Verify User Accounts and Groups in CDH 5 Due to Security

Verify User Accounts and Groups in YARN

During CDH 5 package installation of MapReduce 2.0 (YARN), the following Unix user accounts are automatically created to support security:

This User	Runs These Hadoop Programs
hdfs	HDFS: NameNode, DataNodes, Standby NameNode (if you are using HA)
yarn	YARN: ResourceManager, NodeManager
mapred	YARN: MapReduce JobHistory Server

Important: The HDFS and YARN daemons must run as different Unix users; for example, hdfsand yarn. The MapReduce JobHistory server must run as user mapred. Having all of these users share a common Unix group is recommended; for example, hadoop.

Directory Ownership in the Local Filesystem

Because the HDFS and MapReduce services run as different users, you must be sure to configure the correct directory ownership of the following files on the local filesystem of each host:

File System	Directory	Owner	Permissions (see Footnote 1)
Local	dfs.namenode.name.dir(dfs.name.dir is deprecated but will also work)	hdfs:hdfs	drwx------
Local	dfs.datanode.data.dir(dfs.data.dir is deprecated but will also work)	hdfs:hdfs	drwx------

File System	Directory	Owner	Permissions (see Footnote 1)
Local	yarn.nodemanager.local-dirs	yarn:yarn	drwxr-xr-x
Local	yarn.nodemanager.log-dirs	yarn:yarn	drwxr-xr-x
Local	container-executor	root:yarn	--Sr-s---
Local	conf/container-executor.cfg	root:yarn	r--------

Important: Configuration changes to the Linux container executor could result in local NodeManager directories (such as usercache) being left with incorrect permissions. To avoid this, when making changes using either Cloudera Manager or the command line, first manually remove the existing NodeManager local directories from all configured local directories (yarn.nodemanager.local-dirs), and let the NodeManager recreate the directory structure.

You must also configure the following permissions for the HDFS, YARN and MapReduce log directories (the default locations in /var/log/hadoop-hdfs, /var/log/hadoop-yarn and /var/log/hadoop-mapreduce):

File System	Directory	Owner	Permissions
Local	HDFS_LOG_DIR	hdfs:hdfs	drwxrwxr-x
Local	$YARN_LOG_DIR	yarn:yarn	drwxrwxr-x
Local	MAPRED_LOG_DIR	mapred:mapred	drwxrwxr-x

Directory Ownership on HDFS

The following directories on HDFS must also be configured as follows:

File System	Directory	Owner	Permissions
HDFS	/ (root directory)	hdfs:hadoop	drwxr-xr-x
HDFS	yarn.nodemanager.remote-app-log-dir	yarn:hadoop	drwxrwxrwxt
HDFS	mapreduce.jobhistory.intermediate-done-dir	mapred:hadoop	drwxrwxrwxt
HDFS	mapreduce.jobhistory.done-dir	mapred:hadoop	drwxr-x---

YARN: Changing the Directory Ownership on HDFS

If Hadoop security is enabled, use kinit hdfs to obtain Kerberos credentials for the hdfs user by running the following commands:

```
$ sudo -u hdfs kinit -k -t hdfs.keytab
hdfs/fully.qualified.domain.name@YOUR-REALM.COM

$ hadoop fs -chown hdfs:hadoop /

$ hadoop fs -chmod 755 /
```

If kinit hdfs does not work initially, run kinit -R after running kinit to obtain credentials. See Troubleshooting Authentication Issues. To change the directory ownership on HDFS, run the following commands:

```
$ sudo -u hdfs hadoop fs -chown hdfs:hadoop /

$ sudo -u hdfs hadoop fs -chmod 755 /

$ sudo -u hdfs hadoop fs -chown yarn:hadoop [yarn.nodemanager.remote-
app-log-dir]

$ sudo -u hdfs hadoop fs -chmod 1777 [yarn.nodemanager.remote-app-log-
dir]

$ sudo -u hdfs hadoop fs -chown mapred:hadoop
[mapreduce.jobhistory.intermediate-done-dir]
```

```
$ sudo -u hdfs hadoop fs -chmod 1777
[mapreduce.jobhistory.intermediate-done-dir]

$ sudo -u hdfs hadoop fs -chown mapred:hadoop
[mapreduce.jobhistory.done-dir]

$ sudo -u hdfs hadoop fs -chmod 750 [mapreduce.jobhistory.done-dir]
```

