Cloudera Data Warehouse Private Cloud 1.5.0

Getting Started with Private Cloud

Date published: 2020-08-17
Date modified: 2023-01-25

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Get started with OpenShift and Embedded Container Service standard resource mode requirements

Review the memory, storage, and hardware requirements for getting started with the Cloudera Data Warehouse (CDW) service in standard resource mode on Red Hat OpenShift and Embedded Container Service.

To get started with the CDW service on standard resource mode, make sure you have fulfilled the following requirements:

- CDP Cloudera Manager must be installed and running.
- CDP Private Cloud must be installed and running. See Installing on OpenShift and Installing on ECS for more details.
- An environment must have been registered with Management Console on the private cloud. See CDP Private Cloud Environments for more details.
- In addition to the general requirements, CDW also has the following minimum memory, storage, and hardware requirements for each worker node using the standard resource mode:

The following table lists the minimum and recommended compute (processor), memory, storage, and network bandwidth required for each OpenShift or ECS worker node using the Standard Resource Mode for production use case. Note that the actual node still needs some extra resources to run the operating system, Kubernetes engine, and Cloudera Manager agent on ECS.

Component	Minimum	Recommended
Node Count	4	10
CPU per worker	16 cores [or 8 cores or 16 threads that have Simultaneous Multithreading (SMT) enabled]	32+ cores (can also be achieved by enabling SMT)
Memory per worker	128 GB per node	384 GB* per node
FAST (Fully Automated Storage Tiering) Cache - Locally attached SCSI device(s) on every worker. Preferred: NVMe and SSD. OCP uses Local Storage Operator. ECS uses Local Path Provisioner.	1.2 TB[*] SATA, SSD per host	1.2 TB* NVMe/SSD per host
Network Bandwidth	1 GB/s guaranteed bandwidth to every CDP Private Cloud Base node	10 GB/s guaranteed bandwidth to every CDP Private Cloud Base node

Important: When you add memory and storage, it is very important that you add it in the increments as follows:

- Increments of 128 GB of memory
- Increments of 600 GB of locally attached SSD/NVMe storage

If you add memory or storage that is not in the above increments, the memory and storage that exceeds these increments is not used for executor pods. Instead, the extra memory and storage can be used by other pods that require fewer resources.

For example, if you add 200 GB of memory, only 128 GB is used by the executor pods. If you add 2 TB of locally attached storage, only 1.8 TB is used by the executor pods.

Related Information

Hyper-Threading

[*] Depending on the number of executors you want to run on each physical node, the per-node requirements change proportionally. For example, if you are running 3 executor pods per physical node, you require 384 GB of memory and approximately 1.8TB (600GB per executor) of locally attached SSD/NVMe storage for FAST Cache.

Low resource mode requirements

Review the memory, storage, and hardware requirements for getting started with the Cloudera Data Warehouse (CDW) service in low resource mode on Red Hat OpenShift and Embedded Container Service (ECS).

To get started with the CDW service on Red Hat OpenShift or ECS low resource mode, make sure you have fulfilled the following requirements:

Important: Lowering the minimum hardware requirement reduces the up-front investment to deploy CDW on OpenShift or ECS pods, but it does impact performance. Cloudera recommends that you use the Low Resource Mode option for proof of concept (POC) purposes only. This feature is not recommended for production deployment.

Complex queries and multiple queries on HS2 may fail due to limited memory configurations for HMS and HS2 in the low resource mode.

- CDP Cloudera Manager must be installed and running.
- CDP Private Cloud must be installed and running. See Installing on OpenShift and Installing on ECS for more details.
- An environment must have been registered with Management Console on the private cloud. See CDP Private Cloud Environments for more details.
- In addition to the general requirements, CDW also has the following minimum memory, storage, and hardware requirements for each worker node using the standard resource mode:

Component	Low resource mode deployment
Nodes	4
CPU	4
Memory	48 GB
Storage	3 x 100 GB (SATA) or 2 x 200 GB (SATA)
Network Bandwidth	1 GB/s guaranteed bandwidth to every CDP Private Cloud Base node

Important: When you add memory and storage for low resource mode, it is very important that you add it in the increments stated in the above table:

- increments of 48 GB of memory
- increments of at least 100 GB or 200 GB of SATA storage

If you add memory or storage that is not in the above increments, the memory and storage that exceeds these increments is not used for executor pods. Instead, the extra memory and storage can be used by other pods that require fewer resources.

