Cloudera Data Warehouse Private Cloud 1.5.0

CDW Private Cloud Administration

Date published: 2022-11-18
Date modified: 2023-01-25

https://docs.cloudera.com/

Legal Notice

Contents

Enabling Data Analytics Studio

Data Analytics Studio (DAS) has been deprecated in CDW Private Cloud 1.4.1 and higher and is disabled by default. Cloudera encourages you to use Hue for Hive and Impala workloads. However, if you still need DAS, you can enable it from the Advanced Configuration.

Procedure

1. Log in to the Data Warehouse service as an Administrator.
2. Click Advanced Configuration to go to the **Advanced Settings** page.
3. Select the Enable DAS option.
4. Click Update.

What to do next

Create a new Database Catalog and a Hive Virtual Warehouse.

Disabling Data Analytics Studio

Data Analytics Studio (DAS) in Data Warehouse Private Cloud is disabled by default. However, if you had enabled it and no longer need to use it, then you can disable DAS by deselecting the Enable DAS option from the Advanced Settings page.

About this task

Note: Disabling DAS from the **Advanced Settings** page does not affect the existing Database Catalogs and Virtual Warehouses. To remove and disable DAS from the Hive Virtual Warehouses, you must delete and recreate the Database Catalog and Virtual Warehouse.

Procedure

1. Log in to the Data Warehouse service as an Administrator.
2. Click Advanced Configuration to go to the **Advanced Settings** page.
3. Deselect the Enable DAS option.
4. Click Update.

What to do next

Delete and recreate a new Database Catalog and a Hive Virtual Warehouse.

Predefined Kerberos principals in Cloudera Data Warehouse 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@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 45 characters long and Database Catalog 53 characters long.

About the Refresh option

After changing settings and configurations, you often need to recreate Environments, Database Catalogs, and Virtual Warehouses. The Refresh option enables you to apply changes without the need to recreate them. Learn about the supported use cases in which you can use the Refresh option.

The Refresh option is available in the more options (⋮) menu at the Environment, Database Catalog, and Virtual Warehouse levels.

You might need to refresh the affected Database Catalogs and Virtual Warehouses when you add or update CA certificates for the LDAP server or update database settings such as host, port, database name, username, and password from the **Administration** page on Management Console. A refresh is also needed when you synchronize fresh configurations from the base cluster for the following components: Ozone, Hadoop, Hive, Ranger, and Atlas.

Note: You must refresh Database Catalogs first followed by Virtual Warehouses.

Supported object storage services for Cloudera Data Warehouse Private Cloud

HDFS is the default storage system for Cloudera Data Warehouse (CDW). However, you can enable CDW to access object storage such as AWS S3 and Azure Data Lake Storage (ADLS Gen1 and Gen2) if the CDP Private Cloud base cluster is configured to access it. You can query Hive and Impala tables stored on object stores using Hue.

Important: S3, S3-compatible, and ADLS object storage support is in technical preview and is not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.

When you activate an environment in CDW, all the hadoop configurations files (fs.s3a.*/fs.azure.*) are copied from the core-site.xml file present on the base cluster to the hadoop-core-site.xml file of the Hive and Impala metastore pods, enabling CDW to establish a connection to S3/ADLS.

Following are the key configurations that must be present in the base cluster core-site.xml file for connecting to S3 or S3-compatible storage providers:

- fs.s3a.access.key
- fs.s3a.secret.key
- fs.s3a.endpoint
- fs.s3a.connection.ssl.enabled

Following are the key configurations that must be present in the base cluster core-site.xml file for connecting to ADLS storage provider:

- fs.azure.account.oauth.provider.type
- fs.azure.account.oauth2.client.id
- fs.azure.account.oauth2.client.secret
- fs.azure.account.oauth2.client.endpoint

Important:

Because CDW uses all the base cluster configurations, it is important that you fine-tune and debug these configurations on the base cluster before creating the CDW environment.

If you have installed the Private Cloud Data Services, including CDW, before fine-tuning the base cluster configurations, then you must upload the Amazon/Azure server certificates referenced in the fs.s3a/ fs.azure endpoint configuration on the Management Console Administration CA Certificates tab. Select Miscellaneous as the certificate type from the CA Certificate Type drop-down menu.

The fs.s3a.*/fs.azure configurations are read-only. You can view these configurations from the CONFIGURATION tab on the Database Catalog and Virtual Warehouse details page by selecting the hadoop-core-site.xml option from the Configuration files drop-down menu.

Note: Disabling CDW's access to the third-party S3 providers from the **Advanced Settings** page does not affect the previously created Database Catalogs and Virtual Warehouses. To disable access, you must delete and recreate the Database Catalog and Virtual Warehouse.

Enabling S3 and S3-compatible storage providers in CDW

You can enable Cloudera Data Warehouse (CDW) data service on CDP Private Cloud to access S3 and S3-compatible object storage if the CDP Private Cloud base cluster is configured to access it.

About this task

Note: S3 and S3-compatible object storage support is in technical preview and is not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.

Before you begin

If you have installed the Private Cloud Data Services, including CDW, before fine-tuning the base cluster configurations, then you must upload the Amazon server certificates referenced in the fs.s3a endpoint configuration on the Management Console Administration CA Certificates tab. Select Miscellaneous as the certificate type from the CA Certificate Type drop-down menu.

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Go to Advanced Configurations Advanced Settings page.
3. Select the Enable S3 and S3-compatible object store providers option.
4. Click Update.

Important: If you have upgraded to CDW 1.5.0 and you have enabled the option to use S3, then you need to recreate the environment in CDW.

Enabling ADLS storage providers in CDW

You can enable Cloudera Data Warehouse (CDW) data service on CDP Private Cloud to access Azure Data Lake Storage (ADLS Gen1 and Gen2) object storage if the CDP Private Cloud base cluster is configured to access it.

