TIBCO Software Inc.
Global Headquarters
3307 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005
www.tibco.com

How to Configure Apache Kafka with Kerberos and Microsoft Active Directory for Authentication

This document provides information for configuring Apache Kafka for Kerberos and Microsoft Active Directory for Authentication

Version 1.0 October 2021 Initial Document

For more information, please contact:

TIBCO Software Inc.
3303 Hillview Avenue
Palo Alto, CA 94304
USA

TIBCO®

Table of Contents

TIBCO®

Table of Figures

1 Overview

1.1 Document Purpose

The purpose of this document is to provide information on configuring Apache Kafka and Zookeeper on Linux to use Microsoft Active Directory (AD) for authentication. Active Directory will run on a Windows Server.

The documentation will provide a simple setup of the Simple Authentication and Security Layer (SASL) setup of Zookeeper/Kafka Broker. The configuration can then be expanded to support TLS and multiple brokers if desired (not documented).

The document will outline:

- Create an AD user for both Zookeeper and Kafka on the Windows Server
- Set a Service Principal Name (SPN) to be used with AD and Kerberos
- Create a Kerberos keytab file for Zookeeper and Kafka
- Secure the keytab file
- Configure Kerberos on Linux
- Configure a single Zookeeper and Kafka Broker to use the keytab file for Kerberos authentication
- Connect Zookeeper and Kafka using AD Authentication

1.2 Prerequisites

The following prerequisites must be completed before the configuring Kafka for AD Authentication.

Note: Earlier versions of the different Operating Systems and Apache Kafka should work. All configuration and testing was based on the versions listed below.

- RedHat 8 (or equivalent) must be running and configured with Apache Kafka 2.8 installed. The TIBCO download of Apache Kafka was used, but the *opensource* version from Apache will also work.
- Kerberos 5 installed on the Linux server. This can be a full Kerberos installation of just the *workstation* version. – *sudo yum install krb5-workstation.*
- Windows 2022 Server with Active Directory install/configured. There are a number of sites on the internet which can be used to install and configure Active Directory on the Windows 2022 server. Windows 2019 can also be used.
- **ALL** servers, Linux and Windows, **must** have a *fully qualified DNS name* (FQDN) that **must** be resolvable by all servers. This can be done using a *DNS Server,* or properly configuring */etc/hosts* on both servers.

Note: Throughout this document the following name were used to reference the servers and domain

- MYLAB.LOCAL – domain name

- advm – Windows 2022 server
- kafka1 – RHEL Linux Server

kafka1.mylab.local must resolve to the Linux server, and advm.mylab.local must resolve to the Windows server.

2 Windows Active Directory Configuration

After all of the prerequisites have been met, AD on the Windows Server can be configured for the Zookeeper and Kafka logons. Use this section to configure AD.

Note: The following requires a user with *administrator* access.

2.1 Active Directory Configuration

- Open a *PowerShell* session on the Windows server
- Run *Import-Module ActiveDirectory*
- Run *New-ADUser kafka -AccountPassword (Read-Host -AsSecureString "enter Password")* – enter a strong password
- Run *New-ADUser zookeeper -AccountPassword (Read-Host -AsSecureString "enter Password")* – enter a strong password
- Leave the *PowerShell* session *open*
- Update the user accounts for additional properties:
    - Use *ServerManager – Tools – Active Directory Users and Computers*
    - Click on the *kafka* user, and then ***properties***
    - Click on the *Account* tab
        - The *User logon name* must be updated to *kafka/<FQDN of the Linux Server>*. Using the referenced servers in section 1.2, this would be *kafka/kafka1.mylab.local.* Adjust accordingly
        - For Account options, click on:
            - *Password never expires*
            - *This account supports Kerberos AES 256 bit encryption*
        - Click *Apply* and *OK*
    - Repeat the same steps for the *zookeeper* user
- Exit from Active Directory Users and Computers

2.2 Service Principal Name

The Service Principal Name (SPN) is used in Kerberos/Kafka to know which AD user to use. It will be referenced in the Kerberos and Kafka/Zookeeper configurations.

Note: This section is based on the *server* and *FQDN* referenced in section 1.2. Adjust accordingly.

Return to the *PowerShell* session and do the following:

- Run *setspn -A kafka/kafka1.mylab.local@MYLAB.LOCAL kafka@mylab.local*
- Run *setspn -A kafka/kafka1.mylab.local kafka@mylab.local*
- Run *setspn -A zookeeper/kafka1.mylab.local@MYLAB.LOCAL zookeeper@mylab.local*
- Run *setspn -A zookeeper/kafka1.mylab.local zookeeper@mylab.local*

The *keytab* must still be created, but is after the initial Kerberos steps on Linux are completed. Leave the *PowerShell* session open.