- In addition (whether or not Hadoop security is enabled) create the /tmp directory. For instructions on creating /tmp and setting its permissions, see Step 7: If Necessary, Create the HDFS /tmp Directory.
- In addition (whether or not Hadoop security is enabled), change permissions on the /user/history Directory. See Step 8: Create the history Directory and Set Permissions.

1 In CDH 5, package installation and the Hadoop daemons will automatically configure the correct permissions for you if you configure the directory ownership correctly as shown in the table above.
2 When starting up, MapReduce sets the permissions for the mapreduce.jobtracker.system.dir (or mapred.system.dir) directory in HDFS, assuming the user mapred owns that directory.
3 In CDH 5, package installation and the Hadoop daemons will automatically configure the correct permissions for you if you configure the directory ownership correctly as shown in the two tables above. See also Deploying MapReduce v2 (YARN) on a Cluster.

If you are using CentOS/Red Hat Enterprise Linux 5.6 or higher, or Ubuntu, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user machines. For JCE Policy File installation instructions, see the README.txt file included in the jce_policy-x.zip file.

Alternatively, you can configure Kerberos to not use AES-256 by removing aes256-cts:normal from the supported_enctypes field of the kdc.conf or krb5.conf file. After changing the kdc.conf file, you must restart both the KDC and the kadmin server for those changes to take affect. You may also need to re-create or change the password of the relevant principals, including potentially the Ticket Granting Ticket principal (krbtgt/REALM@REALM). If AES-256 is still used after completing steps, the aes256-cts:normal setting existed when the Kerberos database was created. To fix this, create a new Kerberos database and then restart both the KDC and the kadmin server.

To verify the type of encryption used in your cluster:

1.On the local KDC host, type this command to create a test principal:

If you are using CentOS/Red Hat Enterprise Linux 5.6 or higher, or Ubuntu, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user machines. For JCE Policy File installation instructions, see the README.txt file included in the jce_policy-x.zip file.

2.

3.Alternatively, you can configure Kerberos to not use AES-256 by removing aes256-cts:normal from the supported_enctypes field of the kdc.conf or krb5.conf file. After changing the kdc.conf file, you must restart both the KDC and the kadmin server for those changes to take affect. You may also need to re-create or change the password of the relevant principals, including potentially the Ticket Granting Ticket principal (krbtgt/REALM@REALM). If AES-256 is still used after completing steps, the aes256-cts:normal setting existed when the Kerberos database was created. To fix this, create a new Kerberos database and then restart both the KDC and the kadmin server.

4.

5.To verify the type of encryption used in your cluster:

6.

7.On the local KDC host, type this command to create a test principal:

8.$ kadmin -q "addprinc test"

9.On a cluster host, type this command to start a Kerberos session as test:

10.$ kinit test

11.On a cluster host, type this command to view the encryption type in use:

12.$ klist -e

13.If AES is being used, output like the following is displayed after you type the klist command; note that AES-256 is included in the output:

14.