About this task

Important: ADLS object storage support is in technical preview and is not recommended for production deployments. Cloudera recommends that you try this feature in test and development environments.

Before you begin

If you have installed the Private Cloud Data Services, including CDW, before fine-tuning the base cluster configurations, then you must upload the Azure server certificates referenced in the fs.azure endpoint configuration on the Management Console Administration CA Certificates tab. Select Miscellaneous as the certificate type from the CA Certificate Type drop-down menu.

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Go to the Advanced Configurations Advanced Settings page.
3. Select the Enable ADLS as a storage provider option.
4. Click Update.
5. Refresh the Database Catalog and Virtual Warehouses by clicking ⋮ Refresh on the Database Catalog and Virtual Warehouse tile.

Using Ozone storage with Cloudera Data Warehouse Private Cloud

Apache Ozone is an object store available on the CDP Private Cloud Base cluster which enables you to optimize storage for big data workloads. You can query data residing on Ozone using Hive or Impala from Cloudera Data Warehouse (CDW) Data Service on Private Cloud.

Note: Ozone support is in technical preview and is not recommended for production deployments. Cloudera recommends that you use Ozone with CDW in test and development environments.

Apache Ozone DataNodes support storage density up to 400 TB, unlike HDFS DataNodes which support storage density only up to 100 TB. Apart from the ability to scale to billions of objects or files of varying sizes, applications that use frameworks like Apache Spark, Impala, Apache YARN, and Apache Hive work natively on Ozone without any modifications.

Supported use cases

Ozone filesystem (OFS) is best suited for Hive and Impala in the following use cases:

- To retain HDFS IO performance and other characteristics critical for big data use cases.
- Recommended in an environment with dense nodes using up to 400 TB per node.
- To scale linearly and handle a large number of files and data.
- Recommended with Hadoop and S3 workloads.
- Recommended with native API, fast IO scans, streaming reads, and writes.
- Locality based on network topology (storage separate or together with compute).
- Object-level rename in a bucket.

Advantages

OFS offers the following operational advantages:

- Ability to share physical storage and nodes with HDFS.
- Designed for easy Node-addition, deletion, and decommission for repair.
- Has a security model similar to HDFS.
- Supports Kerberos authentication.
- Supports Data encryption at rest and in flight.
- Supports Ranger Authorization.

Unsupported features

Even though Ozone supports Erasure Coding starting with CDP 7.1.8, Hive and Impala do not support Ozone Erasure Coding in CDW Private Cloud 1.5.0 release.

Related Information

Blog: Apache Ozone and Dense Data Nodes

Setting up Ozone on the CDP Base cluster

To access and use Ozone from Cloudera Data Warehouse (CDW) data service on Private Cloud, you must add and configure the Ozone service on the base cluster.

About this task

Note: Ozone support is in technical preview in CDW 1.4.1 and 1.5.0. Cloudera recommends that you use Ozone with CDW in test and development environments. It is not recommended for production deployments.

Before you begin

Provision an Ozone cluster based on your desired storage capacity.

Procedure

1. Log in to Cloudera Manager as an Administrator.
2. Add and configure the Ozone service on the base cluster.
3. Enable Kerberos on the base cluster before you install the CDW data service.

   Enabling Kerberos on the base cluster automatically enables the Ozone service to use Kerberos. To verify this, go to Ozone service Configuration . The ozone.security.enabled parameter should be set to true and the hadoop.security.authentication parameter should be set to kerberos.
4. SSH into the Ozone host on the base cluster as an Administrator.
5. Obtain the tickets for the Hive or Impala user by using the Kerberos CLI kinit command.
6. Verify the Ozone Service ID for your cluster from the Configuration tab of the Ozone service in Cloudera Manager.

7. Verify that at least one volume and a bucket is available in Ozone by using the service ID you just verified. If a volume and a bucket does not exist, then run the following commands to create a volume in Ozone using the service ID:

```
ozone sh volume create --quota=[***VOLUME-CAPACITY***] --user=[***USERNA
ME***] URI
```

where,

- -q, --quota: Used to specify the maximum size that a volume can occupy in the cluster. This is an optional parameter.
- -u, --user: Used to specify the name of the user who can use the volume. The designated user can create buckets and keys inside the particular volume. This is a mandatory parameter.
- URI: Used to specify the name of the volume to be created. Specify the URI in the following format:

```
[***PREFIX***]://[***SERVICE-ID]/[***VOLUME-NAME***]
```

```
ozone sh volume create --quota=100GB --user=hrt_1 o3://vvslab/testvol
```

8. Create an encrypted or a non-encrypted bucket using the service ID that you just verified by running the following commands:

To create encrypted buckets:

```
ozone sh bucket create -k [***ENCRYPTION-KEY***] [***PREFIX***]://[***SE
RVICE-ID]/[***VOLUME-NAME***]/[***BUCKET-NAME***]
```

```
ozone sh bucket create -k key1 o3://vvslab/testvol/testbucketencrypted
```

Important: You must have the GET_METADATA and GENERATE_EEK permissions on the encryption key to create encrupted buckets on Ozone. The user who needs to read from the encrypted bucket must have the DECRYPT_EEK permission. These permissions are defined in the Ranger KMS policies on the base cluster.

To create non-encrypted buckets:

```
ozone sh bucket create [***PREFIX***]://[***SERVICE-ID]/[***VOLUME-NAME*
**]/[***BUCKET-NAME***]
```

```
ozone sh bucket create o3://vvslab/testvol/testbucket
```

22/08/10 10:25:10 INFO rpc.RpcClient: Creating Bucket: testvol/testbucket, with Versioning false and Storage Type set to DISK and Encryption set to false

9. Verify that the bucket is created by listing the bucket as follows:

```
ozone sh bucket list [***PREFIX***]://[***SERVICE-ID]/[***VOLUME-NAME***
] --length=[***NUMBER-OF-BUCKETS] --prefix=[***BUCKET-PREFIX] --start=[*
**STARTING-BUCKET***]
```

where,

- -l, --length: Used to specify the maximum number of results to return. The default is 100.
- -p, --prefix: Used to list the bucket names that match the specified prefix.
- -s, --start: Used to return results starting with the bucket after the specified value.