3 Kerberos Setup on Linux

3.1 Krb5 configuration

This section will outline the Kerberos setup on the Linux server. Since only the Kerberos workstation is required, only the */etc/krb5.conf* configuration file needs to be updated. Below is an example of the */etc/krb5.conf* configuration file. The necessary changes are identified in red. These line changes either need to be added or modified depending on the existing configuration file. This file must be updated as *root,* or using *sudo.*

Note: If *krb5.conf* is already used by other applications, the highlighted sections must be added to the existing file.

```
# To opt out of the system crypto-policies configuration of krb5, remove the
# symlink at /etc/krb5.conf.d/crypto-policies which will not be recreated.
includedir /etc/krb5.conf.d/

[logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

[libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
    spake_preauth_groups = edwards25519
    default_realm = MYLAB.LOCAL
    default_ccache_name = KEYRING:persistent:%{uid}
    dns_lookup_kdc = false
    default_tkt_enctypes = aes256-cts (1)
    default_tgs_enctypes = aes256-cts (2)
    permitted_enctypes = aes256-cts (3)
[realms]
 MYLAB.LOCAL = { (4)
     kdc = advm.mylab.local (4)
     admin_server =advm.mylab.local (4)
     default_domain = mylab.local (4)
 }

[domain_realm]
 .mylab.local = MYLAB.LOCAL (5)
 mylab.local = MYLAB.LOCAL (5)
 .MYLAB.LOCAL = MYLAB.LOCAL (5)
 MYLAB.LOCAL = MYLAB.LOCAL (5)
```

Figure 1 - krb5.conf example

(1) This line must be added. This is to define AES256 encryption
(2) This line must be added. This is to define AES256 encryption
(3) This line must be added. This defines that AES256 encryption is the only permitted encryption. **Note:** If other encryption types are required, these can be added, but AD user must also be updated on Windows.
(4) This section defines the Kerberos realm, as well as the default domain, the kdc srver, and admin server, which are the same as the AD domain. In this case, *mylab.local.*
(5) The domain_realm definitions. The *domain_realm* definition must be the same as shown, other than the name. This will ensure that both lower and uppercase will be accepted.

Once *krb5.conf* is updated, the permissions must be updated so that the Kafka user can *read* the file, but not modify it.

sudo chmod 744 /etc/krb5.conf

3.2 Getting the KVMO values

Check the Key Version Number (KVNO) for the AD account created in section 2. Usually it is 2, but it could be another integer if the account's password has changed multiple times. On the Linux server machine, run the following commands for both the *zookeeper* and *kafka* users using the domain name. In this case *MYLAB.LOCAL.* Use the password created for the user in section 2.

```
$ kinit kafka@MYLAB.LOCAL
Password for kafka@MYLAB.LOCAL:
$ kvno kafka@MYLAB.LOCAL
kafka@MYLAB.LOCAL: kvno = 2
$ kvno kafka/kafka1.mylab.local@MYLAB.LOCAL
kafka/kafka1.mylab.local@MYLAB.LOCAL: kvno = 2
```

Figure 2 - Getting the KVNO

Repeat the above example to for the zookeeper user. Record the kvno for both users. They will be used in the next step.
Getting the Key Version Number (KVNO) also verifies that Kerberos client is working on Linux, and can communicate with Active Directory (AD) on the Windows Server using the AD username created. If there are any issues, these must be resolved before continuing.

3.3 KTPASS

The next step is to run *ktpass* on the Windows Server to create the user keytabs. The *PowerShell* session used in section 2 should still be available, and can be used.
For this environment, *zookeeper.keytab* and *kafka.keytab* will be created.

Use the *kvno* number recorded in the previous step and the password created previously for the account. In the examples, *kafka1.mylab.local* is the FQDN server name of the Linux server.

