

Securing CDW on Private Cloud

Date published: 2020-08-17

Date modified: 2024-10-18



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

User authentication methods.....	4
Kerberos principals.....	4
Enable SSL for databases.....	5
Enable SSL for MySQL.....	5
Enable SSL for MariaDB.....	8
Enable SSL for Oracle.....	10
Enabling mTLS between the HMS database and CDW on Private Cloud.....	12
SSL-enabled client endpoints.....	12
HDFS encryption with CDW.....	14
Enable access to Kerberized HDFS.....	15
Encrypt spilled Impala data.....	16

Authenticating users in CDW Private Cloud

Cloudera Data Warehouse (CDW) supports LDAP and Kerberos for authenticating users to access databases and tables from Business Intelligence (BI) clients and tools such as Hue, Beeline, Impyla, Impala-shell, and other JDBC clients.



Note: Hive and Impala Virtual Warehouses can use LDAP or Kerberos for authentication, simultaneously, without any pre-configuration.

Authentication using LDAP

If you use LDAP to authenticate users connecting to a Virtual Warehouse from a client shell such as Beeline, Impyla, Impala-shell, and so on, then you must create Bind users. However, the Bind users must either specify the username and password inline with the command or after submitting the command from the client.



Note: You can now use an unsecured LDAP server for authenticating users in the Impala Virtual Warehouse. CDW uses the LDAP configurations that you have configured in the CDP Management Console.

Authentication using Kerberos

Kerberos uses passwords stored in the Kerberos keytab files. This removes the need to specify username and password as parameters inline with the command or after submitting the command from a JDBC client to connect to a Virtual Warehouse in CDW, while adding a layer of security.



Important:

- The Private Cloud Base cluster must be Kerberized to use Kerberos as the authentication mode for CDW.
- CDW requires all user Kerberos principals to be present in the configured LDAP server as well.
- When you enable warehouse-level access control for Hive warehouses or Impala warehouses in the Unified Analytics mode and associate a user group with that Virtual Warehouse, Kerberos authentication is disabled. Only LDAP is used for authentication.

Related Information

[Enabling warehouse-level access control in CDW Private Cloud](#)

How predefined Kerberos principals are used in CDW Private Cloud

By default, Cloudera Data Warehouse (CDW) creates Kerberos principal names for Database Catalogs and Environments using the service hostname and the deterministic namespace name based on the name of the Database Catalog or Environment when you create a Database Catalog or an Environment. However, you can generate and provide the keytabs, if needed.

The service principals for CDW need to be the same as on the base cluster. For more information, see [Customizing Kerberos principals in the CDP Private Base documentation](#).

By default, the host principals are generated programmatically. You can generate and provide the keytabs, but the hostnames in the Kerberos principals are fixed. CDW uses a deterministic namespace and environment IDs for the Kerberos principals.

When you specify an Environment or Database Catalog name, CDW appends a prefix as shown in the following table, as well as the Kerberos principal name based on them:

CDW entity	User-specified name	Namespace IDs with CDW-assigned prefix	Hive Kerberos principal name
Environment	my-test-env	env-my-test-env-default	hive/dwx-env-my-test-env.cdp.local@REALM.EXAMPLE.COM
Database Catalog	my-test-catalog	warehouse-my-test-catalog	hive/metastore-service.warehouse-warehouse-my-test-catalog.svc.cluster.local@REALM.EXAMPLE.COM
Virtual Warehouse	my-impala-warehouse	impala-my-impala-warehouse	NA



Note: The length of the namespace ID after CDW applies a prefix to the Environment or Database Catalog name, including the hyphen (-), should not exceed 63 characters. You can specify an Environment name 35 characters long and Database Catalog 53 characters long.

When using FreeIPA, the environment name can be maximum 17 characters long.

How to enable SSL for MariaDB, MySQL, and Oracle databases

Cloudera requires that you secure the network connection between the default Database Catalog Hive MetaStore (HMS) in Cloudera Data Warehouse (CDW) and the relational database hosting the base cluster's HMS using SSL encryption.