Note: All the existing buckets in Ozone are automatically available to query from Hive and Impala Virtual Warehouses in CDW.

To set Ozone as the default file system, you must configure OFS and add specific properties for the Ozone bucket you created.

What to do next

After setting up Ozone storage on the base cluster, configure CDW to use Hive or Impala to query data residing on the Apache Ozone object store.

Related Information

Enabling Kerberos Authentication for CDP

Kerberos configuration for Ozone

Commands for managing buckets

Managing storage elements by using the command-line interface

Setting up ofs

Configuring the Database Catalog to access the Ozone filesystem

After adding and configuring the Ozone service on the base cluster, creating buckets, and granting Ranger KMS policies to the users, you must configure the Hive MetaStore warehouse directories in the Database Catalog to point to the Ozone filesystem.

About this task

Note: Ozone support is in technical preview in CDW 1.4.1 and 1.5.0. Cloudera recommends that you use Ozone with CDW in test and development environments. It is not recommended for production deployments.

Before you begin

Note: If you have activated an environment in CDW before installing the Ozone service on the base cluster, then you must recreate the CDW environment so that Ozone configurations can be imported into CDW.

By default, the Hive MetaStore (HMS) for Database Catalogs on CDW Private Cloud points to HDFS.

- If you plan to make Ozone as the default FS, you must configure the Database Catalog to point to the Ozone storage system, as described in this topic.
- Alternatively, you can create a database with an Ozone bucket as the base directory so that all tables are created in that directory. Following is a sample command:

```
CREATE DATABASE ozone_db
[LOCATION ofs://ozone1/bucket1/ozone_db/external]
[MANAGEDLOCATION ofs://ozone1/bucket1/ozone_db/managed]
[WITH DBPROPERTIES (property_name=property_value, ...)];
```

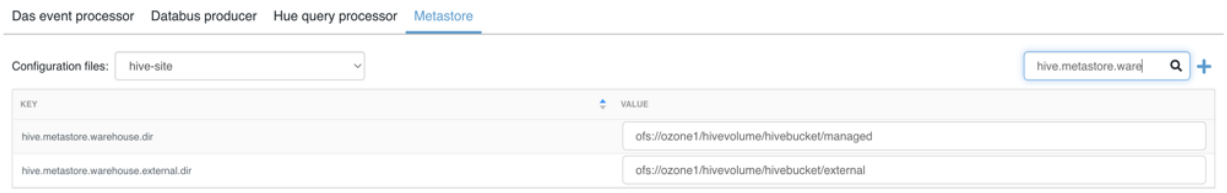
Before you re-configure the Database Catalog settings, make sure there are no running Virtual Warehouses associated with it. Either the Database Catalog has no associated Virtual Warehouses or you have suspended all the Virtual Warehouses associated with it.

Procedure

1. Log in to the Data Warehouse service as a DWAdmin.
2. Activate an environment in CDW.
3. In the Database Catalog tile, click ⋮ Edit CONFIGURATIONS Metastore and select hive-site from the Configuration files drop-down menu.
4. Search for the following configuration properties and update them to Ozone filesystem paths, which start with ofs:
   - hive.metastore.warehouse.dir
   - hive.metastore.warehouse.external.dir

   Note: For the Hive Table creation, the warehouse directory must be set at bucket level or directory level under the hive.metastore.warehouse.dir or hive.metastore.warehouse.external.dir parameters. For more information, see Changing the Hive warehouse location.

   Following is an example of these properties set for a Database Catalog:



   Note: The example values in the screenshot show the Hive warehouse locations in Ozone (set at a directory level) where Hive stores the tables. hivevolume represents the Ozone volume, hivebucket represents the Ozone bucket, and managed and external are directories where Hive stores the managed and external tables.
5. Click Apply Changes and wait for the Database Catalog to finish applying changes.

Results

After configuring the Database Catalog's Hive metastore to point to Ozone, create a Hive or an Impala Virtual Warehouse, or restart an existing Virtual Warehouse. You can then create databases and table using Hue or other SQL clients with your Virtual Warehouse.

Creating a Virtual Warehouse and creating tables on Ozone

After you configure the Database Catalog to point to the Ozone filesystem, verify that the Hive and Impala Virtual Warehouses in Cloudera Data Warehouse (CDW) carry the right configurations, and then you can managing databases and tables residing in Ozone using Hue or other SQL clients.

About this task

Note: Ozone support is in technical preview in CDW 1.4.1 and 1.5.0. Cloudera recommends that you use Ozone with CDW in test and development environments. It is not recommended for production deployments.

Before you begin

Ensure that the Hive MetaStore warehouse directories in the Database Catalog point to the Ozone filesystem on the CDP Private Cloud Base cluster.

Procedure

1. Log in to the Data Warehouse service as a DWAdmin.
2. Create an Impala or Hive Virtual Warehouse.

   Since you have already added Ozone in your base cluster, the required configuration are made available in CDW when you create a Virtual Warehouse.
3. Verify that the Ozone configurations are present in CDW. From your Virtual Warehouse tile, click ⋮ Edit CONFIGURATIONS Impala catalogd and select ozone-site from the Configuration files drop-down menu.

   For Hive, click ⋮ Edit CONFIGURATIONS Hiveserver2 and select ozone-site from the Configuration files drop-down menu.
4. Use Hue or any other SQL clients to start managing databases, managed and external tables.

