Cloudera Professional Services

Security Engagement for The Brattle Group

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Author** | **Description** | **Date** |
| 0.1 | Prashant Sharma | Initial draft after first week | 6/12/2015 |
| 0.2 | Rachel Asher | Added updates from week two: MySQL replication, H/A, move to non-embedded databases, other updates | 7/21/2015 |
| 0.3 | Mubashir Kazia | Update SSSD config to reflect use of Unix/Posix attributes in AD | 9/24/2015 |
| 0.4 | Rachel Asher | SSSD Enablement Process | 10/7/2015 |

Contents

1 Introduction 5

2 Authentication 5

2.1 Kerberos Configuration 5

2.2 Workaround for the absence of LDAPS 5

2.3 Configuration for handling usernames with whitespace and upper case characters 5

2.4 HDFS admin and superuser groups 6

3 Identity Management 6

4 Authorization 8

4.1 HDFS ACLs 8

4.2 Sentry 8

4.3 Synchronizing HDFS ACLs and Sentry Permissions 9

5 Encryption 9

5.1 Preparing Certificates, Keystores and Truststores 9

5.2 Setup SSL for Cloudera Manager 11

5.3 Setup SSL for CDH components 11

5.4 Encryption for Data at rest 12

6 Move from Embedded Database to Production-ready MySQL Database 12

7 MySQL Installation and Replication setup 13

8 High Availability for NameNode services and YARN 15

8.1 HDFS NameNode High Availability 15

8.2 YARN High Availability 16

9 Additional Changes Made to Prepare Cluster for Production 17

9.1 Upgrade from 5.3 to 5.4.2-1 17

9.2 Reconfigure HDD’s on NameNodes to have separate spindles for Journal Node Service and ZooKeeper services. 17

10 Appendix 18

10.1 Changes to the “import\_credentials.sh” script 18

10.2 Changes to the “gen\_credentials\_ad.sh” script 20

10.3 smb.conf for cmbhdpdn008.brattle.net 23

10.4 krb5.conf 24

10.5 sssd.conf 24

# Introduction

This document describes the activities/tasks performed for the Security engagement for The Brattle Group. The cluster that we worked on had CM/CDH 5.3.3 installed on it.

# Authentication

This section describes the configuration done on the cluster to enable authentication for the various components of the Cloudera stack.

## Kerberos Configuration

We followed the “Direct-to-AD” method to enable Kerberos authentication for the cluster. The cluster was kerberized using the Cloudera Manager (CM) wizard to enable Kerberos. The process to do this is documented in detail on the following link:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/cm_sg_intro_kerb.html>

## Workaround for the absence of LDAPS

The above documentation assumes that LDAPS is enabled on the AD side. This was not the case at The Brattle Group. To work around this, we had to change 2 scripts that CM uses to enable Kerberos. The Changes in the scripts are documented in the appendix section ["Changes to the import\_credentials.sh script"](#_Changes_to_the) and ["Changes to the gen\_credentials\_ad.sh script "](#_Changes_to_the_1). Please note that these files will get erased with an upgrade of the Cloudera Manager software, so it is highly recommended to make a backup of the unchanged and the changed file so that these changes can be made to these files in the future versions of CM installed at The Brattle Group. The generate\_credentials\_ad.sh script is invoked whenever a new service is added to the cluster so this change is not just applicable for the initial setup.

## Configuration for handling usernames with whitespace and upper case characters

The Brattle Group has usernames that contain a whitespace (<Fisrtname> <Lastname>). The first alphabet of the first name and the last name are upper case. Linux in general doesn’t handle both these scenarios very well. To handle this scenario we configured the following rule in CM-> HDFS-> Configuration -> “Additional Rules to Map Kerberos Principals to Short Names”:

*RULE:[1:$1](.\*\S.\*)s/ /\_/g/L*

The syntax of the mapping rule is described in great detail on the following links:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/cm_sg_kerbprin_to_sn.html>

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/cdh_sg_kerbprin_to_sn.html?scroll=topic_19_1_unique_3>

Using this rule, a username like “Dwayne [Robinson@BRATTLE.NET](mailto:Robinson@BRATTLE.NET)” will be converted to “dwayne\_robinson”. To so this same conversion on the Linux level, we do some configuration, which is explained in the “Identity Management” section.

## HDFS admin and superuser groups

The “hadoop\_admins” group was made HDFS admin by setting the configuration CM->HDFS->Configuration -> “**Authorized Admin Groups**” to the value “hadoop\_admins”. Similarly, the “hadoop\_admins” group was made HDFS super user by setting the configuration CM-> HDFS-> Configuration -> “**Superuser Group**” to the value “hadoop\_admins”.