You must provide the SSL certificate of the database either by:

- Providing the SSL certificate while installing the Data Services on the [Install Private Cloud Data Services on Existing Container Cloud Configure Kubernetes](#) step under the Additional Certificates section. See [Installing in an internet environment](#).
- Importing the SSL certificate to the trust store on the base cluster before installing CDP Private Cloud.

Configuring MySQL database to use SSL for Data Warehouse

SSL encrypts the connection between the MySQL server and the Hive MetaStore (HMS) on the base cluster. You must enable SSL for the MySQL database before setting up the CDP Private Cloud Data Services and add the MySQL root Certificate Authorities (CA) to the Cloudera Manager truststore.

Procedure

1. SSH into the MySQL database host.
2. Start the MySQL server:

```
service mysqld start
```

3. Establish an encrypted connection with the client:

```
mysql -p --ssl-mode=required
```

4. Verify whether SSL is enabled on MySQL by running the following command:

```
mysql> show global variables like '%ssl%';
```

If SSL is enabled, you see the value of `have_ssl` equal to YES, as follows. Otherwise, you see the value of `have_ssl` equal to DISABLED:

```
+-----+-----+
| Variable_name | Value |
```

+-----+-----+	
have_openssl	YES
have_ssl	YES
...	...

If SSL is enabled, then skip to step 11.

5. Create a certificate authority by running the following commands:

```
mkdir /etc/my.cnf.d/ssl/
cd /etc/my.cnf.d/ssl/
openssl genrsa 2048 > ca-key.pem
```

6. Create a certificate for the server using the CA certificate generated earlier by running the following command:

```
openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca-cert.
pem
openssl req -newkey rsa:2048 -days 365 -nodes -keyout server-key.pem -out
server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
```

7. Create a certificate for the clients using the same CA certificate by running the following command:

```
openssl x509 -req -in server-req.pem -days 365 -CA ca-cert.pem -CAkey ca-
key.pem -set_serial 01 -out server-cert.pem
```

8. Add the following lines in the /etc/my.cnf.d/server.cnf file under the [mysqld] section:

```
ssl-ca=/etc/my.cnf.d/ssl/ca-cert.pem
ssl-cert=/etc/my.cnf.d/ssl/server-cert.pem
ssl-key=/etc/my.cnf.d/ssl/server-key.pem
bind-address=*
```

You can view the content of the server.cnf file by running the following command:

```
vim /etc/my.cnf.d/server.cnf
```

9. Restart the MySQL server:

```
service mysqld restart
```

10. Check the SSL status by running the following commands:

```
mysql -p --ssl-mode=required
> SHOW VARIABLES LIKE '%ssl%';
> status
```

Sample output:

```
> SHOW VARIABLES LIKE '%ssl%';
```

+-----+-----+	
Variable_name	Value
admin_ssl_ca	
admin_ssl_capath	
admin_ssl_cert	
admin_ssl_cipher	
admin_ssl_crl	
admin_ssl_crlpath	
admin_ssl_key	
have_openssl	YES
have_ssl	YES

mysqlx_ssl_ca	
mysqlx_ssl_capath	
mysqlx_ssl_cert	
mysqlx_ssl_cipher	
mysqlx_ssl_crl	
mysqlx_ssl_crlpath	
mysqlx_ssl_key	
performance_schema_show_processlist	OFF
ssl_ca	ca.pem
ssl_capath	
ssl_cert	server-cert.pem
ssl_cipher	
ssl_crl	
ssl_crlpath	
ssl_fips_mode	OFF
ssl_key	server-key.pem

```

+-----+
> status
SSL:  Cipher in use is ECDHE-RSA-AES128-GCM-SHA256

```

11. View the contents of the ssl-client.xml file by running the following commands:

```
export SSL_CLIENT=/etc/hadoop/conf/ssl-client.xml
cat $SSL_CLIENT
```