   Following is a sample command to create an external table:

   ```
   create external table
   [***TABLE-NAME***] (id int, name string)
   location 'ofs://ozone1/s3v/cdw-logs/compute-schal-pvc111-env-1-hive5/ware
   house/tablespace/[***TABLE-NAME***]';
   ```
5. Verify that the required keys are created in the bucket by running the following command:

   ```
   ozone sh bucket ls [***VOLUME***] -p warehouses/tablespace
   ```

Related Information

Adding a new Virtual Warehouse

Configuring Hive/Impala logging on Ozone for Cloudera Data Warehouse Private Cloud

This section describes how to configure Cloudera Data Warehouse (CDW) on Private Cloud to store Hive and Impala logs on Ozone storage.

You can configure CDW to store Hive and Impala logs on CDP Private Cloud storage components, such as Ozone. Ozone is a good choice to store these logs because:

- Ozone efficiently handles files regardless of their size.
- In addition to Ozone's built-in CLI interface, Ozone also supports the HDFS CLI and CLIs that are compatible with AWS clients.
- CDP Private Cloud uses fluentd to push application logs to the storage layer. Ozone is a supported logging "back-end" component and has a fluentd-compatible endpoint for collecting the logs.

Note: Ozone support is in technical preview and is not recommended for production deployments. Cloudera recommends that you use Ozone with CDW in test and development environments.

Specify or create an Ozone bucket for Cloudera Data Warehouse Private Cloud logs

This topic describes how to specify an Ozone bucket to store Cloudera Data Warehouse (CDW) Private Cloud Hive and Impala logs.

About this task

You can either re-use the Ozone bucket that is automatically configured for storing Cloudera Machine Learning (CML) Private Cloud logs or create a new bucket to store CDW logs separately. The Ozone bucket used to store CML logs usually has a cdplogs- prefix.

Procedure

Use one of the following two methods depending on whether you want to use the existing CML log bucket or create a new one for CDW:

- To select an existing Ozone bucket, use the ozone sh bucket list command from the Ozone shell on your Private Cloud Base cluster. The following example shows how you can list buckets by the cdplogs- prefix:

```
ozone sh bucket list o3://ozone1/s3v --prefix=cdplogs
{
   "metadata" : { },
   "volumeName" : "s3v",
   "name" : "cdplogs-av-dwx-env-96c47aa9",
   "storageType" : "DISK",
   "versioning" : false,
   "creationTime" : "2020-08-01T18:29:08.686z",
   "modificationTime" : "2020-08-03T18:29:08.686z",
   "encryptionKeyName" : null,
   "sourceVolume" : null,
   "sourceBucket" : null
}
```

- To create a new bucket on Ozone, use the `ozone sh bucket create` command from the Ozone shell on your Private Cloud Base cluster. The following example shows how to create a new Ozone bucket named cdw-logs-bucket:

```
ozone sh bucket create o3://ozone1/s3v/cdw-logs-bucket
```

Important: Cloudera recommends that you use the hive user because this user automatically has create/read/write permissions on buckets that you create.

Update Cloudera Data Warehouse Private Cloud log configuration to point to Ozone

This topic describes how to configure Cloudera Data Warehouse (CDW) Private Cloud to store logs on Ozone.

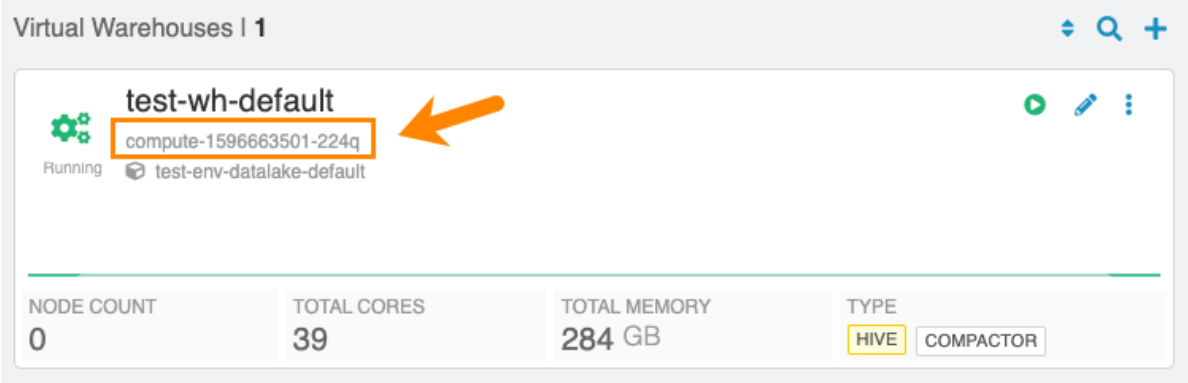
About this task

To configure CDW Private Cloud and the underlying OpenShift cluster to store Hive and Impala logs on Ozone, you must gather some information and prepare a block of code that you will insert into the Virtual Warehouse ConfigMap on the OpenShift pod. These preliminary steps are described in the following section.

Before you begin

Get the following information and prepare the block of code for the Virtual Warehouse ConfigMap before you start the steps of updating the configuration:

- Get the CDW namespace for your Virtual Warehouse:
  1. From the Management Console home page left menu, click Data Warehouse in the left menu. You are taken to the Overview page of CDW Private Cloud service.
  2. Locate the Virtual Warehouse you want to configure log storage for in the right-most column of the page, and locate the CDW namespace, which starts with compute- as shown below:



- Prepare the code block that must be pasted into the OpenShift ConfigMap:

  Here is an example:

```
<match **>
       @type s3
       @log_level debug
       aws_key_id <access-id>
       aws_sec_key <sec-key>
       s3_bucket <bucket-name>
       s3_endpoint <ozone-s3-gateway-endpoint>
       ssl_verify_peer false
       s3_object_key_format
       "<warehouse_prefix>/warehouse/tablespace/external/hive/sys.db/logs
/dt=%Y-%m-%d/${path_tag}/%{time_slice}_${unique_file_key}.log.%{file_ext
ension}"
       time_slice_format %Y-%m-%d-%H-%M
       store_as gzip
       auto_create_bucket false
       check_apikey_on_start false
       force_path_style true
       check_bucket false
       check_object false
       <buffer path_tag, unique_file_key, time, warehouse>
        @type file
        path /tmp/fluentd-buffers/%{unique_file_key}-s3.buffer
        timekey 900 # minute precision for time_slice_format to have minu
te in file name
        timekey_use_utc true
        chunk_limit_size 265m
        flush_mode interval
        flush_interval "900s"
        flush_thread_count 8
        flush_at_shutdown true
       </buffer>
       <format>
        @type single_value
        message_key log
        add_newline true
       </format>
```

```
      </match>
```

In the above code block example:

- <*bucket-name*> indicates the name of the Ozone bucket used for storing the CDW Private Cloud logs.
- <*ozone-s3-gateway-endpoint*> indicates the endpoint of the Ozone S3 Gateway. Get this value from the Ozone S3 Gateway Web UI page of Cloudera Manager.
- <*access_id*> and <*sec_key*> are the AWS access credentials for the Ozone S3 Gateway. Get these values by using the kinit -kt and the ozone s3 getsecre commands on the Private Cloud Base OpenShift cluster.

Procedure

1. Using OpenShift commands, view the OpenShift project for the pod where the CDW Private Cloud instance is running by specifying the CDW namespace for the Virtual Warehouse that you noted in the Before you begin section above.

   For example, if the CDW namespace is compute-1596663501-224q, you can view the OpenShift project with the following command:

   ```
   oc project compute-1596663501-224q
   ```

2. Open the ConfigMap for the Virtual Warehouse that is associated with the CDW namespace. For example:

   ```
   oc edit configmap warehouse-fluentd-config
   ```

   This command opens the ConfigMap in a separate editor that is similar to vi.
3. Replace the match section of the ConfigMap with the code block you prepared in the Before you begin section above, and then save your changes
4. Verify that the new configuration is correctly updated by running the following command:

   ```
   oc get namespace -o yaml | grep fluentd-status
   ```

   If the configuration is successfully updated, the value of the fluentd-status returns an empty string as shown in the following example:

   ```
   com.cloudera/fluentd-status: ""
         com.cloudera/fluentd-status: ""
         com.cloudera/fluentd-status: ""
         com.cloudera/fluentd-status: ""
   ```

Monitor Cloudera Data Warehouse Private Cloud logs on Ozone storage

This topic describes how to monitor Cloudera Data Warehouse (CDW) Private Cloud logs that are stored on Ozone.

About this task

You can use either the Ozone S3 Gateway Web UI in Cloudera Manager or run commands in a terminal window to monitor CDW logs.

Note: Because fluentd buffers the logs and then pushes them to the configured endpoint, Ozone might take up to 15 minutes to display the CDW logs.

Procedure

Use one of the following methods to monitor CDW logs in Ozone:

- Ozone S3 Gateway Web UI in Cloudera Manager:

  Navigate to the following URL:

  https://<*s3-gateway-endpoint*>/<*bucket-name*>?browser=true

  Where:

  - <*s3-gateway-endpoint*> indicates the endpoint of the Ozone S3 Gateway, which you can get from the Ozone S3 Gateway Web UI
  - <*bucket-name*> indicates the Ozone bucket where you are storing the CDW logs.
- Run the following command from the Ozone shell: ozone sh key list o3://<*ozone.service.id*>/s3v/<*bucket-name*>/ --prefix=<*warehouse-prefix*>

  Where:

  - <*ozone.service.id*> indicates the identifier used for your implementation of Ozone.
  - <*bucket-name*> indicates the name of the Ozone bucket where the CDW logs are stored.
  - <*warehouse-prefix*> indicates the Virtual Warehouse identifier.

Analyze Cloudera Data Warehouse Private Cloud logs stored on Ozone

This topic describes how to use Hue or Data Analytics Studio (DAS) to analyze Cloudera Data Warehouse (CDW) Private Cloud logs that are stored on Ozone.

About this task

You can use Hue to analyze Impala logs or DAS to analyze Hive logs.

Note:

You must use the Hue or DAS instance that corresponds to the Virtual Warehouse whose logs are saved on Ozone. To ensure that you use the correct instance, access Hue or DAS by using the drop-down menu in the upper right corner of the Virtual Warehouse tile.

Procedure

1. Using Hue or DAS, create an external table that points to the log data on Ozone:

   CREATE EXTERNAL TABLE <*table-name*> LIKE sys.logs LOCATION 'ofs://<*ozone.service.id*>/s3v/<*bucket-name*>/<*warehouse-prefix*>/warehouse/tablespace/external/hive/sys.db/logs';
2. Run the MSCK REPAIR TABLE command on the table you created in Step 1:

   MSCK REPAIR TABLE <*table-name*>;

Results

After completing the above steps, you can use SQL queries to analyze the log data.

Changing delegation username and password

You specify the delegation username and password while activating an environment. You can change the delegation username or password from the Environment Details page.

Procedure

1. Log in to the Data Warehouse service as a DWAdmin.
2. Go to Environments ⋮ Edit CONFIGURATIONS .

3. Enter a new Delegation Username and/or Delegation Password.

Note:

- The delegation user and the LDAP Bind user configured on the **Administration** page of the Management Console are not necessarily the same user.
- The special characters used in the LDAP Bind user password are not exactly the same as the ones that can be used in the delegation user password, because only the following characters are supported to be used in the LDAP Bind user password: ! # $ % ( ) * + , - . / : ; = ? @ [ ] ^ _ ` { | } ~.
- The following special characters are not supported to be used in the name of the delegation user or in the Distinguished Name of the LDAP Bind user: < > & ' ".