# Identity Management

The CDH services need the user account and group information to be present on each node of the cluster so that group-based authorization can be done correctly. To achieve this, we used a tool called SSSD (System Security Services Daemon). **Please note that this is a component that is not supported by Cloudera, so future support issues can’t be handled by Cloudera support**. Some other products that provide similar functionality are Centrify (<https://www.centrify.com/>) and Authentication Services (<http://software.dell.com/products/authentication-services/>).

Please note that there is an assumption in these steps that the file “/etc/krb5.conf” has been properly setup. For us it was setup as a part of configuring Kerberos for the cluster. The “krb5.conf” file that was generated by CM as a result of the process to setup Kerberos is listed in the Appendix section [“krb5.conf”](#_krb5.conf).

Before configuring SSSD, AD should be prepared by installing Identity Management for Unix <https://msdn.microsoft.com/en-us/library/cc731178.aspx>. This should be installed on all the AD Domain Controllers. Once installed ensure “Domain Computers” group has access to read all the POSIX attributes in AD. Next create a separate OU for Groups that will be used for the cluster. All the Groups and Users that’ll be used on the cluster should be Unix enabled (The users should have uidNumber and should have gidNumber assigned. The users should also be assigned a primary group.)

Following are the steps followed to configure SSSD on each host:

1. Disable caching for passwd and group in nscd. In /etc/nscd.conf make changes below and restart nscd.

*enable-cache passwd no*

*enable-cache group no*

1. Change the Samba configuration file (/etc/samba/smb.conf) in preparation to add the node to the AD domain. The Samba configuration file is listed in Appendix section ["smb.conf for cmbhdpdn008.brattle.net"](#_smb.conf_for_cmbhdpdn008.brattle.net) . **Please note that the “netbios name” is specific to each host.**
2. Run the following command to add the node to the AD domain. Please note that we are using the “cmadmin” account to authenticate to AD for this step:

*net ads join createupn=host/*cmbhdpdn008.brattle.net*@*[*BRATTLE.NET*](http://brattle.net/)*-U cmadmin -S* [*cmbdc1.brattle.net*](http://cmbdc1.brattle.net/)

1. Create a keytab so that the node can authenticate to AD for pulling user and group information using the following command:

*net ads keytab create -U cmadmin*

1. A keytab will be created in the path “/etc/krb5.keytab”. You can test this keytab by running the following command:

*klist -k -t /etc/krb5.keytab*

1. Install SSSD and related packages:

*yum install sssd-client sssd-tools sssd*

1. Change the sssd.conf file as listed in the Appendix section [“sssd.conf”](#_sssd.conf). Please note the **“override\_space” setting** that has been added to replace the whitespace in the user name with an “\_” character. After changing sssd.conf, change the permissions on the file as follows:

*chmod 600 /etc/sssd/sssd.conf*

1. Start the SSSD service

*service sssd start*

1. Run the following command so that SSSD comes up automatically on a machine reboot:

*chkconfig sssd on*

1. Test the SSSD service by running the following commands and check that the information returned is correct:

*id kclark*

*groups project1*

*getent passwd*

*getent group*

# Authorization

This section describes the steps taken to configure authorization for the cluster.

## HDFS ACLs

We enabled HDFS ACLs by going to CM->HDFS->Configuration and checking the checkbox for the property “Enable Access Control Lists”.

## Sentry

The Sentry service was installed by following the documentation below:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/sg_sentry_service_install.html>

The Sentry service was enabled for Hive, Impala and HUE by following the documentation below:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/sg_sentry_service_config.html?scroll=concept_ul2_tkx_p4_unique_2#concept_z5b_42s_p4_unique_1>

Please note that the “hadoop\_admins” group was set as the Sentry Admin group in Cloudera Manager. This setting can be accessed by going to CM-> Sentry -> Configuration -> “Admin Groups

”.

## Synchronizing HDFS ACLs and Sentry Permissions

We enabled synchronization of Sentry permissions to HDFS ACLs by following the documentation below:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/sg_hdfs_sentry_sync.html>

# Encryption

This section describes the process to setup Encryption for data in transit.