12. Obtain the truststore's location and password by running the following commands:

```
export TRUSTSTORE_LOCATION=$(xmllint --xpath "//configuration/property[name='ssl.client.truststore.location']/value/text()" $SSL_CLIENT)
```

```
export TRUSTSTORE_PASSWORD=$(xmllint --xpath "//configuration/property[name='ssl.client.truststore.password']/value/text()" $SSL_CLIENT)
```

13. Verify the contents of the truststore by running the following command:

```
/usr/java/default/bin/keytool -list -rfc -keystore $TRUSTSTORE_LOCATION -storetype JKS -storepass $TRUSTSTORE_PASSWORD
```

14. Import the MySQL root certificate by running the following command:

```
/usr/java/default/bin/keytool -importcert -alias mysql -file /var/lib/mysql/ca.pem -keystore $TRUSTSTORE_LOCATION -storetype jks -noprompt -storepass $TRUSTSTORE_PASSWORD
```

15. Verify the contents of the truststore again by running the following command:

```
/usr/java/default/bin/keytool -list -rfc -keystore $TRUSTSTORE_LOCATION -storetype JKS -storepass $TRUSTSTORE_PASSWORD
```

Results

When you install CDP Private Cloud Data Services after adding the MySQL root CA to the Cloudera Manager truststore, the installer propagates the MySQL root CA from the Cloudera Manager truststore to CDP Private Cloud. HMS in the default Database Catalog can now connect to the MySQL server on the base cluster using an SSL-encrypted connection.

Configuring MariaDB database to use SSL for Data Warehouse

SSL encrypts the connection between the MariaDB server and the Hive MetaStore (HMS) on the base cluster. You must enable SSL for the MariaDB database before setting up the CDP Private Cloud Data Services.

Procedure

1. SSH into the MariaDB database host.
2. Start the MariaDB server:

```
service mysqld start
```

3. Establish an encrypted connection with the client:

```
mysql -p --ssl=true
```

4. Verify whether SSL is enabled on MariaDB by running the following command:

```
mysql> show global variables like '%ssl%';
```

If SSL is enabled, you see the value of `have_ssl` equal to YES, as follows. Otherwise, you see the value of `have_ssl` equal to DISABLED:

Variable_name	Value
have_openssl	YES
have_ssl	YES
...	...

If SSL is enabled, then skip to step 11.

5. Create a certificate authority by running the following commands:

```
mkdir /etc/my.cnf.d/ssl/
cd /etc/my.cnf.d/ssl/
openssl genrsa 2048 > ca-key.pem
```

6. Create a certificate for the server using the CA certificate generated earlier by running the following command:

```
openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca-cert.pem
openssl req -newkey rsa:2048 -days 365 -nodes -keyout server-key.pem -out server-req.pem
openssl rsa -in server-key.pem -out server-key.pem
```

7. Create a certificate for the clients using the same CA certificate by running the following command:

```
openssl x509 -req -in server-req.pem -days 365 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem
```

8. Add the following lines in the `/etc/my.cnf.d/server.cnf` file under the `[mysqld]` section:

```
ssl-ca=/etc/my.cnf.d/ssl/ca-cert.pem
ssl-cert=/etc/my.cnf.d/ssl/server-cert.pem
ssl-key=/etc/my.cnf.d/ssl/server-key.pem
```



```
bind-address=*
```

You can view the content of the server.cnf file by running the following command:

```
vim /etc/my.cnf.d/server.cnf
```

9. Restart the MariaDB server:

```
service mysqld restart
```

10. Check the SSL status by running the following commands:

```
mysql -p --ssl=true
> SHOW VARIABLES LIKE '%ssl%';
> status
```

Sample output:

```
> SHOW VARIABLES LIKE '%ssl%';
+-----+-----+
| Variable_name | Value                                     |
+-----+-----+
| have_openssl  | YES                                     |
| have_ssl      | YES                                     |
| ssl_ca        | /etc/my.cnf.d/ssl/ca-cert.pem         |
| ssl_capath    |                                         |
| ssl_cert      | /etc/my.cnf.d/ssl/server-cert.pem     |
| ssl_cipher    |                                         |
| ssl_crl       |                                         |
| ssl_crlpath   |                                         |
| ssl_key       | /etc/my.cnf.d/ssl/server-key.pem      |
| version_ssl_library | OpenSSL 1.0.2k-fips 26 Jan 2017 |
+-----+-----+

> status
SSL:  Cipher in use is DHE-RSA-AES256-GCM-SHA384
```