Note: Other encryption types can be used. *Krb5.conf* on the Linux server must permit these, and *ktpass* must be ran for the other encryptions.

```
cd c:\Users\<windows user>\Desktop

ktpass /princ kafka/kakfa1.mylab.local@MYLAB.LOCAL /ptype KRB5_NT_PRINCIPAL
/crytpo aes256-sha1 /mapuser kafka@mylab.local /out k1.keytab -setpass -setupn
/kvno <kafka kvno> /pass <kafka password>

ktpass /princ kafka@MYLAB.LOCAL /ptype KRB5_NT_PRINCIPAL /crytpo aes256-
sha1 /mapuser kafka@mylab.local /out kafka.keytab /in k1.keytab -setpass
-setupn /kvno <kafka kvno> /pass <kafka password>

ktpass /princ zookeeper/kakfa1.mylab.local@MYLAB.LOCAL /ptype KRB5_NT_PRINCIPAL
/crytpo aes256-sha1 /mapuser zookeeper@mylab.local /out z1.keytab -setpass -
setupn /kvno <zookeeper kvno> /pass <zookeeepr password>

ktpass /princ zookeeper@MYLAB.LOCAL /ptype KRB5_NT_PRINCIPAL /crytpo
aes256-sha1 /mapuser zookeeper@mylab.local /out zookeepey.keytab /in
z1.keytab -setpass -setupn /kvno <zookeeper kvno> /pass <zookeeper
password>
```

Figure 3 - Create the Keytabs

Once the *kafka.keytab* and *zookeeper.keytab* have been created, these should be copied to the Linux server using *scp.* Copy the keytabs to a *secure* directory, such as */etc/keytab.*

On the Linux server, change the ownership and permissions, so it is owned by *root,* but can be read by the user account where Kafka is installed.

```
$ sudo chown /etc/keytab/*.keytab
$ sudo cmmod 744 /etc/keytab/*.keytab
```

The Linux and Windows environments should now be ready to configure Apache Zookeeper and Kafka.

4 Zookeeper and Kafka Setup for Kerberos

Zookeeper and Kafka can now be configured to use SASL for authentication with Kerberos/Active Directory.

4.1 General Setup

There are some general setup steps which need to be performed before setting up the Apache components. This section will outline these steps.

4.1.1 JAAS Configuration File

Add a suitably modified JAAS file similar to the one below to the Kafka broker's config directory. In the following example, it is *kafka_server_jaas.conf*. In the below example, the server is *kafka1,* and the domain is *mylab.local.*

Note: This document is based on one Kafka Broker and one Zookeeper that are on the same Linux server. In the case with multiple brokers, each broker should have its own keytab setup using the steps previously discussed.

```
KafkaServer {
        com.sun.security.auth.module.Krb5LoginModule required
        useKeyTab=true
        storeKey=true
        keyTab="/etc/keytab/kafka.keytab"
        principal="kafka/kafka1.mylab.local@MYLAB.LOCAL";
    };
// Zookeeper client authentication
Client {
       com.sun.security.auth.module.Krb5LoginModule required
       useKeyTab=true
       storeKey=true
       keyTab="/etc/keytab/kafka.keytab"
       principal="kafka/kafka1.mylab.local@MYLAB.LOCAL";
    };
KafkaClient {
        com.sun.security.auth.module.Krb5LoginModule required
        useKeyTab=true
        storeKey=true
        keyTab="/etc/keytab/kafka.keytab"
        principal="kafka/kafka1.mylab.local@MYLAB.LOCAL";
    };
Server {
        com.sun.security.auth.module.Krb5LoginModule required
        useKeyTab=true
        keyTab="/etc/keytab/zookeeper.keytab"
        storeKey=true
        useTicketCache=false
        principal="zookeeper/kafka1.mylab.local@MYLAB.LOCAL";
};
```

Figure 4 - JASS configuration file example

In this example:

- KafkaServer is used to connect between the Kafka Brokers
- Client is used by Kafka to connect to Zookeeper
- KafkaClient is used by the Kafka clients to connect the Kafka Broker
- Server is used by Zookeeper to authenticate

4.1.2 KAFKA_OPTS

The *KAFKA_OPTS* environment variable must be updated based on the following example. If *KAFKA_OPTS* already exists, the following must be added. Substitute the appropriate location for the *jaas configuration file.*

It is recommend that environment variable be added to *.bash_profile* and *exported* before running the Apache components.

```
export KAFKA_OPTS=-Djava.security.krb5.conf=/etc/krb5.conf -
Djava.security.auth.login.config=/opt/tibco/akd/core/2.8/config/kafka_server_ja
as.conf
```

Figure 5 - KAFKA_OPTS example

4.2 Kafka Setup

Zookeeper does not require any configuration changes. However, the Kafka configuration file, *server.properties,* does need some updates. The following lines must be updated or added to server.properties.
As throughout this document, *kafka1* is the Linux server, while the domain is *mylab.local.*

Note: The following is only adding *authentication* to Kafka, if ssl/tls is also required, see the Apache Kafka documentation for details for enabling ssl. For details, see https://kafka.apache.org/090/documentation.html#security_sasl.

```
listeners=SASL_PLAINTEXT://kafka1.mylab.local:9092
sasl.kerberos.service.name=kafka
security.inter.broker.protocol=SASL_PLAINTEXT
```

Figure 6 - Server.properties additions

Zookeeper and the Kafka broker can be started now.