## Preparing Certificates, Keystores and Truststores

Following are the steps to prepare certificates, keystores and truststores for a node (cmbhdpdn008.brattle.net) in the cluster. Please note that these steps need to be executed for all the nodes in the cluster

1. Create a Certificate Signing Request (CSR) as shown below. This command will ask for information about the organization and the common name (CN) for the certificate. The CN should be the FQDN of the hostname for which the certificate is being requested. So for the example shown below the CN should be “cmbhdpdn008.brattle.net”. Also, the key or the CSR should not be protected by a passphrase as some components in Hadoop can’t work with private keys that have a passphrase:

*openssl req -out cmbhdpdn008.brattle.net.csr.pem -new -newkey rsa:2048 -nodes -keyout .cmbhdpdn008.brattle.net.key*

1. Send the CSR file (cmbhdpdn008.brattle.net*.csr.pem*) created above to the certificates team and get a Base-64/PEM format certificate from them. We will call this certificate “cmbhdpdn008.brattle.net.cer” for these steps.
2. Get the root CA certificate and the issuing CA certificate in BASE-64/PEM format. We will call these certificates “tbg\_root\_ca.cer” and “tbg\_issuing\_ca.cer” for these steps.
3. Create a combined certificate file that has the node certificate and the issuing authority certificate:

*cat cmbhdpdn008.brattle.net.cer > cmbhdpdn008.brattle.net.combined.cer*

*cat tbg\_issuing\_ca.cer >> cmbhdpdn008.brattle.net.combined.cer*

1. Create a PKCS 12 file which encapsulates the private key and the certificate of a host (combined with the issuing CA cert)

*openssl pkcs12 -export -in cmbhdpdn008.brattle.net.combined.cer -inkey cmbhdpdn008.brattle.net.key -out cmbhdpdn008.brattle.net.p12 -name cmbhdpdn008.brattle.net -passin pass: -passout pass:password*

1. Create a Java keystore. Please keep all the passwords highlighted in red below and in the steps above the same to reduce confusion. Also, some components in the Hadoop stack don’t support different passwords for the keystore and the private key.

*/usr/java/jdk1.7.0\_67-cloudera/bin/keytool -importkeystore -srckeystore cmbhdpdn008.brattle.net.p12 -srcstoretype PKCS12 -srcstorepass password -alias cmbhdpdn008.brattle.net -deststorepass password -destkeypass password -destkeystore server.jks*

1. Create a java truststore. In the below steps we are first creating a copy of the default java truststore which ships with the JDK. Then we are importing the root CA cert for “The brattle group” in that truststore. Finally, we are changing the password from the default password of “changeit” to the password “password”. It is again recommended to keep this password the same as the password that you chose in the earlier steps to reduce confusion.

*cp /usr/java/jdk1.7.0\_67-cloudera/jre/lib/security/cacerts jssecacerts*

*/usr/java/jdk1.7.0\_67-cloudera/bin/keytool -importcert -file tbg\_root\_ca.cer -alias tbg\_root\_ca --keystore jssecacerts -storepass changeit*

*/usr/java/jdk1.7.0\_67-cloudera/bin/keytool -storepasswd -new password -keystore jssecacerts*

1. Create the following folder hierarchy for storing these certificates:

*mkdir /opt/cloudera/security/truststore*

*mkdir /opt/cloudera/security/jks/*

*mkdir /opt/cloudera/security/x509/*

1. Move the files that we created above to the respective folders as shown below:

*cp server.jks /opt/cloudera/security/jks/*

*cp jssecacerts /opt/cloudera/security/truststore/*

*cp cmbhdpdn008.brattle.net.combined.cer /opt/cloudera/security/x509/server.pem*

*cp .cmbhdpdn008.brattle.net.key /opt/cloudera/security/x509/.server.key.pem*

*cp tbg\_root\_ca.cer /opt/cloudera/security/truststore/cacert.pem*

## Setup SSL for Cloudera Manager

SSL/TLS for Cloudera manager was setup by following the documentation below:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/cm_sg_config_tls_security.html>

Please note that we setup Level 1 configuration for Cloudera manager.

## Setup SSL for CDH components

SSL/TLS was setup for the CDH components installed on the cluster by following the documentation below:

<http://www.cloudera.com/content/cloudera/en/documentation/core/v5-3-x/topics/cm_sg_hadoop_ssl_cm.html>

## Encryption for Data at rest

Encryption for data at rest was de-scoped from the SOW as “The Brattle Group” thought that there was no need for it at the time of the Cloudera security engagement.

# Move from Embedded Database to Production-ready MySQL Database

The Cloudera Manager Server and the server's Activity Monitor, Service Monitor, Report Manager, and Host Monitor all require databases, as does Cloudera Navigator and the Hive Metastore. Cloudera Manager does support deploying different types of databases in a single environment, but doing so may create unexpected complications. Cloudera recommends choosing one of the three database providers to use for all of the Cloudera Manager databases.