11. View the contents of the ssl-client.xml file by running the following commands:

```
export SSL_CLIENT=/etc/hadoop/conf/ssl-client.xml
cat $SSL_CLIENT
```

12. Obtain the truststore's location and password by running the following commands:

```
export TRUSTSTORE_LOCATION=$(xmllint --xpath "//configuration/property[n
ame='ssl.client.truststore.location']/value/text()" $SSL_CLIENT)
```

```
export TRUSTSTORE_PASSWORD=$(xmllint --xpath "//configuration/property[n
ame='ssl.client.truststore.password']/value/text()" $SSL_CLIENT)
```

13. Verify the contents of the truststore by running the following command:

```
/usr/java/default/bin/keytool -list -rfc -keystore $TRUSTSTORE_LOCATION -
storetype JKS -storepass $TRUSTSTORE_PASSWORD
```

14. Import the MariaDB root certificate by running the following command:

```
/usr/java/default/bin/keytool -importcert -alias mariadb -file /etc/my.c
nf.d/ssl/ca-cert.pem -keystore $TRUSTSTORE_LOCATION -storetype jks -nopr
ompt -storepass $TRUSTSTORE_PASSWORD
```

15. Verify the contents of the truststore again by running the following command:

```
/usr/java/default/bin/keytool -list -rfc -keystore $TRUSTSTORE_LOCATION -storetype JKS -storepass $TRUSTSTORE_PASSWORD
```

Results

When you install CDP Private Cloud Data Services after adding the MariaDB root CA to the Cloudera Manager truststore, the installer propagates the MariaDB root CA from the Cloudera Manager truststore to CDP Private Cloud. HMS in the default Database Catalog can now connect to the MariaDB server on the base cluster using an SSL-encrypted connection.

Configuring Oracle database to use SSL for Data Warehouse

You must enable SSL for the Oracle database before setting up the CDP Private Cloud Data Services. Enabling SSL establishes a secure channel between the client (CDP-side) and the server (Oracle database server).

About this task

To enable SSL, you need to configure SSL only on the server side. The client-side configurations are present in CDP.

Procedure

1. SSH into the Oracle database server host.
2. Change to the "oracle" user as follows:

```
sudo -su oracle
```

3. Append the location of ORACLE_HOME to the PATH environment variable by running the following commands:

```
export ORACLE_HOME=/opt/oracle/product/19c/dbhome_1
export PATH=${PATH}:${ORACLE_HOME}/bin
```

4. Create an auto-login wallet by running the following command:

```
orapki wallet create -wallet /opt/oracle/product/19c/dbhome_1/wallet -auto_login
```

An auto-login wallet uses SSL's single sign-on functionality. The users do not need to specify password each time they open the wallet.

5. Add a self-signed certificate to this wallet by running the following command:

```
orapki wallet add -wallet /opt/oracle/product/19c/dbhome_1/wallet -dn "CN=server" -keysize 4096 -self_signed -validity 365
```

6. Export the certificate from the Oracle wallet by running the following command:

```
orapki wallet export -wallet /opt/oracle/product/19c/dbhome_1/wallet -dn "CN=server" -cert server_ca.cert
```

This exports a certificate with the subject's distinguished name (-dn) (CN=server) from a wallet to the file that is specified by -cert (server_ca.cert).