Virtual Warehouse low resource mode resource requirements

The following requirements are in addition to the low resource mode requirements listed in the previous section.

Table 1: Impala Virtual Warehouse low resource mode requirements

Component	vCPU	Memory	Local Storage	Number of pods in XSMALL Virtual Warehouse
Coordinator (2)	2 x 0.4	2 x 24 GB	2 x 100 GB	2
Executor (2)	2 x 3	2 x 24 GB	2 x 100 GB	2
Statestore	0.1	512 MB	--	1
Catalogd	0.4	16 GB	--	1

Component	vCPU	Memory	Local Storage	Number of pods in XSMALL Virtual Warehouse
Auto-scaler	0.1	1 GB	--	1
Hue (backend)	0.5	8 GB	--	1
Hue (frontend)	--	--	--	1
Total for XSMALL Virtual Warehouse	8 (7.9)	121.5 GB	400 GB - 3 volumes	--

Impala Admission Control Configuration

- Maximum concurrent queries per executor: 4
- Maximum query memory limit: 8 GB

Table 2: Hive Virtual Warehouse low resource mode requirements

Component	vCPU	Memory	Local Storage	Number of pods in XSMALL Virtual Warehouse
Coordinator (2)	2 x 1	2 x 4 GB	2 x 100 GB	2
Executor (2)	2 x 4	2 x 48 GB (16 GB heap; 32 GB off-heap)	2 x 100 GB	2
HiveServer2	1	16 GB	--	1
DAS	0.5	4 GB	--	1
Hue (backend)	0.5	8 GB	--	1
Hue (frontend)	--	--	--	1
Standalone compute operator	0.1	100 MB (.1 GB)	--	--
Standalone query executor (separate)	Same as executor	Same as executor	Same as executor	--
Total for XSMALL Virtual Warehouse	21 (20.6)	237 GB (236.1)	400 GB - 4 volumes	--

Database Catalog low resource mode requirements

The HiveMetaStore (HMS) and the DAS event processor each use 2 CPUs and 8 GB of memory. Because HMS pods are in High Availability mode, they need a total of 4 CPUs and 16 GB of memory.

Total CPUs required: 6

Total memory required: 24 GB

Data Visualization low resource requirements

Table 3: Data Visualization low resource mode requirements

vCPU	Memory	Local Storage	Number of pods in XSMALL Virtual Warehouse
0.5	8 GB	--	1