Currently, we recommend using MySQL as an external database for scalability in a production environment. All of the primary services at The Brattle Group have been moved to MySQL databases. The steps to move from the PostGreSQL database to MySQL are documented here:

1. Stop all cluster services including the Cloudera Manager monitoring services and Agents on all hosts, using the Cloudera Manager UI.

2. Invoke the API call to /api/v10/cm/deployment. Example that directs the resulting output to a file for later use:

*$ curl -v -u admin:password http://your\_cloudera\_manager\_host:7180/api/v4/cm/deployment > path/to/file*

Note: It is critical this file be retained and protected as it will be needed to re-import the functional configuration into the new database.

3. Take a fail-safe database backup from the embedded postgres database or make a tarball (or other copy) of the /var/lib/cloudera-scm-server-db/ directory to a safe location. Either will accomplish the needed outcome, just pick the most convenient for your environment. And backup contents of the directory /etc/cloudera-scm-server/\*

4. On the server where the Cloudera Manager embedded database is running:

*$ service cloudera-scm-server stop*

*$ service cloudera-scm-server-db stop*

*$ chkconfig cloudera-scm-server-db off*

5. If not already done, ensure the MYSQL JDBC connector is installed and in the correct location [1] see "[Installing the MySQL JDBC Connector](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_ig_mysql.html?scroll=cmig_topic_5_6#cmig_topic_5_5_3_unique_1)".

6. Install, configure, and start the databases following the published instructions for MySQL detailed [here](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_ig_mysql.html?scroll=cmig_topic_5_6#cmig_topic_5_5_1_unique_1).

* Replace the default ‘my.cnf’ file with the currently recommended configuration.
* Create databases for SCM and any other services that you plan to move. The other services can be moved by backing up the existing databases, creating the new one as listed in the [install document](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_ig_mysql.html?scroll=cmig_topic_5_6#concept_dsg_3mq_bl_unique_1), and restoring the backed up file.

7. When complete, start the Cloudera Manager server service:

*$ service cloudera-scm-server start*

8. Log in to the Cloudera Manager web user interface and upload the enterprise license key.

9. After license upload do the following:

*$ service cloudera-scm-server restart*

10. From the command line of the node where the configuration export (from step 2) was performed, issue:

$ curl --upload-file /path/to/file -u youruser:yourpassword http://your\_cloudera\_manager\_host:7180/api/v10/cm/deployment?deleteCurrentDeployment=true

11. Log in to Cloudera Manager and verify configuration options are now all present.

12. To avoid later contention: yum erase cloudera-manager-server-db

Note: Ensure the file /etc/cloudera-scm-server/db.properties reflects the detail for the newly-established database and not the previous embedded postgres database.

13. Update configuration for all other services that have been moved via their Service Configurations, listed under ‘Database’.

As noted above, this has all been completed at The Brattle Group.

# MySQL Installation and Replication setup

Following are the steps followed to enable MySQL replication on the Brattle Group cluster:

1. Install and prepare the MySQL server with the supplied ‘my.cnf’ configuration file, install the JDBC MySQL driver and run the secure installation script as listed [here](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_ig_mysql.html?scroll=cmig_topic_5_6#cmig_topic_5_5_2_unique_1).
2. Stop all cluster services including the Cloudera Manager monitoring services and Agents on all hosts, using the Cloudera Manager UI.
3. Stop MySQL on both the origin host and replicating host using the following command:

*“service mysqld stop”*

1. Add the following to the ‘my.cnf’ files on each host using 1 as the ID for the origin host and 2 for the replication host:

*log-bin=mysql-bin     #Added for MySQL Replication*

*server-id    = 1     #Note: this should be 2 on the slave node*

1. Start MySQL on both hosts using:

*“service mysqld start”*

1. Log into the Origin/Master host MySQL instance and run the following commands replacing any fields in brackets with your company-specific information:

*[$ mysql]# mysql -u root -p*

*mysql> GRANT REPLICATION SLAVE ON \*.\* TO 'root’@‘<[f](http://hadoopadmin2.evariant.com/)qdn>' IDENTIFIED BY ‘<password>';*

*mysql> SET GLOBAL binlog\_format = 'ROW';*

*mysql> FLUSH PRIVILEGES;*

*mysql> FLUSH TABLES WITH READ LOCK;*

1. Press ‘Ctrl+Z’ to put this process in the background and log back in as a new session to maintain the READ LOCK and run:

*[$ mysql]# mysql -u root -p*

*mysql> SHOW MASTER STATUS; (take note of log file and position)*

1. Log into the MySQL instance on the Replication/Slave host and run the following commands replacing any fields in brackets with your company-specific information and including the log file and position from the previous step.