15.Ticket cache: FILE:/tmp/krb5cc_0

```
Default principal: test@SCM

Valid starting     Expires            Service principal

05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/SCM@SCM

    Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256 CTS mode with $ kadmin -q "addprinc test"
```

On a cluster host, type this command to start a Kerberos session as test:

```
$ kinit test
```

On a cluster host, type this command to view the encryption type in use:

```
$ klist -e
```

If AES is being used, output like the following is displayed after you type the klist command; note that AES-256 is included in the output:

```
Ticket cache: FILE:/tmp/krb5cc_0

Default principal: test@SCM

Valid starting     Expires            Service principal

05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/SCM@SCM

    Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256 CTS mode with
```

Step 4: Create and Deploy the Kerberos Principals and Keytab Files

A Kerberos principal is used in a Kerberos-secured system to represent a unique identity. Kerberos assigns tickets to Kerberos principals to enable them to access Kerberos-secured Hadoop services. For Hadoop, the principals should be of the format username/fully.qualified.domain.name@YOUR-REALM.COM. In this guide, the term username in the username/fully.qualified.domain.name@YOUR-REALM.COM principal refers to the username of an existing Unix account, such as hdfs or mapred.

A keytab is a file containing pairs of Kerberos principals and an encrypted copy of that principal's key. The keytab files are unique to

each host since their keys include the hostname. This file is used to authenticate a principal on a host to Kerberos without human interaction or storing a password in a plain text file. Because having access to the keytab file for a principal allows one to act as that principal, access to the keytab files should be tightly secured. They should be readable by a minimal set of users, should be stored on local disk, and should not be included in machine backups, unless access to those backups is as secure as access to the local machine.

Important:

On every machine in your cluster, there must be a keytab file for the hdfs user and a keytab file for the mapred user. The hdfs keytab file must contain entries for the hdfs principal and a HTTP principal, and the mapred keytab file must contain entries for the mapred principal and a HTTP principal. On each respective machine, the HTTP principal will be the same in both keytab files.

On every machine in your cluster, there must be a keytab file for the yarn user. The yarn keytab file must contain entries for the yarn principal and a HTTPprincipal. On each respective machine, the HTTP principal in the yarn keytab file will be the same as the HTTP principal in the hdfs and mapred keytab files.

Note:

The following instructions illustrate an example of creating keytab files for MIT Kerberos. If you are using another version of Kerberos, refer to your Kerberos documentation for instructions. You may use either kadmin or kadmin.local to run these commands.

When to Use kadmin.local and kadmin

When creating the Kerberos principals and keytabs, you can use kadmin.local or kadmin depending on your access and account:

- If you have root access to the KDC machine, but you do not have a Kerberos admin account, use kadmin.local.
- If you do not have root access to the KDC machine, but you do have a Kerberos admin account, use kadmin.
- If you have both root access to the KDC machine and a Kerberos admin account, you can use either one.

To start kadmin.local (on the KDC machine) or kadmin from any machine, run this command:

```
$ sudo kadmin.local
```

OR:

```
$ kadmin
```

Note:

In this guide, kadmin is shown as the prompt for commands in the kadmin shell, but you can type the same commands at the kadmin.local prompt in the kadmin.local shell.

Note:

Running kadmin.local may prompt you for a password because it is being run via sudo. You should provide your Unix password. Running kadmin may prompt you for a password because you need Kerberos admin privileges. You should provide your Kerberos admin password.

To create the Kerberos principals

Do the following steps for every host in your cluster. Run the commands in the kadmin.local or kadmin shell, replacing the fully.qualified.domain.name in the commands with the fully qualified domain name of each host. Replace YOUR-REALM.COM with the name of the Kerberos realm your Hadoop cluster is in.

1. In the kadmin.local or kadmin shell, create the hdfs principal. This principal is used for the NameNode, Secondary NameNode, and DataNodes.

```
kadmin:  addprinc -randkey hdfs/fully.qualified.domain.name@YOUR-REALM.COM
```

Note:
If your Kerberos administrator or company has a policy about principal names that does not allow you to use the format shown above, you can work around that issue by configuring the<kerberos principal> to <short name> mapping that is built into Hadoop. For more information, see Configuring the Mapping from Kerberos Principals to Short Names.