The Zookeeper log file should have a reference similar to the following showing a successful start with authentication.

```
[2021-10-18 18:05:34,057] INFO TGT refresh thread started. (org.apache.zookeeper.Login)
[2021-10-18 18:05:34,057] DEBUG Client principal is
"zookeeper/kafka1.mylab.local@MYLAB.LOCAL". (org.apache.zookeeper.Login)
[2021-10-18 18:05:34,057] DEBUG Server principal is "krbtgt/MYLAB.LOCAL@MYLAB.LOCAL".
(org.apache.zookeeper.Login)
```

```
[2021-10-18 18:05:34,058] INFO Configuring NIO connection handler with 10s sessionless
connection timeout, 1 selector thread(s), 8 worker threads, and 64 kB direct buffers. (
org.apache.zookeeper.server.NIOServerCnxnFactory)
[2021-10-18 18:05:34,060] INFO TGT valid starting at:        Mon Oct 18 18:05:34 UTC
2021 (org.apache.zookeeper.Login)
[2021-10-18 18:05:34,060] INFO TGT expires:                  Tue Oct 19 04:05:34 UTC
2021 (org.apache.zookeeper.Login)
[2021-10-18 18:05:34,060] INFO TGT refresh sleeping until: Tue Oct 19 02:09:00 UTC 2021
(org.apache.zookeeper.Login)
.
.
.
[2021-10-18 18:06:04,151] INFO Setting authorizedID:
kafka/kafka1.mylab.local@MYLAB.LOCAL
(org.apache.zookeeper.server.auth.SaslServerCallbackHandler)
[2021-10-18 18:06:04,151] INFO adding SASL authorization for authorizationID:
kafka/kafka1.mylab.local@MYLAB.LOCAL (org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-18 18:06:04,163] DEBUG Checking session 0x10000742c6b0000
(org.apache.zookeeper.server.SessionTrackerImpl)
[2021-10-18 18:06:04,164] DEBUG Processing ACL: 31,s{'world,'anyone}
 (org.apache.zookeeper.server.PrepRequestProcessor)
[2021-10-18 18:06:04,165] DEBUG Permission requested: 4
(org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-18 18:06:04,165] DEBUG ACLs for node: [31,s{'world,'anyone}
] (org.apache.zookeeper.server.ZooKeeperServer)
[2021-10-18 18:06:04,165] DEBUG Client credentials:
['sasl,'kafka/kafka1.mylab.local@MYLAB.LOCAL
, 'ip,'10.0.1.5
] (org.apache.zookeeper.server.ZooKeeperServer)
```

Figure 7 - Zookeeper successful start with authentication

The Kafka log file should have a reference similar to the following showing a successful start with authentication.

```
[2021-10-18 20:05:22,281] INFO Successfully logged in.
(org.apache.kafka.common.security.authenticator.AbstractLogin)
[2021-10-18 20:05:22,281] INFO [Principal=kafka/kafka1.mylab.local@MYLAB.LOCAL]: TGT
refresh thread started. (org.apache.kafka.common.security.kerberos.KerberosLogin)
[2021-10-18 20:05:22,282] INFO [Principal=kafka/kafka1.mylab.local@MYLAB.LOCAL]: TGT
valid starting at: Mon Oct 18 20:05:22 UTC 2021
(org.apache.kafka.common.security.kerbero
s.KerberosLogin)
[2021-10-18 20:05:22,282] INFO [Principal=kafka/kafka1.mylab.local@MYLAB.LOCAL]: TGT
expires: Tue Oct 19 06:05:22 UTC 2021
(org.apache.kafka.common.security.kerberos.Kerberos
Login)
[2021-10-18 20:05:22,282] INFO [Principal=kafka/kafka1.mylab.local@MYLAB.LOCAL]: TGT
refresh sleeping until: Tue Oct 19 04:33:18 UTC 2021
(org.apache.kafka.common.security.ke
rberos.KerberosLogin)
[2021-10-18 20:05:22,300] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Created
data-plane acceptor and processors for endpoint : ListenerName(SASL_PLAINTEXT) (kafka.n
etwork.SocketServer)
[2021-10-18 20:05:22,328] INFO [broker-0-to-controller-send-thread]: Starting
(kafka.server.BrokerToControllerRequestThread)
```

Figure 8 - Kafka successful start with authentication