*CHANGE MASTER TO*

*MASTER\_HOST=‘*[*<*](http://hadoopadmin1.evariant.com/)*fqdn>',*

*MASTER\_USER=‘<user>r',*

*MASTER\_PASSWORD=‘<password>',*

*MASTER\_LOG\_FILE='<log file>',*

*MASTER\_LOG\_POS=<log position>;*

*START SLAVE;*

1. If there was existing data in the Origin/Master database, it will have to be exported and imported to the replication as only new data will be synced.
2. To remove the read lock, restart the MySQL server on the Origin/Master node with

*‘Service mysqld restart’*

# High Availability for NameNode services and YARN

## HDFS NameNode High Availability

In earlier releases, the NameNode was a single point of failure (SPOF) in an HDFS cluster. Each cluster had a single NameNode, and if that machine or process became unavailable, the cluster as a whole would be unavailable until the NameNode was either restarted or brought up on a separate machine. The Secondary NameNode did not provide failover capability.

This reduced the total availability of the HDFS cluster in two major ways:

1. In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.
2. Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

The HDFS HA feature addresses the above problems by providing the option of running two NameNodes in the same cluster, in an Active/Passive configuration. These are referred to as the **Active** NameNode and the **Standby** NameNode. Unlike the Secondary NameNode, the Standby NameNode is hot standby, allowing a fast failover to a new NameNode in the case that a machine crashes, or a graceful administrator-initiated failover for the purpose of planned maintenance. You cannot have more than two NameNodes.

In a typical HA cluster, two separate machines are configured as NameNodes. At any point in time, one of the NameNodes is in an Active state, and the other is in a Standby state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.

In order for the Standby node to keep its state synchronized with the Active node in this implementation, both nodes communicate with a group of separate daemons called JournalNodes. When any namespace modification is performed by the Active node, it durably logs a record of the modification to a majority of these JournalNodes. The Standby node is capable of reading the edits from the JournalNodes, and is constantly watching them for changes to the edit log. As the Standby Node sees the edits, it applies them to its own namespace. In the event of a failover, the Standby will ensure that it has read all of the edits from the JournalNodes before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.

In order to provide a fast failover, it is also necessary that the Standby node has up-to-date information regarding the location of blocks in the cluster. In order to achieve this, the DataNodes are configured with the location of both NameNodes, and they send block location information and heartbeats to both.

It is vital for the correct operation of an HA cluster that only one of the NameNodes be active at a time. Otherwise, the namespace state would quickly diverge between the two, risking data loss or other incorrect results. In order to ensure this property and prevent the so-called "split-brain scenario," the JournalNodes will only ever allow a single NameNode to be a writer at a time. During a failover, the NameNode which is to become active will simply take over the role of writing to the JournalNodes, which will effectively prevent the other NameNode from continuing in the Active state, allowing the new Active NameNode to safely proceed with failover.

Directions for configuring HDFS High Availability can be found [here](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_hag_hdfs_ha_enabling.html#cmug_topic_5_12_unique_1).

## YARN High Availability

The YARN ResourceManager (RM) is responsible for tracking the resources in a cluster and scheduling applications (for example, MapReduce jobs). Before CDH 5, the RM was a single point of failure in a YARN cluster. The RM high availability (HA) feature adds redundancy in the form of an Active/Standby RM pair to remove this single point of failure. Furthermore, upon failover from the Standby RM to the Active, the applications can resume from their last check-pointed state; for example, completed map tasks in a MapReduce job are not re-run on a subsequent attempt. This allows events such the following to be handled without any significant performance effect on running applications.:

* Unplanned events such as machine crashes
* Planned maintenance events such as software or hardware upgrades on the machine running the ResourceManager.

RM HA requires ZooKeeper and HDFS services to be running.

RM HA is implemented by means of an active-standby pair of RMs. On start-up, each RM is in the standby state: the process is started, but the state is not loaded. When transitioning to active, the RM loads the internal state from the designated state store and starts all the internal services. The stimulus to transition-to-active comes from either the administrator (through the CLI) or through the integrated failover controller when automatic failover is enabled. The subsections that follow provide more details about the components of RM HA.