2. Create the mapred principal. If you are using MRv1, the mapred principal is used for the JobTracker and TaskTrackers. If you are using YARN, the mapred principal is used for the MapReduce Job History Server.

```
kadmin:  addprinc -randkey mapred/fully.qualified.domain.name@YOUR-REALM.COM
```

3. **YARN only:** Create the yarn principal. This principal is used for the ResourceManager and NodeManager.

```
kadmin:  addprinc -randkey yarn/fully.qualified.domain.name@YOUR-
REALM.COM
```

4. Create the HTTP principal.

```
kadmin:  addprinc -randkey HTTP/fully.qualified.domain.name@YOUR-
REALM.COM
```

Important:
The HTTP principal must be in the format HTTP/fully.qualified.domain.name@YOUR-REALM.COM. The first component of the principal must be the literal string "HTTP". This format is standard for HTTP principals in SPNEGO and is hard-coded in Hadoop. It cannot be deviated from.

To create the Kerberos keytab files

Important:

The instructions in this section for creating keytab files require using the Kerberos norandkeyoption in the xst command. If your version of Kerberos does not support the norandkey option, or if you cannot use kadmin.local, then use these alternate instructions to create appropriate Kerberos keytab files. After using those alternate instructions to create the keytab files, continue with the next section To deploy the Kerberos keytab files.

Do the following steps for every host in your cluster. Run the commands in the kadmin.local or kadmin shell, replacing the fully.qualified.domain.name in the commands with the fully qualified domain name of each host:

1. Create the hdfs keytab file that will contain the hdfs principal and HTTP principal. This keytab file is used for the NameNode, Secondary NameNode, and DataNodes.

```
kadmin:  xst -norandkey -k hdfs.keytab
hdfs/fully.qualified.domain.name HTTP/fully.qualified.domain.name
```

2. Create the mapred keytab file that will contain the mapred principal and HTTP principal. If you are using MRv1, the mapred keytab file is used for the JobTracker and TaskTrackers. If you are using YARN, the mapred keytab file is used for the MapReduce Job History Server.

```
kadmin:  xst -norandkey -k mapred.keytab
mapred/fully.qualified.domain.name HTTP/fully.qualified.domain.name
```

3. **YARN only:** Create the yarn keytab file that will contain the yarn principal and HTTP principal. This keytab file is used for the ResourceManager and NodeManager.

```
kadmin:  xst -norandkey -k yarn.keytab
yarn/fully.qualified.domain.name HTTP/fully.qualified.domain.name
```

4. Use klist to display the keytab file entries; a correctly-created hdfs keytab file should look something like this:

```
$ klist -e -k -t hdfs.keytab

Keytab name: WRFILE:hdfs.keytab

slot KVNO Principal

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

   1    7    HTTP/fully.qualified.domain.name@YOUR-REALM.COM (DES cbc
mode with CRC-32)

   2    7    HTTP/fully.qualified.domain.name@YOUR-REALM.COM (Triple
DES cbc mode with HMAC/sha1)

   3    7    hdfs/fully.qualified.domain.name@YOUR-REALM.COM (DES cbc
mode with CRC-32)

   4    7    hdfs/fully.qualified.domain.name@YOUR-REALM.COM (Triple
DES cbc mode with HMAC/sha1)
```

5. Continue with the next section To deploy the Kerberos keytab files.

To deploy the Kerberos keytab files

On every node in the cluster, repeat the following steps to deploy the hdfs.keytab and mapred.keytab files. You will also deploy the yarn.keytab file.

1. On the host machine, copy or move the keytab files to a directory that Hadoop can access, such as /etc/hadoop/conf.

a.

```
$ sudo mv hdfs.keytab mapred.keytab yarn.keytab /etc/hadoop/conf/
```

b. Make sure that the hdfs.keytab file is only readable by the hdfs user, and that the mapred.keytab file is only readable by the mapred user.

```
$ sudo chown hdfs:hadoop /etc/hadoop/conf/hdfs.keytab
$ sudo chown mapred:hadoop /etc/hadoop/conf/mapred.keytab
$ sudo chmod 400 /etc/hadoop/conf/*.keytab
```