7. Add the following lines to the /opt/oracle/product/19c/dbhome_1/network/admin/listener.ora configuration file:

```
SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
(SOURCE =
```

```
(METHOD = FILE)
(METHOD_DATA =
  (DIRECTORY = /opt/oracle/product/19c/dbhome_1/wallet)
)
)
Register a new address in LISTENER:
(ADDRESS = (PROTOCOL = TCPS)(HOST = [***HOST***])(PORT = 2484))
```

8. Add the following lines to the /opt/oracle/product/19c/dbhome_1/network/admin/sqlnet.ora profile configuration file:

```
SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = /opt/oracle/product/19c/dbhome_1/wallet)
    )
  )
```

9. Add the following lines to the /opt/oracle/product/19c/dbhome_1/network/admin/tnsnames.ora configuration file:

```
ORCLPDB1_SSL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCPS)(HOST = [***HOST***])(PORT = 2484))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCLPDB1)
    )
    (SECURITY =
      (MY_WALLET_DIRECTORY = /opt/oracle/product/19c/dbhome_1/wallet)
    )
  )
```

10. Restart the listener by running the following commands:

```
lsnrctl stop
lsnrctl start
```

11. Copy the content of the certificate that you exported earlier and add it to the keystore on the base cluster instances.

Paste the copied content to the ca-cert.pem file.

12. Fetch the keystore password from the /etc/hadoop/conf/ssl-client.xml file by running the following command:

```
/usr/java/default/bin/keytool -importcert -alias oracle -file ca-cert.pem -keystore /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks -storetype jks -noprompt -storepass [***PASSWORD***]
```

13. Log in to Cloudera Manager as an Administrator.

14. Go to Clusters Hive service Configuration Hive Metastore Server Advanced Configuration Snippet (Safety

Valve) for hive-site.xml and click **+** to add the following:

- Name: javax.jdo.option.ConnectionURL
- Value: jdbc:oracle:thin:@tcps://[***BASE_CLUSTER_HOSTNAME***]:2484/ORCLPDB1?javax.net.ssl.trustStore=/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks&javax.net.ssl.trustStorePassword=[***PASSWORD***]&oracle.net.ssl_server_dn_match=false

15. Change the port to 2484 in the Hive Metastore Database Port field.

16. Click Save Changes.

17. Restart the Hive service.

Enabling mTLS between the HMS database and CDW on Private Cloud

In setups where mTLS is used for secure authentication and communication between HMS service and HMS databases, Cloudera Data Warehouse (CDW) also supports setting up mTLS for this connectivity. The Hive MetaStore (HMS) pod in CDW needs a client certificate and client private key to set up the mTLS authentication.

About this task



Note: CDW does not support password-protected private keys.

Before you begin

- Verify whether mTLS is configured on the CDP Base cluster and it is the only means of authentication to the HMS database. Perform this task only if you meet this condition.
- Ensure that the database client certificate and private key files are present on the CDP Base cluster.
- You must have an environment available that you want to activate in CDW.

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Go to the **Environment(s)** tab and click Activate corresponding to the environment you want to activate in CDW.
3. Browse and upload the database client certificate and database client private key in PEM format.
4. Click ACTIVATE.

SSL-enabled endpoints for Virtual Warehouse clients in CDW on Private Cloud

In Cloudera Data Warehouse (CDW) Private Cloud, all client endpoints have been SSL-enabled. This requires that you configure the SSL certificates for client endpoints.

The client endpoints for web applications and Virtual Warehouse client URLs are SSL-enabled. The following endpoints use the OpenShift/Embedded Container Service cluster default certificate:

- Hue
- Impala coordinator
- HiveServer2

Domain name changes

To use the OpenShift/Embedded Container Service cluster wildcard certificate, the DNS names have been changed. The environment ID sub domain from the domain name has been removed. This creates a flat DNA structure so the cluster wildcard certificate can be applied to the endpoints.

Generating a truststore for a self-signed certificate

You can query the service certificate and convert it to a JKS truststore using the following steps:

1. Retrieve the certificate:

```
$ openssl s_client -showcerts -connect hs2-my-cwh1.apps.cdw.mycloud.myfirm.com:443 -servername
hs2-my-cwh1.apps.cdw.mycloud.myfirm.com </dev/null|openssl x509 -outform
PEM > <mycertfile>.pem
```

2. Convert the PEM file to a truststore. You will be prompted for a password.

```
$ keytool -import -alias hs2-my-cw1.apps.cdw.mycloud.myfirm.com -file
<mycertfile>.pem -keystore <mycert>.jks
```

Opening SSL-enabled connections with Database Catalog clients

The CDW Virtual Warehouse clients like beeline and impala-shell can open SSL-enabled connections as described in this section.

