allowed and connection will fail.

If more lax security is preferred, the wildcard value \* may be used to allow impersonation from any host or of any user.

<https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html>

### ACLs (Access Control Lists)

In addition to the traditional POSIX permissions model, HDFS also supports POSIX ACLs (Access Control Lists). ACLs are useful for implementing permission requirements that differ from the natural organizational hierarchy of users and groups. An ACL provides a way to set different permissions for specific named users or named groups, not only the file's owner and the file's group.

By default, support for ACLs is disabled, and the NameNode disallows creation of ACLs. To enable support for ACLs, set dfs.namenode.acls.enabled to true in the NameNode configuration.

An ACL consists of a set of ACL entries. Each ACL entry names a specific user or group and grants or denies read, write and execute permissions for that specific user or group. For example:

user::rw-

user:bruce:rwx #effective:r--

group::r-x #effective:r--

group:sales:rwx #effective:r--

mask::r--

other::r--

ACL entries consist of a type, an optional name and a permission string. For display purposes, ':' is used as the delimiter between each field. In this example ACL, the file owner has read-write access, the file group has read-execute access and others have read access. So far, this is equivalent to setting the file's permission bits to 654.

Additionally, there are 2 extended ACL entries for the named user bruce and the named group sales, both granted full access. The mask is a special ACL entry that filters the permissions granted to all named user entries and named group entries, and also the unnamed group entry. In the example, the mask has only read permissions, and we can see that the effective permissions of several ACL entries have been filtered accordingly.

Every ACL must have a mask. If the user doesn't supply a mask while setting an ACL, then a mask is inserted automatically by calculating the union of permissions on all entries that would be filtered by the mask.

Running chmod on a file that has an ACL actually changes the permissions of the mask. Since the mask acts as a filter, this effectively constrains the permissions of all extended ACL entries instead of changing just the group entry and possibly missing other extended ACL entries.

The model also differentiates between an "access ACL", which defines the rules to enforce during permission checks, and a "default ACL", which defines the ACL entries that new child files or sub-directories receive automatically during creation. For example:

user::rwx

group::r-x

other::r-x

default:user::rwx

default:user:bruce:rwx #effective:r-x

default:group::r-x

default:group:sales:rwx #effective:r-x

default:mask::r-x

default:other::r-x

Only directories may have a default ACL. When a new file or sub-directory is created, it automatically copies the default ACL of its parent into its own access ACL. A new sub-directory also copies it to its own default ACL. In this way, the default ACL will be copied down through arbitrarily deep levels of the file system tree as new sub-directories get created.

The exact permission values in the new child's access ACL are subject to filtering by the mode parameter. Considering the default umask of 022, this is typically 755 for new directories and 644 for new files. The mode parameter filters the copied permission values for the unnamed user (file owner), the mask and other. Using this particular example ACL, and creating a new sub-directory with 755 for the mode, this mode filtering has no effect on the final result. However, if we consider creation of a file with 644 for the mode, then mode filtering causes the new file's ACL to receive read-write for the unnamed user (file owner), read for the mask and read for others. This mask also means that effective permissions for named user bruce and named group sales are only read.

### Security

The following section outlines the additional changes needed to migrate a secure cluster.

New YARN Kerberos service principals should be created for the ResourceManager and NodeManager, using the pattern used for other Hadoop services, i.e. yarn@<HOST>. The mapred principal should still be used for the JobHistoryServer. If you are using Cloudera Manager to configure security, this will be taken care of automatically.

As in MRv1, a configuration must be set to have the user that submits a job own its task processes. The equivalent of MRv1’s LinuxTaskController is the LinuxContainerExecutor. In a secure setup, NodeManager configurations should set yarn.nodemanager.container-executor.class toorg.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor. Properties set in the taskcontroller.cfg configuration file should be migrated to their analagous properties in the container-executor.cfg file.

In secure setups, configuring hadoop-policy.xml allows administrators to set up access control lists on internal protocols. The following is a table of MRv1 options and their MRv2 equivalents:

<http://www.cloudera.com/documentation/cdh/5-1-x/CDH5-Installation-Guide/cdh5ig_mapreduce_to_yarn_migrate.html>

Queue access control lists (ACLs) are now placed in the Fair Scheduler configuration file instead of the JobTracker configuration. A list of users and groups that can submit jobs to a queue can be placed in aclSubmitApps in the queue’s configuration. The queue administration ACL is no longer supported, but will be in a future release.

MRv1 MRv2 Comment

security.task.umbilical.protocol.acl security.job.task.protocol.acl As in MRv1, this should never be set to anything other than \*

security.inter.tracker.protocol.acl security.resourcetracker.protocol.acl

security.job.submission.protocol.acl security.applicationclient.protocol.acl

security.admin.operations.protocol.acl security.resourcemanager-administration.protocol.acl

security.applicationmaster.protocol.acl No MRv1 equivalent

security.containermanagement.protocol.acl No MRv1 equivalent

security.resourcelocalizer.protocol.acl No MRv1 equivalent

security.job.client.protocol.acl No MRv1 equivalen

**Pending questions**

1. What is POSIX permissions model?