4. Click Apply Changes.

Enabling the option to create additional Database Catalogs in CDW Private Cloud

Starting with the Private Cloud 1.5.0 release, you cannot create additional non-default Database Catalogs without enabling the “Create multiple Database Catalogs” option on the Advanced Settings page.

About this task

In Cloudera Data Warehouse (CDW) Private Cloud, a default Database Catalog is created when you activate an environment and is specific to that environment. The default Database Catalog uses a CDW-managed database. If you create additional Database Catalogs, then non-default Database Catalogs are created which require a separate, external database that CDW does not manage.

Cloudera encourages you to use the default Database Catalogs because the ability to create and use non-default Database Catalogs will be removed in future releases. To enable creating non-default Database Catalogs:

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Go to the Advanced Configuration Advanced Settings page.
3. Select the Create multiple Database Catalogs option.
4. Click Update.

SSL-enabled endpoints for Virtual Warehouse clients in Cloudera Data Warehouse Private Cloud

In Cloudera Data Warehouse (CDW) Private Cloud 1.1, all client endpoints have been SSL-enabled. This requires that you configure the SSL certificates for client endpoints.

In CDW Private Cloud 1.1 and higher, 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
- Data Analytics Studio (DAS) webapp
- 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.myfi
rm.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;transportMo
de=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.

Disable the SSL or TLS requirement for HMS database

Cloudera Data Warehouse (CDW) on Private Cloud requires that you enable SSL or TLS on the database that you use for the Hive MetaStore (HMS) on the base cluster. However, if you have not configured SSL or TLS on the HMS database and you want to continue using CDW, then you must disable the SSL requirement, so that the Database Catalog does not fail to start in an attempt to establish a secure connection with an unsecured database.

About this task

The disable-sslmode.sh script used to disable CDW's requirement to have an SSL-enabled database is designed to work on Machintosh OS.

Before you begin

You must have the DWAdmin role and must have activated an environment in CDW.

Procedure

1. Log in to the Data Warehouse service as DWAdmin.
2. Obtain the namespace of the Database Catalog from the Database Catalog tile.

   The Database Catalog namespace is on the line under the Environment name.
3. Open a terminal session on your computer.
4. Install “yq” and “kubectl” command-line tools.
5. Create a work directory.