Note:
To enable you to use the same configuration files on every host, we recommend that you use the same name for the keytab files on every host.

c. Make sure that the yarn.keytab file is only readable by the yarn user.

```
$ sudo chown yarn:hadoop /etc/hadoop/conf/yarn.keytab
$ sudo chmod 400 /etc/hadoop/conf/yarn.keytab
```

Step 5: Shut Down the Cluster

To enable security in Hadoop, you must stop all Hadoop daemons in your cluster and then change some configuration properties. You must stop all daemons in the cluster because after one Hadoop daemon has been restarted with the configuration properties set to enable security, daemons running without security enabled will be unable to communicate with that daemon. This requirement to shut down all daemons makes it impossible to do a rolling upgrade to enable security on a Hadoop cluster.

To shut down the cluster, run the following command on every node in your cluster (as root):

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

Step 6: Enable Hadoop Security

We recommend that all of the Hadoop configuration files throughout the cluster have the same contents.

To enable Hadoop security, add the following properties to the core-site.xml file *on every machine*in the cluster:

```
<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>
```

Enabling Service-Level Authorization for Hadoop Services

Service-level authorizations prevent users from accessing a cluster at the course-grained level. For example, when Authorized Users and Authorized Groups are setup properly, an unauthorized user cannot use the hdfs shell to list the contents of HDFS. This also limits the exposure of world-readable files to an explicit set of users instead of all authenticated users, which could be, for example, every user in Active Directory.

The hadoop-policy.xml file maintains access control lists (ACL) for Hadoop services. Each ACL consists of comma-separated lists of users and groups separated by a space. For example:

```
user_a,user_b group_a,group_b
```

If you only want to specify a set of users, add a comma-separated list of users followed by a blank space. Similarly, to specify only authorized groups, use a blank space at the beginning. A * can be used to give access to all users.

For example, to give users, ann, bob, and groups, group_a, group_b access to Hadoop's DataNodeProtocol service, modify the security.datanode.protocol.acl property in hadoop-policy.xml. Similarly, to give all users access to the InterTrackerProtocol service, modify security.inter.tracker.protocol.acl as follows:

```
<property>
    <name>security.datanode.protocol.acl</name>
    <value>ann,bob group_a,group_b</value>
```

```
    <description>ACL for DatanodeProtocol, which is used by datanodes
to

    communicate with the namenode.</description>

</property>


<property>

    <name>security.inter.tracker.protocol.acl</name>

    <value>*</value>

    <description>ACL for InterTrackerProtocol, which is used by
tasktrackers to

    communicate with the jobtracker.</description>

</property>
```

Step 7: Configure Secure HDFS

When following the instructions in this section to configure the properties in the hdfs-site.xml file, keep the following important guidelines in mind:

- The properties for each daemon (NameNode, Secondary NameNode, and DataNode) must specify both the HDFS and HTTP principals, as well as the path to the HDFS keytab file.
- The Kerberos principals for the NameNode, Secondary NameNode, and DataNode are configured in the hdfs-site.xml file. The same hdfs-site.xml file with *all three* of these principals must be installed on every host machine in the cluster. That is, it is not sufficient to have the NameNode principal configured on the NameNode host machine only. This is because, for example, the DataNode must know the principal name of the NameNode in order to send heartbeats to it. Kerberos authentication is bi-directional.
- The special string _HOST in the properties is replaced at run-time by the fully qualified domain name of the host machine where the daemon is running. This requires that reverse DNS is properly working on all the hosts configured this way. You may use _HOST only as the entirety of the second component of a principal name. For example, hdfs/_HOST@YOUR-REALM.COM is valid, but hdfs._HOST@YOUR-REALM.COM and hdfs/_HOST.example.com@YOUR-REALM.COM are not.