Directions for enabling this through Cloudera Manager can be found [here](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_hag_rm_ha_config.html#concept_bxf_vcx_5m_unique_1).

# Additional Changes Made to Prepare Cluster for Production

## Upgrade from 5.3 to 5.4.2-1

The Brattle Group cluster has been updated to the newest versions of both Cloudera Manager and Cloudera Hadoop (CDH). To see incorporated upgrades, please see the release notes [here](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/rg_release_notes_cm.html).

## Reconfigure HDD’s on NameNodes to have separate spindles for Journal Node Service and ZooKeeper services.

Although we were only able to do this for 3 of the nodes required for these services, due to hardware constraints, we broke up the LVMs on two of the Master nodes in the cluster and reconfigured things such that, on these nodes, ZK and JN each have their own hard drives as recommended by best practices.

This greatly improves I/O for critical cluster services.

# Appendix

## Changes to the “import\_credentials.sh” script

Following is the script with the changes highlighted in red. This script can be found under the path “/usr/share/cmf/bin” on the CM node.

*#!/usr/bin/env bash*

*# Copyright (c) 2014 Cloudera, Inc. All rights reserved.*

*set -e*

*set -x*

*# Explicitly add RHEL5/6 and SLES11 locations to path*

*export PATH=/usr/kerberos/bin:/usr/kerberos/sbin:/usr/lib/mit/sbin:/usr/sbin:$PATH*

*KEYTAB\_OUT=$1*

*USER=$2*

*PASSWD=$3*

*KVNO=$4*

*# Determine if sleep is needed before echoing password.*

*# This is needed on Centos/RHEL 5 where ktutil doesn't*

*# accept password from stdin.*

*SLEEP=0*

*RHEL\_FILE=/etc/redhat-release*

*if [ -f $RHEL\_FILE ]; then*

*set +e # Ignore errors in grep*

*grep Tikanga $RHEL\_FILE*

*if [ $? -eq 0 ]; then*

*SLEEP=1*

*fi*

*if [ $SLEEP -eq 0 ]; then*

*grep 'CentOS release 5' $RHEL\_FILE*

*if [ $? -eq 0 ]; then*

*SLEEP=1*

*fi*

*fi*

*if [ $SLEEP -eq 0 ]; then*

*grep 'Scientific Linux release 5' $RHEL\_FILE*

*if [ $? -eq 0 ]; then*

*SLEEP=1*

*fi*

*fi*

*set -e*

*fi*

*if [ -z "$KRB5\_CONFIG" ]; then*

*echo "Using system default krb5.conf path."*

*else*

*echo "Using custom config path '$KRB5\_CONFIG', contents below:"*

*cat $KRB5\_CONFIG*

*fi*

*# Export password to keytab*

*IFS=' ' read -a ENC\_ARR <<< "$ENC\_TYPES"*

*{*

*for ENC in "${ENC\_ARR[@]}"*

*do*

*echo "addent -password -p $USER -k $KVNO -e $ENC"*

*if [ $SLEEP -eq 1 ]; then*

*sleep 1*

*fi*

*echo "$PASSWD"*

*done*

*echo "wkt $KEYTAB\_OUT"*

*} | ktutil*

*chmod 600 $KEYTAB\_OUT*

*# Do a kinit to validate that everything works*

*kinit -k -t $KEYTAB\_OUT $USER*

*# If this is not AD admin account, return from here*

*if [ "$AD\_ADMIN" != "true" ]; then*

*exit 0*

*fi*

*# With AD do a simple search to make sure everything works.*

*# Set properties needed for ldapsearch to work.*

*# Tell GSSAPI not to negotiate a security or privacy layer since*

*# AD doesn't support nested security or privacy layers*

*LDAP\_CONF=`mktemp /tmp/cm\_ldap.XXXXXXXX`*

*echo "TLS\_REQCERT never" >> $LDAP\_CONF*

*echo "sasl\_secprops minssf=0,maxssf=0" >> $LDAP\_CONF*

*export LDAPCONF=$LDAP\_CONF*

*AD\_SERVER=$(echo $AD\_SERVER | sed -e "s/ldaps/ldap/g")*

*AD\_SERVER=$(echo $AD\_SERVER | sed -e "s/636/389/g")*

*set +e # Allow failures to SASL so we can see if simple auth works*

*ldapsearch -LLL -H "$AD\_SERVER" -b "$DOMAIN" "userPrincipalName=$USER"*

*if [ $? -ne 0 ]; then*

*echo "ldapsearch did not work with SASL authentication. Trying with simple authentication"*

