Paper 1168-2017

Using Shared Accounts in Kerberized Hadoop Clusters with SAS®: How Can I Do That?

Michael Shealy, Cached Consulting, LLC

ABSTRACT

Using shared accounts to access third-party database servers is a common architecture in SAS® environments. SAS software can support seamless user access to shared accounts in databases such as Oracle and MySQL, via group definitions and outbound authentication domains in Metadata. However, the configurations necessary to leverage shared accounts in Kerberized Hadoop clusters are more complicated. Kerberos tickets must often be generated and maintained in order to simply access the Hadoop environment, and those tickets must allow access as the shared account instead of an individual user's account. In all cases, key prerequisites and configurations must be put into place in order for seamless Hadoop access to function with the shared account. Methods for implementing these arrangements in SAS environments can be non-intuitive.

INTRODUCTION

This paper will start by outlining general architectures of shared accounts in third-party database environments. It will then present several methods of managing remote access to shared accounts in Kerberized Hadoop environments using SAS, including specific implementation details, code samples and security implications.

Note that this paper is not intended to provide a comprehensive description of every available implementation option. Instead, the author intends to show the general complexities involved in seamlessly accessing remote Kerberized Hadoop instances with shared accounts, a set of common use-cases, and a group of options which may be used directly or in modified form to fit various real-world scenarios.

This paper assumes a high-level familiarity with Kerberos, particularly how Kerberos uses tickets to authenticate users and allow access to services. A general Kerberos overview can be found in the Recommended Readings section at the end of the paper. Example code and configurations for this paper were developed on a SAS 9.4M3 system running on Redhat Enterprise Linux 6.

SHARED ACCOUNT DATABASE ACCESS

A common architecture in multi-tier application environments is to perform any required user authentication at a web/application layer, while providing access to the program's data via a common shared account at the database tier. Figure 1 is a simple example of such an architecture:

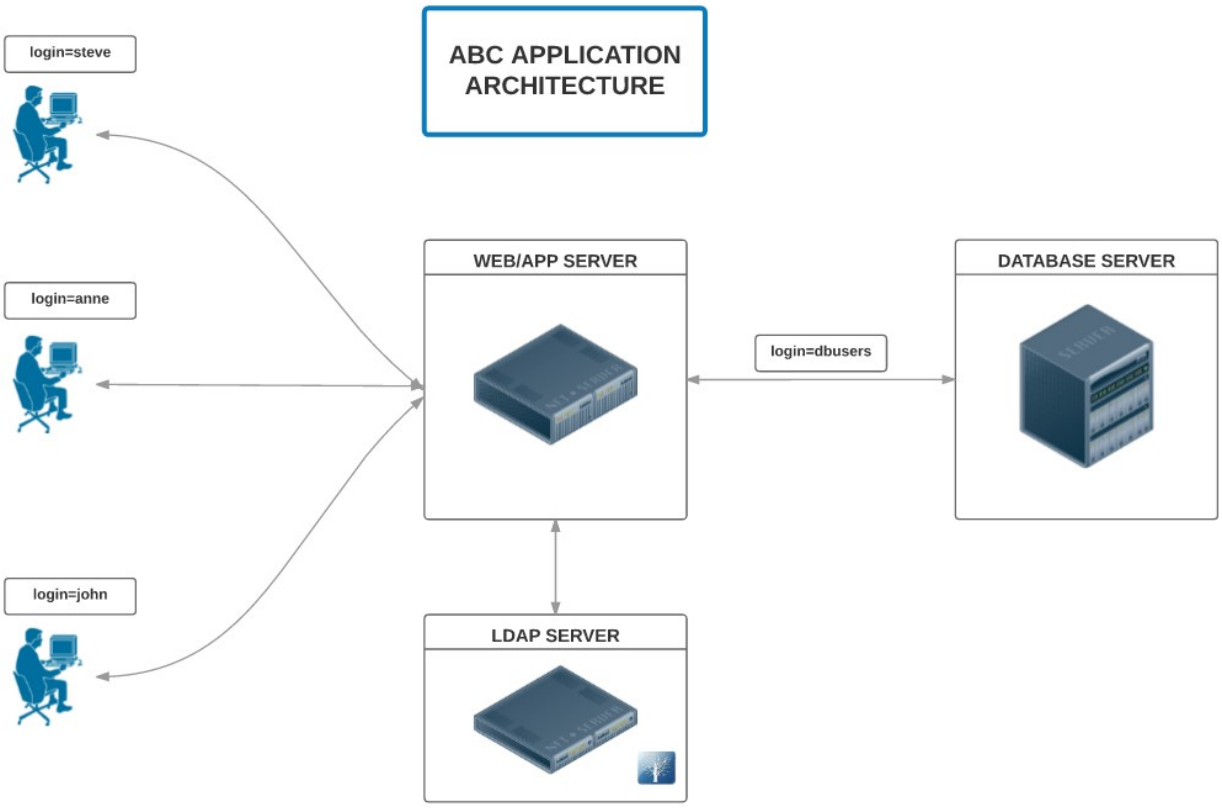


Figure 1. Generic Application Architecture With A Shared Database Account

Steve, Anne and John all authenticate themselves at the Web/Application Tier against a list of valid users maintained in an LDAP repository. Once they have been authorized to access the application, a common shared account ("dbusers") is then used by the Application Tier for all communication to the Database Tier. This architecture significantly reduces administrative complexity at the Database Tier, by removing the need to provision users and manage security requirements inside the database for each application user.