Beeline

A beeline connection can be created using a JDBC connection string. Specifying the username and password with the '-n' and the '-p' options returns an error. The beeline CLI prompts for credentials:

```
$ beeline
beeline> !connect
jdbc:hive2://hs2-my-cwh1.apps.cdw.mycloud.myfirm.com:443/default;transportMode=http;httpPath=cliservice;
    ssl=true;retries=3;sslTrustStore=<JKS-path>;trustStorePassword
=<***password***>
Enter username for jdbc:hive2://hs2-my-cwh1.apps.cdw.mycloud.myfirm.com:443/
default:<my-user-name>
Enter password for jdbc:hive2://hs2-my-cwh1.apps.cdw.mycloud.myfirm.com:443/
default:<*****>
```



Important: The value for <JKS-path> is generated in the above section "Generating a truststore for a self-signed certificate."

impala-shell

The impala-shell CLI opens a TLS/SSL-enabled connection when you use the '--ssl' option. If '--ca_cert' is not set, impala-shell enables TLS/SSL, but does not validate the server certificate. Set the '--ca_cert' CLI option to the local path name that points to the third-party CA certificate, or to a copy of the server certificate in the case you have a self-signed server certificate:

```
$ impala-shell --protocol='hs2-http' -i "coordinator-my-iwh2.apps.cdw.myclou
d.myfirm.com:443" --ssl
```

OpenShift routes

OpenShift routes are used to expose the user-facing services in the CDW Private Cloud deployment. Route objects can perform edge TLS termination using the cluster-deployed certificate for the endpoints. If the cluster certificate must be rotated, the routes can pick up the new certificate automatically. It is not necessary to re-deploy or to manually configure the service in order to pick up the changes.

HDFS encryption in the context of Data Warehouse on Private Cloud

Cloudera Data Warehouse (CDW) Data Service and its components such as Hive, Impala, and Hue can read and write encrypted data to HDFS on a Private Cloud Base cluster.

How HDFS encryption works with CDW

Encryption and decryption of the data happens in the HDFS client library. The client library is part of the client application such as, Hive, Impala, Spark, or any service that is reading or writing the data. To use this functionality encapsulated in the HDFS client library, the services must have access to the Hadoop Key Management Server (KMS) to retrieve the master key. KMS is a part of the Ranger service that runs in the base cluster. Cloudera recommends that you configure a secure cluster and then establish a secure channel between the encrypted HDFS cluster and the service using TLS.

All authorizations need an authenticated security principal – a user id, if it is a user, or a service account if it is a service.

SQL engines, such as Impala and Hive, and Hue as their front-end user interface need to authenticate the user connecting to them in order to authorize the user for various database-level operations, such as SELECT or INSERT, and to pass this user to the HDFS encryption ops to be authorized for those.

Understanding HDFS encryption zones in the context of CDW

Encryption Zone (EZ) is a directory in HDFS whose contents are automatically encrypted during write operations and decrypted during read operations. CDW can access data stored on the base cluster's HDFS, which can be set up with HDFS encryption.

You can configure the base cluster to have one or more HDFS encryption zones, a sub-directory encrypted with a particular master key. You can, then, store Hive and Impala tables in that sub-directory or you can store the entire Hive and Impala Warehouse in an encryption zone, encrypting the tables and metadata. CDW can then access the shared data from a Virtual Warehouse running in that extension.



Important: You can copy data between two encryption zones. Moving data between the encryption zones is neither possible nor recommended. SQL commands such as Impala's LOAD DATA INPATH statement copy the data instead of moving it when the data needs to cross encryption zone boundaries (reencrypting the data as necessary).