- When performing the _HOST substitution for the Kerberos principal names, the NameNode determines its own hostname based on the configured value of fs.default.name, whereas the DataNodes determine their hostnames based on the result of reverse DNS resolution on the DataNode hosts. Likewise, the JobTracker uses the configured value of mapred.job.tracker to determine its hostname whereas the TaskTrackers, like the DataNodes, use reverse DNS.
- The dfs.datanode.address and dfs.datanode.http.address port numbers for the DataNode *must* be below 1024, because this provides part of the security mechanism to make it impossible for a user to run a map task which impersonates a DataNode. The port numbers for the NameNode and Secondary NameNode can be anything you want, but the default port numbers are good ones to use.
  Continue reading:

- To configure secure HDFS
- To enable TLS/SSL for HDFS

To configure secure HDFS

Add the following properties to the hdfs-site.xml file *on every machine* in the cluster. Replace these example values shown below with the correct settings for your site: *path to the HDFS keytab*, *YOUR-REALM.COM*, *fully qualified domain name of NN*, and *fully qualified domain name of 2NN*

```
<!-- 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@YOUR-REALM.COM</value>
</property>
<property>
  <name>dfs.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@YOUR-REALM.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@YOUR-REALM.COM</value>
</property>
<property>

<name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@YOUR-REALM.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@YOUR-REALM.COM</value>
</property>

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

To enable TLS/SSL for HDFS

Add the following property to hdfs-site.xml on *every machine* in your cluster.

```
<property>
<name>dfs.http.policy</name>
<value>HTTPS_ONLY</value>
</property>
```

Step 11: Set Variables for Secure DataNodes
In order to allow DataNodes to start on a secure Hadoop cluster, you must set the following variables on all DataNodes in /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/libexec/bigtop-utils/
```

Step 12: Start up the NameNode
You are now ready to start the NameNode. Use the service command to run the /etc/init.d script.

```
$ sudo service hadoop-hdfs-namenode start
```

You'll see some extra information in the logs such as:

```
10/10/25 17:01:46 INFO security.UserGroupInformation:

Login successful for user hdfs/fully.qualified.domain.name@YOUR-
REALM.COM using keytab file /etc/hadoop/conf/hdfs.keytab
```

and:

```
12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO)
filter to getDelegationToken

12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO)
filter to renewDelegationToken

12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO)
filter to cancelDelegationToken

12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO)
filter to fsck

12/05/23 18:18:31 INFO http.HttpServer: Adding Kerberos (SPNEGO)
filter to getimage

12/05/23 18:18:31 INFO http.HttpServer: Jetty bound to port 50070

12/05/23 18:18:31 INFO mortbay.log: jetty-6.1.26
```

```
12/05/23 18:18:31 INFO server.KerberosAuthenticationHandler: Login
using keytab /etc/hadoop/conf/hdfs.keytab, for principal
HTTP/fully.qualified.domain.name@YOUR-REALM.COM

12/05/23 18:18:31 INFO server.KerberosAuthenticationHandler:
Initialized, principal [HTTP/fully.qualified.domain.name@YOUR-
REALM.COM] from keytab [/etc/hadoop/conf/hdfs.keytab]
```

You can verify that the NameNode is working properly by opening a web browser to http://*machine*:50070/ where *machine* is the name of the machine where the NameNode is running.

We also recommend testing that the NameNode is working properly by performing a metadata-only HDFS operation, which will now require correct Kerberos credentials. For example:

```
$ hadoop fs -ls
```

Step 13: Start up a DataNode

Begin by starting one DataNode only to make sure it can properly connect to the NameNode. Use the service command to run the /etc/init.d script.

```
$ sudo service hadoop-hdfs-datanode start
```

You'll see some extra information in the logs such as:

```
10/10/25 17:21:41 INFO security.UserGroupInformation:

Login successful for user hdfs/fully.qualified.domain.name@YOUR-
REALM.COM using keytab file /etc/hadoop/conf/hdfs.keytab
```

If you can get a single DataNode running and you can see it registering with the NameNode in the logs, then start up all the DataNodes. You should now be able to do all HDFS operations.