Many SAS environments employ a similar architecture when accessing third-party databases via SAS/ACCESS products. A common example would be Figure 2, where individual SAS Enterprise Guide or SAS Studio users first authenticate against a Metadata server with LDAP integration (either Host-based or Direct LDAP), and then connect to a Workspace server on the same host:

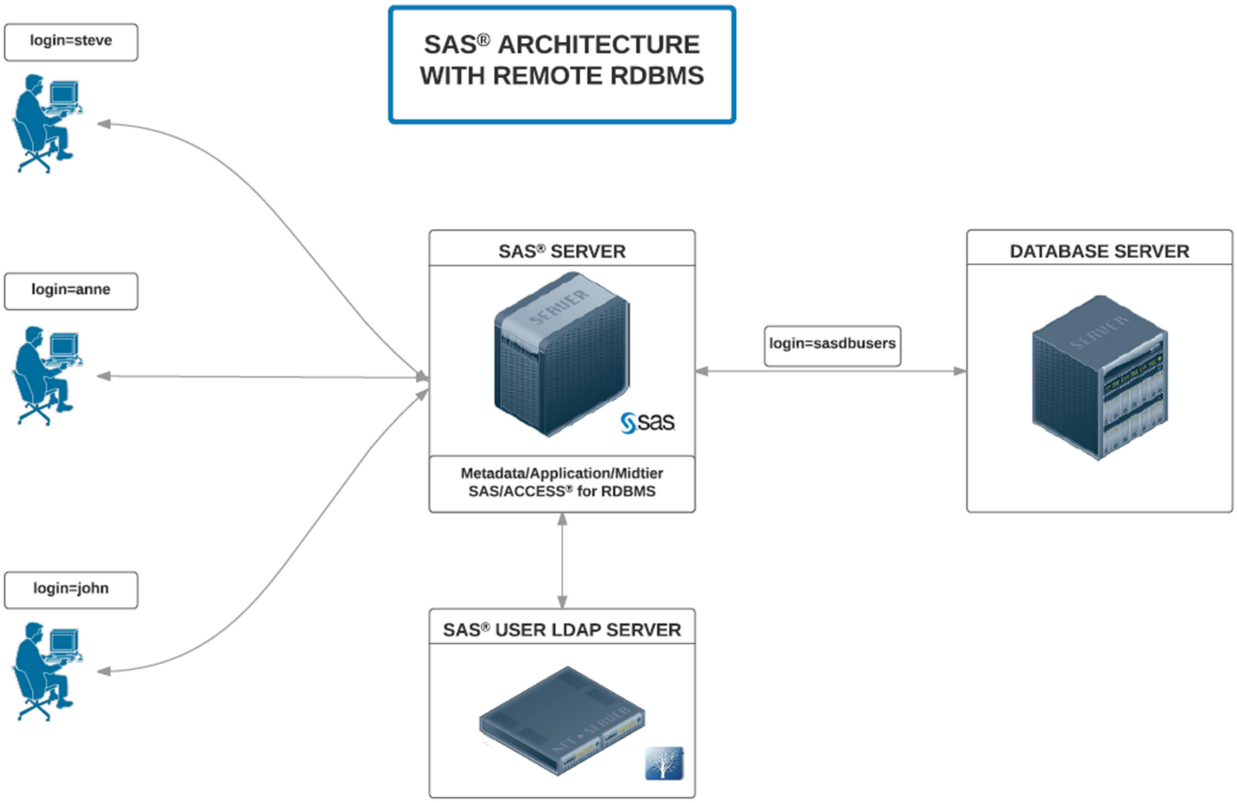


Figure 2. SAS® Environment With Third-Party RDBMS Access Using A Shared Account

The local SAS Administrator can then modify SAS Metadata to allow users seamless access to the database server over a shared account, using user groups and authentication domains. This procedure is outlined in the "How to Store Passwords for a Third-Party Server" section of the *SAS® 9.4 Intelligence Platform: Security Administration Guide*. As in the earlier example, the administrative overhead of managing each SAS user's individual identity inside the database is then able to be set aside.

Seamless access to a Kerberized Hadoop instance using a shared-account is not as straightforward, however, as we shall now see.