   You use this directory to create and save the disable-sslmode.sh file, which you create in the next step.
6. Create the disable-sslmode.sh file on your computer by copying the following lines:

```
#!/bin/bash
set -euo pipefail
#set -euvx


SHORT=k:,n:,c:,u,a,b:,w:,h

>&2 echo "OS-Type: ${OSTYPE}"

if [[ "${OSTYPE}" == "darwin"* ]]; then
  >&2 echo "Using only short options in getopt"
  OPTS=$(getopt $SHORT "$@")
#elif [[ "${OSTYPE}" == "linux-gnu"* ]]; then
elif [[ "${OSTYPE}" == "linux-"* ]]; then
  >&2 echo "Using both short and long options in getopt"
  LONG=kubeconfig:,namespace:,config-map:,clean-up,apply,backup:,working-d
irectory:,help
  OPTS=$(getopt -a -n generate --options $SHORT --longoptions $LONG -- "$
@")
else
  >&2 echo "Unsupported OS type: ${OSTYPE}"
  exit 249
fi

eval set -- "${OPTS}"

usage() {
```

```
  >&2 echo <<EOF "Updates the SSL mode of javax.jdo.option.ConnectionURL
 in hive-site.xml of the specified config map in specified namespace.

Please take attention that during the update the kube config file is cop
ied into a temporary directory in /tmp

Prerequisites:
    - kubectl >= 1.23
    - yq >= 4.x

Usage: $0 -k|--kubeconfig <path-to-kube-config-file> -n|--namespace <k8s-n
amespace> [-s|--ssl-mode <ssl-mode-to-be-set>] [-c|--config-map <name-of-
config-map-to-update>]
Mandatory parameters:
    -k|--kubeconfig <path-to-a-kubeconfig-yaml-file>
        the location of kube config file to use
    -n|--namespace <namespace-name>
        the kubernetes namespace to edit

Optional parameters:
    -c|--config-map <config-map-name-to-update>
        the name of config map to update
        default: hive-conf-metastore

    -w|--working-directory <a-directory>
        a directory where temporary created files are stored
        if not specified then something is created under /tmp using com
mand mktemp -d

    -u|--clean-up
        deletes the resources created during update

    -a|--apply
        applies the changes
        if not specified it only generates the new config map file but is
not applying that.

    -b|--backup <backup-file-name>
        creates a backup in tar.gz (aka tgz) format.
        it will contain all files in working-directory

    -h|--help
        this description
"
EOF
}

# KUBECONFIG="shared-os-dev-01.yaml"
# NAMESPACE="compute-sj-230509t1723-hive"
CONFIG_MAP_NAME="hive-conf-metastore"
# BACKUP_FILE_NAME="/dev/null"
APPLY="false"
CLEAN_UP="false"
echo ${OPTS}

while :
do
  case "$1" in
    -k | --kubeconfig )
      KUBECONFIG="$2"
      shift 2
      ;;
    -n | --namespace )
      NAMESPACE="$2"
```

```
      shift 2
      ;;
    -c | --config-map )
      CONFIG_MAP_NAME="$2"
      shift 2
      ;;
    -b | --backup )
      BACKUP_FILE_NAME="$2"
      >&2 echo "Creating backup at the end: ${BACKUP_FILE_NAME}"
      shift 2
      ;;
    -w | --working-directory )
      WD="$2"
      shift 2
      ;;
    -a | --apply )
      APPLY="true"
      >&2 echo "Applying changes at the end"
      shift 1
      ;;
    -u | --clean-up )
      CLEAN_UP="true"
      >&2 echo "Cleaning up at the end"
      shift 1
      ;;
    -h | --help )
      usage
      exit 2
      ;;
    --)
      shift
      break
      ;;
    *)
      >&2 echo "unexpected option: $1"
      ;;
  esac
done
which yq >/dev/null || (echo "yq not found"; exit 251)
YQ_CURRENT_VERSION=`yq --version | awk '{ print $4; }' | cut -c 2-`
YQ_REQUIRED_VERSION=4.18.1

if [ "$(printf '%s\n' "${YQ_REQUIRED_VERSION}" "${YQ_CURRENT_VERSION}" |
 sort -V | head -n1)" = "${YQ_REQUIRED_VERSION}" ]; then
  >&2 echo "version of yq (${YQ_CURRENT_VERSION}) is greater than or eq
ual to ${YQ_REQUIRED_VERSION} ... OK"
else
  >&2 echo "version of 'yq' must be greater than or equal to '${YQ_REQUIR
ED_VERSION}' but it is '${YQ_CURRENT_VERSION}' ... FAILED"
  exit 248
fi

which kubectl
which kubectl >/dev/null || (echo "kubectl not found"; exit 252)

if [[ -z "${KUBECONFIG+x}" ]]; then
  >&2 echo "ERROR: missing kubeconfig use  -k or --kubeconfig"
 usage
 exit 254
fi

if [[ ! -f ${KUBECONFIG} ]]; then
  >&2 echo "ERROR: kubeconfig file does not exist: ${KUBECONFIG}"
  exit 246
```

```
fi
KUBECTL_CLIENT_CURRENT_VERSION="$(kubectl --kubeconfig ${KUBECONFIG} ver
sion -o yaml | yq '.clientVersion | [.major,.minor] | join(".")')"
KUBECTL_CLIENT_REQUIRED_VERSION=1.23
if [ "$(printf '%s\n' "${KUBECTL_CLIENT_REQUIRED_VERSION}" "${KUBECTL_C
LIENT_CURRENT_VERSION}" | sort -V | head -n1)" = "${KUBECTL_CLIENT_REQUI
RED_VERSION}" ]; then
  >&2 echo "version of kubectl (${KUBECTL_CLIENT_CURRENT_VERSION}) is grea
ter than or equal to ${KUBECTL_CLIENT_REQUIRED_VERSION} ... OK"
else
  >&2 echo "version of 'kubectl' must be greater than or equal to '${KU
BECTL_CLIENT_REQUIRED_VERSION}' but it is '${KUBECTL_CLIENT_CURRENT_VERS
ION}' ... FAILED"
  exit 247
fi


if [[ -z "${NAMESPACE+x}" ]]; then
  >&2 echo "ERROR: missing namespace use  -n or --namespace"
 usage
 exit 253
fi

if [[ -z "${WD+x}" ]]; then
 WD=$(mktemp -d)
fi
if [[ -z ${BACKUP_FILE_NAME+x} ]]; then
 BACKUP_FILE_NAME="backup.${NAMESPACE}.${CONFIG_MAP_NAME}.$(date -u +%Y%m
%dT%H%M%SZ).tar.gz"
fi


[[ -d "${WD}" ]] || mkdir -p "${WD}"
cat <<EOF > "${WD}/logs"
Working directory: ${WD}
kubeconfig: ${KUBECONFIG}
namespace: ${NAMESPACE}
config-map: ${CONFIG_MAP_NAME}
apply: ${APPLY}
backup: ${BACKUP_FILE_NAME}
clean-up: ${CLEAN_UP}
EOF

>&2 cat "${WD}/logs"
LOCAL_KUBECONFIG="$(basename "${KUBECONFIG}")"
cp "${KUBECONFIG}" "${WD}/${LOCAL_KUBECONFIG}"

pushd "${WD}"