*ldapsearch -LLL -H "$AD\_SERVER" -b "$DOMAIN" -x -D $USER -w $PASSWD "userPrincipalName=$USER"*

*if [ $? -ne 0 ]; then*

*echo "Failed to do ldapsearch."*

*echo "Please make sure Active Directory configuration is correctly specified and LDAP over SSL is enabled."*

*exit 1*

*fi*

*# Simple authentication worked. Store the password in output file.*

*echo -n $PASSWD > $KEYTAB\_OUT*

*fi*

*set -e*

*rm -f $LDAP\_CONF*

## Changes to the “gen\_credentials\_ad.sh” script

Following is the script with the changes highlighted in red. This script can be found under the path “/usr/share/cmf/bin” on the CM node:

*#!/usr/bin/env bash*

*# Copyright (c) 2014 Cloudera, Inc. All rights reserved.*

*set -e*

*set -x*

*# Explicitly add RHEL5/6 and SLES11 locations to path*

*export PATH=/usr/kerberos/bin:/usr/kerberos/sbin:/usr/lib/mit/sbin:/usr/sbin:$PATH*

*KEYTAB\_OUT=$1*

*PRINC=$2*

*USER=$3*

*PASSWD=$4*

*DIST\_NAME="CN=$USER,$DOMAIN"*

*if [ -z "$KRB5\_CONFIG" ]; then*

*echo "Using system default krb5.conf path."*

*else*

*echo "Using custom config path '$KRB5\_CONFIG', contents below:"*

*cat $KRB5\_CONFIG*

*fi*

*SIMPLE\_PWD\_STR=""*

*if [ "$SIMPLE\_AUTH\_PASSWORD\_KEY" = "" ]; then*

*kinit -k -t $CMF\_KEYTAB\_FILE $CMF\_PRINCIPAL*

*AD\_SERVER=$(echo $AD\_SERVER | sed -e "s/ldaps/ldap/g")*

*AD\_SERVER=$(echo $AD\_SERVER | sed -e "s/636/389/g")*

*else*

*SIMPLE\_PWD\_STR="-x -D $CMF\_PRINCIPAL -w $SIMPLE\_AUTH\_PASSWORD\_KEY"*

*# Set properties needed for ldapmodify to work.*

*# Tell GSSAPI not to negotiate a security or privacy layer since*

*# AD doesn't support nested security or privacy layers*

*LDAP\_CONF=`mktemp /tmp/cm\_ldap.XXXXXXXX`*

*echo "TLS\_REQCERT never" >> $LDAP\_CONF*

*echo "sasl\_secprops minssf=0,maxssf=0" >> $LDAP\_CONF*

*export LDAPCONF=$LDAP\_CONF*

*fi*

*# AD lets you create multiple accounts with same principal*

*# as long as distinguished name is unique. So we should check*

*# if the principal already exists and let user know about it.*

*PRINC\_SEARCH=`ldapsearch -LLL -H "$AD\_SERVER" -b "$DOMAIN" $SIMPLE\_PWD\_STR "userPrincipalName=$PRINC"`*

*set +e # Allow non-zero return from grep*

*# ldapsearch by default wraps its output to 78 chars. Though it provides an option*

*# to not wrap on newer versions, older ones don't have it. So we can't search for*

*# principal name in the output, but there will certainly be a line starting with*

*# userPrincipalName if a match is found. So we search for that instead.*

*echo $PRINC\_SEARCH | grep -q userPrincipalName*

*if [ $? -eq 0 ]; then*

*echo "$PRINC already exists in Active Directory. Please delete it before re-generating it from Cloudera Manager."*

*exit 1*

*fi*

*set -e*

*# Add account in AD*

*# servicePrincipalName is obtained from $PRINC by removing "@REALM" from the end.*