KERBEROS AUTHENTICATION METHODS

There are two common methods for authenticating to a Kerberized Hadoop instance in SAS environments. The authentication method chosen will affect both the prerequisites and the configuration of the deployment:

- **Native Kerberos Authentication:** In a Native Kerberos authentication architecture, a Kerberos Ticket Granting Ticket (TGT) for the shared account (*not* the individual SAS user) must be acquired on each SAS host which connects to Hadoop, which is then used to acquire Kerberos Service Tickets for access to the Hadoop instance. As SAS does not directly acquire or manage Kerberos tickets itself, it is generally left to system administrators (both SAS and OS) to implement a configuration where the necessary Kerberos tickets are available when access to the Kerberized Hadoop instance is needed. This paper will offer options for managing this availability in later sections.
- **Perimeter Security Authentication via User/Password over HTTPS:** Apache Knox Gateway provides a common REST API interface for individual Hadoop components, such as Hive, HBase and HDFS. It offers this service by implementing a reverse-proxy access architecture. Authentication to the Hadoop cluster through Knox involves sending an account username and password over a secured http session (https).

SEAMLESS DATABASE CONNECTIVITY PREREQUISITES

Each authentication method has a strict set of prerequisites, if seamless database access is to be successfully implemented:

1. In both Native Kerberos and Perimeter Security authentication architectures, a complete and accurate set of jar files must be acquired from the remote Hadoop cluster. These files must be placed on each SAS host which communicates with the cluster, in the location referred to by the environment variable SAS_HADOOP_JAR_PATH. The *SAS 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS* guide has detailed instructions for completing this task for each supported version of Hadoop**.**
2. In both Native Kerberos and Perimeter Security authentication architectures, a complete and accurate set of configuration files must be acquired from the remote Hadoop cluster. These files must be placed on each SAS host which communicates with the cluster, in the location referred to by the environment variable SAS_HADOOP_CONFIG_PATH. The *SAS 9.4 Hadoop Configuration Guide for Base SAS and SAS/ACCESS* guide has detailed instructions for completing this task for each supported version of Hadoop.
3. In Native Kerberos authentication architectures, each SAS host must be properly configured as part of the same Kerberos Realm as the Hadoop datastore, or be a member of a Kerberos realm with a valid trust-relationship to the Hadoop instance's realm. This is necessary to both acquire the shared account Kerberos TGT, and have it be valid for supporting access to the Hadoop cluster. This configuration task is generally handled by an environment's OS and Kerberos System Administrators. Further information on this topic can be found in the "Hadoop with Kerberos – Architecture Considerations" document from SAS.
4. In Native Kerberos authentication architectures, a keytab must be generated for the shared database account, and placed in a location on each SAS host accessible by all operating system accounts which will run processes that connect to the Hadoop datastore. In the case of Stored Process and Pooled Workspace servers, this is often the sassrv account. For Workspace Servers, these are often the IDs of the individual users of the environment. A keytab is a binary file containing Kerberos users ("principals") and encrypted keys, which can be used to authenticate in a Kerberos environment without needing to enter a password.
5. In Perimeter Security architectures using Apache Knox, SAS Hot Fix W11006 (or higher) must be applied. This Hot Fix, originally released in October 2015, brought support to SAS 9.4M3 for communication with Kerberized Hadoop instances which use Knox for perimeter-based security.
6. In many environments, the encryption requirements of the Kerberos implementation mandate the use of the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files. Updated policy files for most operating systems are available at Oracle's website, though AIX implementations will need to acquire their policy files from IBM. Note that the update needs to be applied to the Java distribution used by SAS, generally located at !SAS_HOME/SASPrivateJavaRuntimeEnvironment.

PREREQUISITE VALIDATION

Once the Connectivity Prerequisites above have been put into place, a validation of a basic LIBNAME statement should be performed against the remote Hadoop cluster. The environment *must* be able to pass this test, before the seamless database access options which follow can succeed.

- **Native Kerberos Authentication Architecture**: A user must first be able to acquire the initial Ticket Granting Ticket *as the shared account user*, via the shared account keytab generated in the prerequisite steps. To validate this in a Linux environment, the user can log into the SAS host (or acquire access as the sassrv account, if planning to use Stored Process or Pooled Workspace servers) and execute a command such as:

```
$ export KRB5CCNAME=/tmp/krb5cc_sasdbusers_`id -u`
$ /usr/bin/kinit -k -t /opt/app/sas/9.4/keytabs/sasdbusers.keytab \
  sasdbusers@MYREALM.COM
```

To validate that the ticket was acquired correctly, execute a “klist”. This should show a valid ticket in the user’s cache, such as:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_sasdbusers_5493
Default principal: sasdbusers@MYREALM.COM