Security requirements for Cloudera Data Warehouse Private Cloud

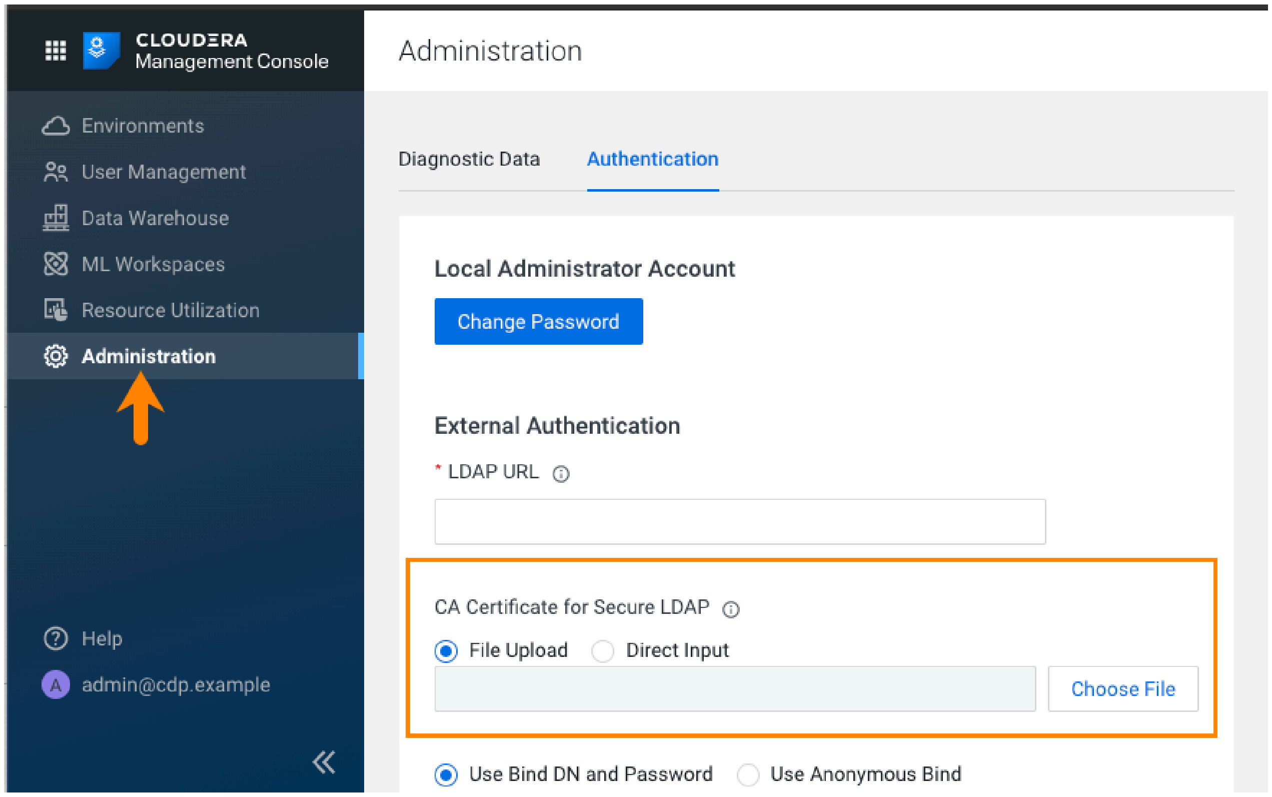
This topic describes security requirements needed to install and run Cloudera Data Warehouse (CDW) Private Cloud service on Red Hat OpenShift and Embedded Container Service (ECS) clusters.

Required OpenShift/ECS cluster permissions

The CDW service requires the "cluster-admin" role on the OpenShift and ECS cluster in order to install correctly. The "cluster-admin" role enables namespace creation and the use of the OpenShift Local Storage Operator for local storage.

CDP Private Cloud LDAP certificate requirement

A certificate authority (CA) certificate for secure LDAP must be uploaded to the Administration page of Management Console to run CDW Private Cloud service:



Base cluster database requirements for Cloudera Data Warehouse Private Cloud

You must be aware of the requirements for the database that is used for the Hive Metastore on the base cluster (Cloudera Manager side) for Cloudera Data Warehouse (CDW) Private Cloud.

CDW supports MariaDB, MySQL, PostgreSQL version 12 and 10, and Oracle databases for the Hive Metastore (HMS) on the base CDP cluster (Cloudera Manager side).

Note: CDW does not support PostgreSQL database version 11.

On a non-default Database Catalog, HMS, Hue, and DAS use an embedded or external PostgreSQL database that is defined when you install CDP Private Cloud. On a default Database Catalog, Hue and DAS use an embedded or external PostgreSQL database that is defined when you install CDP Private Cloud, and HMS can use either MariaDB, MySQL, Oracle, or PostgreSQL databases.

Important: CDW no longer requires permission to drop or create database schemas on your external Database Management Systems (DBMS). You must create schemas for DAS and Hue on your DBMS and configure it as a backend database by specifying the database name during environment activation.