Conditions for enabling Impala to read and write encrypted data to HDFS

To access data in an encryption zone, you must set up authorization for various user principals.

To allow Impala to read and write encrypted data stored on HDFS, you must meet the following authorization permissions:

- You must have permissions to perform key operations for creating and accessing keys that encrypt your encryption zone (one key per zone).
- You must have read and write permissions to the HDFS sub-directory, which is your encryption zone.
- You must have permissions to perform various actions in the Hadoop SQL policy area in Ranger. For example, the ability to specify the LOCATION clause for a CREATE TABLE statement is a specific permission you have to grant to a user, which may be necessary if you have an encryption zone outside your warehouse directory and you want to write or read data there.

Encryption keys for the encryption zones are also managed in Ranger on the base cluster, together with their permissions (key ACLs).

You must grant these permissions to the user identities fetched from LDAP to make the base cluster and the CDW cluster refer to the same user identities in Ranger on the base cluster. You must configure the Ranger UserSync so that the base cluster can pull in the LDAP user identities.

You must also configure the Management Console to point to the same LDAP instance for authentication so that the base cluster and the CDW clusters are synchronized for user authentication.

If you are using Impala on encrypted source data, then ensure that data written to disk temporarily during processing is encrypted to keep the data confidential even during processing. For more information, see *Configuring Impala Virtual Warehouses to encrypt spilled data in CDW on Private Cloud*.



Note: The CDW cache is not encrypted. You must use a third-party disk encryption solution to encrypt data that is cached. Encrypting and decrypting data can degrade performance.

Related Information

[HDFS Transparent Encryption \(Cloudera documentation\)](#)

[Transparent Encryption in HDFS \(Apache documentation\)](#)

[HDFS encryption: Rename and Trash considerations](#)

[HDFS encryption: Distcp considerations](#)

[Creating Encryption Zones](#)

[Transparent Encryption Recommendations for Impala](#)

[Transparent Encryption Recommendations for Hive](#)

[Transparent Encryption Recommendations for Hue](#)

[Configuring Impala Virtual Warehouses to encrypt spilled data in CDW on Private Cloud](#)

Enabling browsing files on Kerberized HDFS from Hue in CDW Private Cloud

Because SPNEGO is disabled in Cloudera Data Warehouse (CDW) Private Cloud, by default, you cannot browse files on Kerberized HDFS cluster using the Hue File Browser. To enable access, you must set the `security_enabled` property to `true` in the Hue Advanced Configurations (`hue-safety-valve`) in CDW.

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Select a Virtual Warehouse on which you want to enable access to the Kerberized HDFS cluster from Hue and

click  Edit .

The **Virtual Warehouse Details** page is displayed.

3. Go to **Configurations Hue** , select `hue-safety -valve` from the Configuration files drop-down list, and add the following lines:

```
[hadoop]
[[hdfs_clusters]]
[[[default]]]
security_enabled=true
```

4. Click **Apply Changes**.

Configuring Impala Virtual Warehouses to encrypt spilled data in CDW on Private Cloud

If you have encrypted HDFS on the base CDP cluster, then Cloudera recommends that you configure an Impala Virtual Warehouse to write temporary data to disk during query processing in an encrypted format using the AES-256-CFB encryption for complete security.


About this task

In CDP Private Cloud, the temporary data is spilled to the local storage, the location of which is hard coded by the system.



Important: Impala does not selectively encrypt data based on whether the source data is already encrypted in HDFS. This results in at most 15 percent performance degradation when data is spilled.

Procedure

1. Log in to the Data Warehouse service as an administrator.
2. Go to Impala Virtual Warehouse  Edit CONFIGURATIONS Impala coordinator and select flagfile from the Configuration files drop-down list.
3. Set the value of the `disk_spill_encryption` property to true.
4. Click APPLY.
5. Go to the Impala executor tab and select flagfile from the Configuration files drop-down list.
6. Set the value of the `disk_spill_encryption` property to true.
7. Click APPLY.
8. Restart the Impala Virtual Warehouse.