*# password needs to be specified in unicode using "iconv"*

*# account is set to never expire by setting accountExpires to 0*

*# userAccountControl is 66048 which is obtained by adding*

*# 512 (for "normal" account) and 65536 (password never expires)*

*ldapmodify -H "$AD\_SERVER" $SIMPLE\_PWD\_STR <<-%EOF*

*dn: $DIST\_NAME*

*changetype: add*

*objectClass: top*

*objectClass: person*

*objectClass: organizationalPerson*

*objectClass: user*

*distinguishedName: $DIST\_NAME*

*sAMAccountName: $USER*

*servicePrincipalName: $(echo $PRINC | sed -e "s/\@$CMF\_REALM//g")*

*userPrincipalName: $PRINC*

*unicodePwd:: `echo -n "\"$PASSWD\"" | iconv -f UTF8 -t UTF16LE| base64 -w 0`*

*accountExpires: 0*

*userAccountControl: 66048*

*%EOF*

*rm -f $LDAP\_CONF*

*# Determine if sleep is needed before echoing password.*

*# This is needed on Centos/RHEL 5 where ktutil doesn't*

*# accept password from stdin.*

*SLEEP=0*

*RHEL\_FILE=/etc/redhat-release*

*if [ -f $RHEL\_FILE ]; then*

*set +e # Ignore errors in grep*

*grep Tikanga $RHEL\_FILE*

*if [ $? -eq 0 ]; then*

*SLEEP=1*

*fi*

*if [ $SLEEP -eq 0 ]; then*

*grep 'CentOS release 5' $RHEL\_FILE*

*if [ $? -eq 0 ]; then*

*SLEEP=1*

*fi*

*fi*

*if [ $SLEEP -eq 0 ]; then*

*grep 'Scientific Linux release 5' $RHEL\_FILE*

*if [ $? -eq 0 ]; then*

*SLEEP=1*

*fi*

*fi*

*set -e*

*fi*

*# Export password to keytab*

*IFS=' ' read -a ENC\_ARR <<< "$ENC\_TYPES"*

*{*

*for ENC in "${ENC\_ARR[@]}"*

*do*

*echo "addent -password -p $PRINC -k 1 -e $ENC"*

*if [ $SLEEP -eq 1 ]; then*

*sleep 1*

*fi*

*echo "$PASSWD"*

*done*

*echo "wkt $KEYTAB\_OUT"*

*} | ktutil*

*chmod 600 $KEYTAB\_OUT*

## smb.conf for cmbhdpdn008.brattle.net

*[global]  
netbios name = cmbhdpdn008  
workgroup = CAMBRIDGE  
security = ADS  
realm = BRATTLE.NET  
encrypt passwords = yes  
idmap config \*:backend = tdb  
idmap config \*:range = 70001-80000  
idmap config CAMBRIDGE:backend = ad  
idmap config CAMBRIDGE:schema\_mode = rfc2307  
idmap config CAMBRIDGE:range = 500-40000  
winbind nss info = rfc2307  
winbind trusted domains only = no  
winbind use default domain = yes  
winbind enum users = yes  
winbind enum groups = yes*

*[demoshare]  
path = /srv/samba/test  
read only = no*

## krb5.conf

*[libdefaults]  
default\_realm = BRATTLE.NET  
dns\_lookup\_kdc = false  
dns\_lookup\_realm = false  
ticket\_lifetime = 36000  
renew\_lifetime = 604800  
forwardable = true  
default\_tgs\_enctypes = rc4-hmac  
default\_tkt\_enctypes = rc4-hmac  
permitted\_enctypes = rc4-hmac  
udp\_preference\_limit = 1*

*kdc\_timeout = 3000  
[realms]  
BRATTLE.NET = {  
kdc = cmbdc1.brattle.net  
kdc = cmbdc2.brattle.net  
}*

## sssd.conf

*[sssd]  
config\_file\_version = 2  
services = nss, pam  
domains = brattle.net  
#override\_space = \_*

*debug\_level = 0*

*[nss]  
override\_homedir = /home/%u  
default\_shell = /bin/bash  
reconnection\_retries = 3*

*[pam]  
reconnection\_retries = 3*

*[domain/brattle.net]*

*debug\_level = 0  
enumerate = true  
id\_provider = ad  
chpass\_provider = ad  
auth\_provider = ad*

*access\_provider = simple*

*simple\_allow\_groups = prod\_ssh\_login\_edge  
ad\_server = cmbdc1.brattle.net  
ad\_backup\_server = cmbdc2.brattle.net  
ldap\_schema = ad*

*ldap\_user\_principal = nosuchattr*

*ldap\_id\_mapping = false (causing issue on cmbhdpdn008)*

*ldap\_force\_upper\_case\_realm = true*

*case\_sensitive = false*

*krb5\_realm = BRATTLE.NET*

*ldap\_access\_order = filter,expire*

*ldap\_account\_expire\_policy = ad*

*cache\_credentials = true*

*account\_cache\_expiration = 15*

*enum\_cache\_timeout = 120*

*entry\_cache\_nowait\_percentage = 50*

*entry\_cache\_nowait\_timeout = 28800*

*ldap\_group\_search\_base = ou=Groups,ou=Linux,dc=brattle,dc=net*