Note: Cloudera recommends that you use an embedded database for the HMS and the Control Plane service. You can use the Data Recovery Service for backing up and restoring Kubernetes namespaces behind CDW entities (Database Catalogs and Virtual Warehosues).

If you are using PostgreSQL, MySQL, MariaDB, or Oracle database for the Hive Metastore on the base cluster, then it must meet the following requirements:

- SSL-enabled.
- Uses the same keystore containing an embedded certificate as Ranger and Atlas.

To use the same keystore with an embedded certificate for Ranger and Atlas:

- If you are using Auto-TLS:

  In the Management Console **Administration** page, go to the **CA Certificates** tab and select External Database from the CA Certificate Type drop-down menu. Upload the CA certificates either by uploading a file or by direct input.
- If you are not using Auto-TLS:

  Ensure that the public certificate of the certificate authority (CA) that signed the Hive metastore database's certificate is present in Cloudera Manager's JKS truststore. If the certificate is self-signed, import that certificate into Cloudera Manager's JKS truststore: In the Management Console Administration page, find the path to Cloudera Manager's JKS truststore by navigating to Administration Settings Security Cloudera Manager TLS/SSL Client Trust Store File . Import the CA's certificate into that JKS file.

  To add the certificate name to an existing or a new JKS file, use the following keytool command, which uses the same example certificate name:

  keytool -import -alias postgres -file /path/to/postgres.pem -storetype JKS   -keystore /path/to/cm.jks

  Where /path/to/cm.jks is the JKS file that is configured by Cloudera Manager.

  This ensures that the file specified for Cloudera Manager TLS/SSL Client Trust Store File is passed to Management Console and workloads.

  Note: If you have a JRE11 keystore you must convert it to a JRE8 keystore using the following keytool command:

```
keytool -importkeystore -srckeystore
             <PATH-TO-MY-PFX-FILE.PFX> -srcstoretype pkcs12 -srcstore
pass
                <***PASSWORD***>  -destkeystore
                <PATH-TO-CLIENT-CERTIFICATE.JKS> -deststoretype JKS
                -deststorepass <***PASSWORD***>
```

Creating your first Virtual Warehouse

This topic provides the high-level steps needed to create a Virtual Warehouse in Cloudera Data Warehouse (CDW) Private Cloud on both Red Hat OpenShift 4.6 clusters and Embedded Container Service (ECS) clusters.

Before you begin

Important: (On OpenShift environments) To activate an environment for the CDW service, someone with adequate permissions must use the Red Hat OpenShift Local Storage Operator to create a local file system on an SSD/NVMe for each OpenShift worker node and then mount it to a known location on the worker node. This creates space for local caching. The process is documented in Activating OpenShift and ECS environments.

On ECS clusters, CDW automatically creates the local file system. No additional steps are needed.

About this task

After you have registered an environment with Management Console, click Data Warehouse in the left menu to navigate to the Cloudera Data Warehouse (CDW) service. Then perform the following steps:

Procedure

1. Activate the environment to use with the CDW service. See Activating OpenShift and ECS environments.
2. Use the default Database Catalog that was created automatically when the environment was activated in Step 1, or create a new Database Catalog. See Adding a new Database Catalog.
3. After creating the Database Catalog, you can create a Virtual Warehouse. See Adding a new Virtual Warehouse.

Results

Now that you have a Virtual Warehouse, and you can start submitting workloads.

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.

Related Information

Configuring MySQL database to use SSL for Data Warehouse

Configuring MariaDB database to use SSL for Data Warehouse

Configuring Oracle database to use SSL for Data Warehouse

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[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 MySQL root certificate by running the following command:

```
/usr/java/default/bin/keytool -importcert -alias mysql -file /var/lib/my
sql/ca.pem -keystore $TRUSTSTORE_LOCATION -storetype jks -noprompt -stor
epass $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 propogates 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 propogates 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 -au
    to_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 "C
    N=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.pe
    m -keystore /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_trusts
    tore.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.