Valid starting     Expires            Service principal
07/23/16 07:50:57  07/24/16 07:50:57  krbtgt/MYREALM.COM@MYREALM.COM
      renew until 07/23/16 07:50:57
```

Be sure that the Default Principal is the shared database account, in this case sasdbusers@MYREALM.COM.

After acquiring the Kerberos TGT, the user should then be able to manually create a library reference to the remote Hadoop datastore. With a valid set of configuration and jar files collected from the remote Hadoop cluster, and the SAS_HADOOP_CONFIG_PATH and SAS_HADOOP_JAR_PATH set properly, the command after executing “!SASROOT/sas -nodms” may be as simple as:

```
1? LIBNAME myhdp HADOOP server='hadoopsvr.mydomain.com';

NOTE: Libref MYHDP was successfully assigned as follows:
    Engine:        HADOOP
    Physical Name: jdbc:hive2://hadoopsvr.mydomain.com:10000/default
```

If additional options are required to properly establish the connection, the comprehensive list of parameters for the Hadoop LIBNAME statement is available in the *SAS/ACCESS(R) 9.4 for Relational Databases: Reference*. Note that a username and password is not needed in the LIBNAME statement, as the Kerberos TGT acquired with the kinit above is being used to authenticate to the Hadoop cluster.

Notice that we placed the storage location (“cache”) of the shared account Kerberos TGT in a custom location by manually setting the KRB5CCNAME environment variable. This was to ensure that our LIBNAME connection to the Hadoop instance used the shared account credentials for authenticating against the cluster, instead of any other Kerberos credentials which may exist inside the default Kerberos ticket cache for the individual user. Such credential collisions may occur in environments where Kerberos is also used to authenticate individual users into the SAS hosts.

- **Perimeter Security Architecture**: Accessing the Hadoop cluster in a Perimeter Security architecture does not require a Kerberos ticket for the shared account on the SAS host, as the authentication occurs via the delivery of the shared account username and password over HTTPS. Therefore, after acquiring the correct Hadoop jar and config files and applying Hot Fix W11006 (or higher), a LIBNAME statement similar to the following can be executed:

```
LIBNAME myhdp HADOOP
server="hadoopsvr.mydomain.com"
schema='default'
user='sasdbusers'
pwd='abc123'
uri="jdbc:hive2://hadoopsvr.mydomain.com:8443/default;ssl=true; \
transportMode=http;httpPath=gateway/default/hive; \
sslTrustStore=/opt/app/sas/9.4/knox/gateway.jks; \
trustStorePassword=MyStorePassword";
```

This LIBNAME statement will send the shared account username and password over an HTTPS connection to the Knox gateway detailed in the uri statement, and attempt to authenticate.

Depending on your Hadoop environment's SSL configuration at the Knox gateway, the sslTrustStore and trustStorePassword may not be necessary. Your local Hadoop administrator should be able to provide guidance in this area.

Now, on to configuring seamless database access!

OPTION 1: PERIMETER SECURITY USING KNOX

In most cases, using Knox to manage Shared Account access is the easiest implementation, provided Knox is implemented in your Hadoop ecosystem. As shown in Figure 3, the Knox interface is a REST API allowing access to Hadoop services:

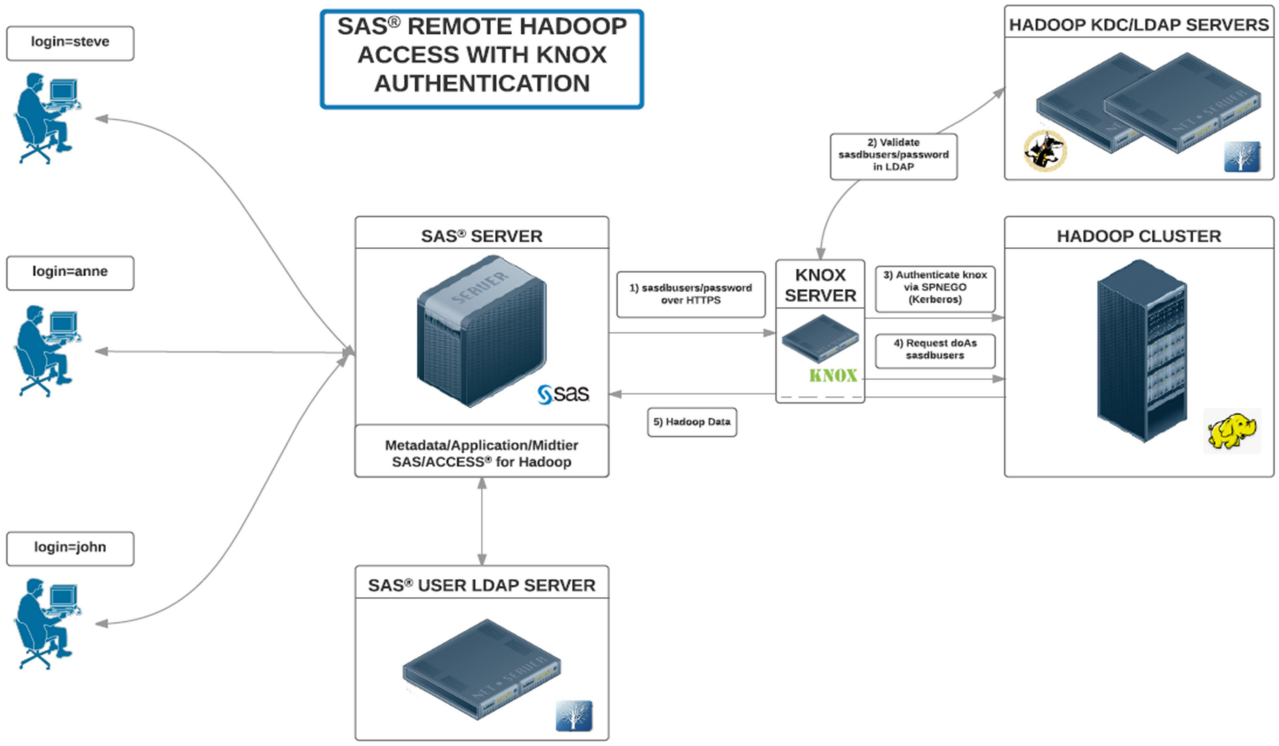


Figure 3. SAS® Environment With Remote Hadoop Access, Knox Architecture

To provide access to the remote Hadoop datastore inside a specific SAS application domain, for example, our $APPSERVER_ROOT's appserver_autoexec_usermods.sas file could simply contain a statement similar to our Prerequisite Validation:

```
LIBNAME myhdp HADOOP
server="hadoopsvr.mydomain.com"
schema='default'
user='sasdbusers'
pwd="{SAS002}93BBD64A349A24B02A6751FE050DD80E33E177993623D6650B3276D4"
uri="jdbc:hive2://hadoopsvr.mydomain.com:8443/default;ssl=true; \
transportMode=http;httpPath=gateway/default/hive; \
sslTrustStore=/opt/app/sas/9.4/knox/gateway.jks; \
trustStorePassword=MyStorePassword";
```

Because the LIBNAME statement will be placed in a readable file on the filesystem in this case, and contain the password to the shared account, the password should always be encoded with the "PROC PWENCODE" procedure. Passwords in cleartext pose a high security risk. To encode the shared

account password, a statement similar to the following can be executed in a SAS session, and the output pasted into the “pwd” line of the LIBNAME statement above:

```
1? PROC PWENCODE in='abc123';
2? run;

{SAS002}93BBD64A349A24B02A6751FE050DD80E33E177993623D6650B3276D4
```

Do note that communication with the Hadoop cluster in a Knox architecture is over HTTPS, which may have a different performance profile than accessing the Hadoop cluster directly over a socket connection in a Native Kerberos implementation. If both Perimeter Security and Native Kerberos authentication options are available to your project, you may want to include a performance evaluation of both configurations before choosing a final implementation architecture.

OPTION 2: NATIVE KERBEROS AUTHENTICATION FOR DAILY USERS

Using Native Kerberos authentication to seamlessly connect to a Hadoop instance presents an additional challenge: managing a Kerberos TGT for the shared database account. In our Prerequisite Validation step, we acquired the TGT by logging into our Linux host and manually running “kinit” on the command line. However, we are looking for a way to provide access to our Hadoop cluster’s data seamlessly, which means we would like for our users to not need to execute time-consuming manual steps.

Our Figure 4 shows an example SAS environment of SAS Studio and Enterprise Guide users:

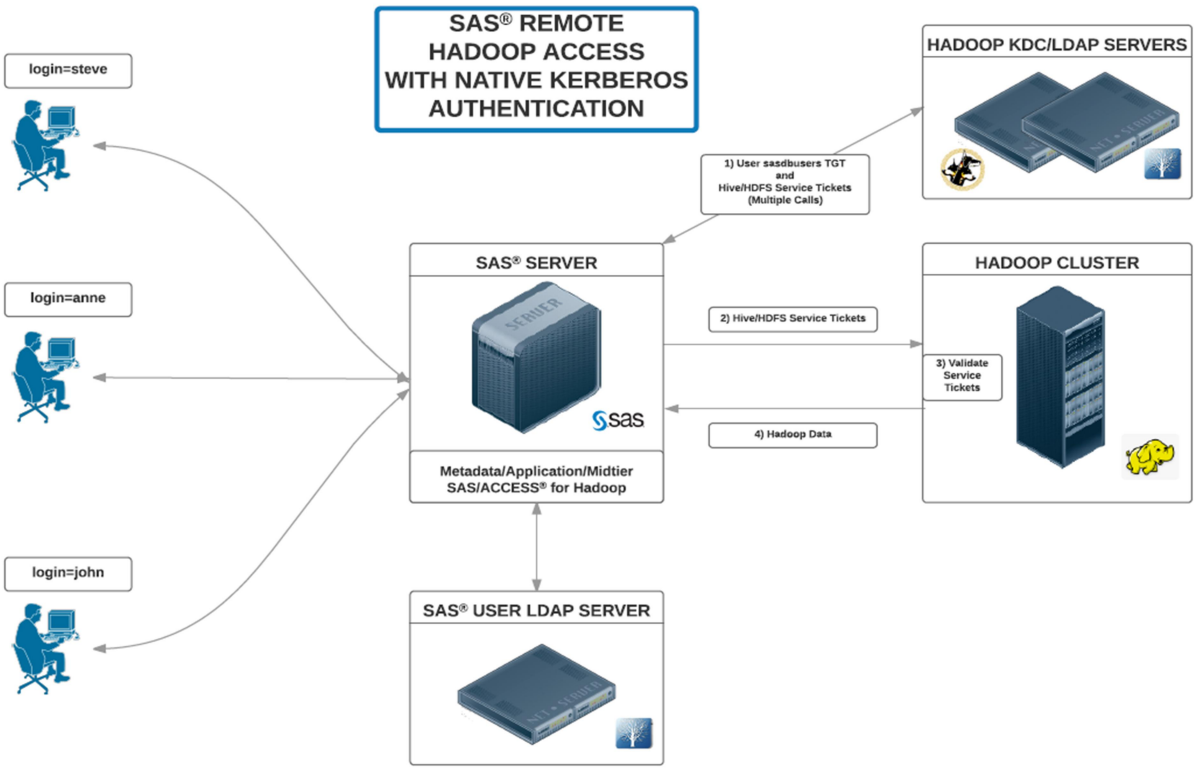


Figure 4. SAS® Environment With Remote Hadoop Access, Native Kerberos Architecture

If our users only work business hours and log off in the evening, we can accomplish seamless access by modifying a pair of files. In $APPSERVER_ROOT/WorkspaceServer/WorkspaceServer_usermods.sh, we can add lines such as the following to acquire a Kerberos TGT for the shared account:

```
## Each user should maintain their own private Kerberos credentials cache
## Therefore, the use of uid

uid=$(id -u)
export KRB5CCNAME=/tmp/krb5cc_sasdbusers_${uid}
KEYTAB_LOC=/opt/app/sas/9.4/keytabs/sasdbusers.keytab

## Turn on tracing, only as needed
#export KRB5_TRACE=/tmp/krb_trace_${uid}.log

/usr/bin/kinit -k -t $KEYTAB_LOC -c $KRB5CCNAME sasdbusers@MYREALM.COM
```

This code performs the same steps we executed during our Prerequisite Validation step, but this time it will execute the code for the user automatically at the initiation of a Workspace Server instance.

Then, in $APPSERVER_ROOT/WorkspaceServer/autoexec_usermods.sas, we can add a line such as the following to stand up the library reference to our Hadoop instance:

```
LIBNAME myhdp HADOOP server='hadoopsvr.mydomain.com';
```

When a user then logs into SAS Studio or Enterprise Guide, they should now see the MYHDP library available for access, as in Display 1:

Display 1. SAS® Studio With Kerberized Hadoop Library

Instead of placing the LIBNAME statement in the autoexec_usermods.sas file, the SAS Administrator could instead use the instructions in the "Establishing Connectivity To Hadoop" section of the *SAS 9.4 Intelligence Platform: Data Administration Guide* to register a Hadoop Server and a Hadoop Library in Metadata. For a Kerberized Hadoop instance, it is important to place the Hive Kerberos Principal found in the Hadoop cluster's hive-site.xml (one of the Hadoop configuration files collected during the prerequisite stage) on the Options->AdvancedOptions->Connection tab, and to choose Default Login = "None" under Connection Details.

Notice that by using the keytab file in the kinit statement, a user is not required to input the password to the shared account at any point. From a security point of view, this means that any user with access to the keytab file can acquire access to the data stored in the Hadoop cluster under the shared account ID. The keytab file should therefore be carefully protected using standard OS group permissions or user/group-based filesystem ACLs, to allow keytab access only to those users who specifically require access to the Hadoop datastore as the shared account.

OPTION 3: NATIVE KERBEROS AUTHENTICATION FOR ENTERPRISE MINER OR LONG-RUNNING PROCESSES

While our Option 2 will work well in an environment of business-hour SAS Studio and Enterprise Guide users, there are other cases where this approach will have weaknesses:

- If a SAS process remains alive for longer than 10-12 hours, the Kerberos TGT may expire, and cause our session to lose access to our Hadoop datastore. This may occur if an Enterprise Guide user leaves his session up overnight, or during a long-running batch job, or with Stored Process and Pooled Workspace server processes. (The 10-12 hour timeframe is a common default setting for TGT expiration in Kerberos environments. The specific setting in your environment can be found by checking the "Expires" column in the output of the klist command, after acquiring the shared account TGT with kinit.)
- When running Enterprise Miner, which starts many Workspace Servers to perform specific subsets of work, our Option 2 code will acquire a Kerberos TGT at each Workspace Server launch, even if a valid TGT already exists in the user's custom ticket cache…leading to unnecessary load on the Kerberos infrastructure.

To improve our implementation for these cases, we can make use of the freeware "kstart" utility, available online and in the RHEL6 EPEL yum repository. Kstart is a modified version of kinit, which can automatically manage the renewal of Kerberos tickets.

After installing kstart on our SAS hosts, we can have our Kerberos tickets auto-renewed for long-running processes such as Stored Process and Pooled Workspace Servers by replacing our Option 2 code with the following in our $APPSERVER_ROOT/appservercontext_env_usermods.sh file:

```
## Each user should maintain their own private Kerberos credentials cache
## Therefore, the use of uid

uid=$(id -u)
export KRB5CCNAME=/tmp/krb5cc_sasdbusers_${uid}
K5START_PIDFILE=/tmp/k5start_${uid}.pid
KEYTAB_LOC=/opt/app/sas/9.4/keytabs/sasdbusers.keytab
K5START_CMD="/usr/bin/k5start -b -p $K5START_PIDFILE -f $KEYTAB_LOC -l \
  10h -K 10 sasdbusers@MYREALM.COM"

## Turn on tracing, only as needed
#export KRB5_TRACE=/tmp/krb_trace_${uid}.log

## Check if we already have a Kerberos ticket,
## and if we're already running k5start.
## This will determine our actions later.

TGT_CHK=$(/usr/bin/klist -s -c /tmp/krb5cc_sasdbusers_${uid} \
  > /dev/null 2>&1; echo $?)

if [ -s $K5START_PIDFILE ]; then
  K5PROC_CHK=$(/bin/ps -p `cat $K5START_PIDFILE` | grep k5start \
    > /dev/null 2>&1; echo $?)
else
  K5PROC_CHK=1
fi

# We don't want to accidentally end up with multiple k5starts running for
# the same user, so let's be smart if we don't have a
# TGT but kstart seems to already be running.
```

```
if [ "$K5PROC_CHK" -eq 0 ]; then

  if [ "$TGT_CHK" -ne 0 ]; then

    # We appear to have a k5start process that isn't functioning
    # correctly, because we don’t have a  ticket. Kill it before
    # we start up another.
    /bin/kill -9 `cat $K5START_PIDFILE`
    $K5START_CMD

  fi

else

  $K5START_CMD

fi
```

This code causes the k5start process to check every 10 minutes (-K 10) whether the Kerberos TGT for the shared account is nearing expiration, and if it is, it will automatically renew the TGT. This behavior will allow our long-session Enterprise Guide users, long-running batch jobs, Stored Process servers, and Pooled Workspace servers to not lose connectivity to our Hadoop datastore.

In addition, the code starts one session of k5start at the initial launch of a process by a specific user, and then checks on launches of subsequent processes to see if a k5start and TGT already exist. If they do, the code will not attempt to start another k5start process or acquire a ticket. This behavior will help keep Enterprise Miner users from placing a large amount of load on the realm’s Kerberos infrastructure.

Then in our $APPSERVER_ROOT/appserver_autoexec_usermods.sas file, we can place our Hadoop LIBNAME statement:

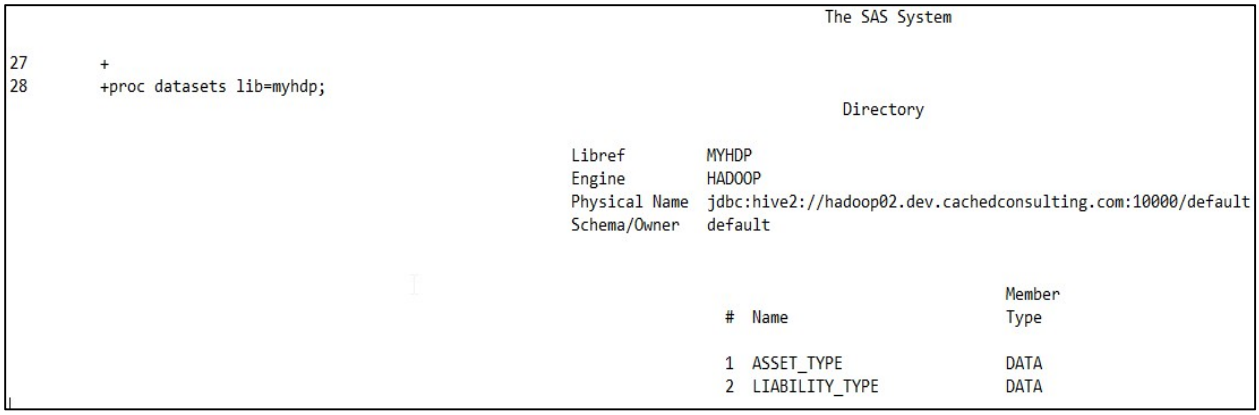
```
LIBNAME myhdp HADOOP server='hadoopsvr.mydomain.com';
```

As with Option 2, instead of placing the LIBNAME statement in the autoexec_usermods.sas file, the SAS Administrator could also use the instructions in the “Establishing Connectivity To Hadoop” section of the *SAS 9.4 Intelligence Platform: Data Administration Guide* to register a Hadoop Server and a Hadoop Library in Metadata. For a Kerberized Hadoop instance, it is important to place the Hive Kerberos Principal found in the Hadoop cluster’s hive-site.xml (one of the Hadoop configuration files collected during the prerequisite stage) on the Options->AdvancedOptions->Connection tab, and to choose Default Login = “None” under Connection Details.

After executing a Stored Process on our server with the code above deployed, we should see a k5start process running as our sassrv account, managing our Kerberos TGT:

```
$ ps -ef |grep k5start | grep sassrv
   sassrv   29893   1  0 14:57 ?   00:00:00 /usr/bin/k5start -b -p
   /tmp/k5start_502.pid -f /opt/app/sas/9.4/keytabs/sasdbusers.keytab -l
   10h -K 10 sasdbuserss@MYREALM.COM
```

And in the output of a stored process that simply runs the PROC DATASETS procedure, we should see that we have seamless access to our Hadoop datastore, as in Display 2:



Display 2. SAS® Stored Process Web Application Output

As with Option 2, notice that by using the keytab file in the kstart command, a user is not required to input the password to the shared account at any point. From a security point of view, this means that any user with access to the keytab file can acquire access to the data stored in the Hadoop cluster under the shared account ID. The keytab file should therefore be carefully protected using standard OS group permissions or user/group-based filesystem ACLs, to allow keytab access only to those users who specifically require access to the Hadoop datastore as the shared account.

CONCLUSION

Using shared accounts to seamlessly access third-party databases is common in SAS environments, but seamlessly accessing remote Hadoop datastores which are Kerberized can be tricky. Careful implementation of Kerberos and Hadoop-connectivity prerequisites is the key first step, followed by a validation that non-seamless baseline connectivity can be established to the Hadoop cluster. Once those steps are completed, the implementation of custom code into the SAS process instantiation scripts can automate the acquisition of Kerberos tickets and the creation of library references. Integration with helpful third-party tools such as kstart can be extremely valuable as well, in order to better support use-cases with long-running individual processes or large numbers of SAS processes per user.

REFERENCES

- SAS Institute Inc. 2015. *SAS® 9.4 Hadoop Configuration Guide for Base SAS® and SAS/ACCESS®, Second Edition*. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. *SAS/ACCESS® 9.4 for Relational Databases: Reference, Seventh Edition*. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. *SAS® 9.4 Intelligence Platform, Data Administration Guide, Fifth Edition*. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. 2013. *SAS® 9.4 Intelligence Platform: Security Administration Guide, Second Edition*. Cary, NC: SAS Institute Inc.
- Rogers, Stuart and Keefer, Tom. 2014. *Hadoop with Kerberos – Architecture Considerations*. Cary, NC: SAS Institute Inc.
- Minder, Kevin and McCay, Larry. 2014. “Securing Hadoop’s REST APIs Apache Knox Gateway.” Hadoop Summit 2014. Available at http://www.slideshare.net/Hadoop_Summit/th-130p211minder.

ACKNOWLEDGEMENTS

I would like to thank Stuart Rogers and Tom Keefer for their work in developing best practices and documentation in the area of SAS, Hadoop and Kerberos integrations. This paper is merely an extension

of their excellent efforts. I would also like to offer an additional special thanks to Stuart, for suggesting an evaluation of kstart as a way to manage the long-running-process ticket-timeout issue.

I would also like to thank Spencer, Don, and Rebecca Hayes for not only reviewing and offering valuable suggestions to this paper, but also for persuading me that a career move into SAS architecture and implementation would be both challenging and rewarding.

RECOMMENDED READINGS

- *Explain like I'm 5 Kerberos.* Available at http://www.roguelynn.com/words/explain-like-im-5-kerberos/
- *What is a keytab, and how do I use one?* Available at https://kb.iu.edu/d/aumh
- *Kstart General Overview.* Available at https://www.eyrie.org/~eagle/software/kstart/readme.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Michael Shealy
Cached Consulting, LLC
(678) 477-3759
michael.shealy@cachedconsulting.com
www.cachedconsulting.com