ORIGINAL_CM_NAME="original.configmap.${CONFIG_MAP_NAME}.yaml"
ORIGINAL_HIVE_SITE_XML="${ORIGINAL_CM_NAME}.hive-site.xml"
UPDATED_CM_NAME="updated.configmap.${CONFIG_MAP_NAME}.yaml"
export UPDATED_HIVE_SITE_XML="${UPDATED_CM_NAME}.hive-site.xml"
env | grep -i path
>&2 echo "kubectl --kubeconfig \"${LOCAL_KUBECONFIG}\" get configmap \"$
{CONFIG_MAP_NAME}\" -n \"${NAMESPACE}\" -o yaml > \"${ORIGINAL_CM_NAME}\""
kubectl --kubeconfig "${LOCAL_KUBECONFIG}" get configmap "${CONFIG_MAP_NA
ME}" -n "${NAMESPACE}" -o yaml > "${ORIGINAL_CM_NAME}"
yq '.data."hive-site.xml"' > "${ORIGINAL_HIVE_SITE_XML}" < "${ORIGINAL_CM
_NAME}"
disableSslModeInPostgreSql() {
 local CONNECTION_URL="${1//sslmode=[^&]*/sslmode=disable}"
 CONNECTION_URL="${CONNECTION_URL//sslrootcert=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//&&&*/&}"
```

```
 if [[ "${CONNECTION_URL:0-1}" == "&" ]]; then
  CONNECTION_URL="${CONNECTION_URL::-1}"
 fi
  if [[ "${CONNECTION_URL:0-1}" == "?" ]]; then
   CONNECTION_URL="${CONNECTION_URL::-1}"
 fi
 echo "${CONNECTION_URL}"
}

disableSslModeInMySql() {
 local CONNECTION_URL="${1//sslMode=[^&]*/sslMode=PREFERRED}"
 CONNECTION_URL="${CONNECTION_URL//trustCertificateKeyStoreUrl=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//trustCertificateKeyStorePassword=[^&]*
/}"
 CONNECTION_URL="${CONNECTION_URL//&&&*/&}"
 if [[ "${CONNECTION_URL:0-1}" == "&" ]]; then
  CONNECTION_URL="${CONNECTION_URL::-1}"
 fi
  if [[ "${CONNECTION_URL:0-1}" == "?" ]]; then
   CONNECTION_URL="${CONNECTION_URL::-1}"
 fi
 echo "${CONNECTION_URL}"
}

disableSslModeInOracle() {
  local CONNECTION_URL="${1//javax.net.ssl.trustStore=[^&]*/}"
  CONNECTION_URL="${CONNECTION_URL//@tcps:/@tcp:}"
 CONNECTION_URL="${CONNECTION_URL//javax.net.ssl.trustStoreType=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//javax.net.ssl.trustStorePassword=[^&
]*/}"
 CONNECTION_URL="${CONNECTION_URL//javax.net.ssl.keyStore=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//javax.net.ssl.keyStoreType=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//javax.net.ssl.keyStorePassword=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//oracle.net.ssl_cipher_suites=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//oracle.net.ssl_server_dn_match=[^&]*/}"
 CONNECTION_URL="${CONNECTION_URL//oracle.net.authentication_services=[^&
]*/}"
 CONNECTION_URL="${CONNECTION_URL//&&&*/&}"
  if [[ "${CONNECTION_URL:0-1}" == "&" ]]; then
   CONNECTION_URL="${CONNECTION_URL::-1}"
 fi
  if [[ "${CONNECTION_URL:0-1}" == "?" ]]; then
   CONNECTION_URL="${CONNECTION_URL::-1}"
 fi
 echo "${CONNECTION_URL}"
}

ORIGINAL_CONNECTION_URL=$(yq -p=xml '.Configuration.property[]|select(.na
me=="javax.jdo.option.ConnectionURL")|.value' < "${ORIGINAL_HIVE_SITE_XM
L}");
if [[ -n ${ORIGINAL_CONNECTION_URL} ]]; then
 >&2 echo "javax.jso.option.ConnectioURL=${ORIGINAL_CONNECTION_URL}";
 echo "javax.jso.option.ConnectioURL=${ORIGINAL_CONNECTION_URL}" >> logs;

 IFS=':' read -a token <<< "${ORIGINAL_CONNECTION_URL}"
 DRIVER="${token[1]}"
 >&2 echo "JDBC-Driver: ${DRIVER}"
 echo "Driver: ${DRIVER}" >> logs

 case "${DRIVER}" in
  postgres | postgresql )
   export NEW_CONNECTION_URL=$(disableSslModeInPostgreSql "${ORIGINAL_CON
NECTION_URL}")
   ;;
```

```
  mysql | mariadb )
   export NEW_CONNECTION_URL=$(disableSslModeInMySql "${ORIGINAL_CONNEC
TION_URL}")
   ;;
  oracle )
   export NEW_CONNECTION_URL=$(disableSslModeInOracle "${ORIGINAL_CONNE
CTION_URL}")
   ;;
  * )
   >&2 echo "Unsupported JDBC-Driver: ${DRIVER}"
   exit 250
   ;;
 esac


 >&2 echo "New ConnectionURL: ${NEW_CONNECTION_URL}";
 echo "New ConnectionURL: ${NEW_CONNECTION_URL}" >> logs;
 yq -p=xml '.Configuration.property|=map(select(.name=="javax.jdo.option
.ConnectionURL").value=env(NEW_CONNECTION_URL))' -o=xml > "${UPDATED_HIV
E_SITE_XML}" 2>> logs < "${ORIGINAL_HIVE_SITE_XML}"
 kubectl --kubeconfig "${LOCAL_KUBECONFIG}" get configmap "${CONFIG_MAP
_NAME}" -n "${NAMESPACE}" -o yaml | yq '.data."hive-site.xml" |= load_st
r(env(UPDATED_HIVE_SITE_XML))' > "${UPDATED_CM_NAME}" 2>>logs

 if [[ ${APPLY} == "true" ]]; then
  >&2 echo "Applying: kubectl --kubeconfig \"${LOCAL_KUBECONFIG}\" apply -
f ${UPDATED_CM_NAME}"
  echo "Applying: kubectl --kubeconfig \"${LOCAL_KUBECONFIG}\" apply -f ${
UPDATED_CM_NAME}" >> logs
  kubectl --kubeconfig "${LOCAL_KUBECONFIG}" apply -f "${UPDATED_CM_NAME}"
 >> logs 2>&1
 fi

fi;

popd

if [[ -n ${BACKUP_FILE_NAME+x} ]]; then
 >&2 echo "Creating backup from working-directory: ${BACKUP_FILE_NAME}"
 echo "Creating backup from working-directory: ${BACKUP_FILE_NAME}" >> "${
WD}/logs"
 tar czf "${BACKUP_FILE_NAME}" -C "${WD}" .
fi

if [[ "${CLEAN_UP}" == "true" ]]; then
 >&2 echo "Clean-up working-directory: ${WD}"
 rm -rf "${WD}"
fi
```

7. Run the disable-sslmode.sh script as follows:

```
./disable-sslmode.sh -k [***KUBECONFIG-FILENAME***].yaml -n [***DBC-NAME
SPACE***] -c hive-conf-metastore -a -w [***WORK-DIRECTORY***]
```

Replace [***KUBECONFIG-FILENAME***] with the actual kubeconfig yaml file and [***DBC-NAMESPACE***] with the Database Catalog namespace that you noted earlier.

Note: On MacOS, use short option flags in the command, such as “-n” instead of “--namespace”.

Sample output:

```
Working directory: /tmp/tmp.vQnpYUrIq5
kubeconfig: [***KUBECONFIG-FILENAME***].yaml
namespace: [***DBC-NAMESPACE***]
config-map: hive-conf-metastore
apply: true
backup: backup.[***DBC-NAMESPACE***].hive-conf-metastore.20230517T13195
5Z.tar.gz
clean-up: false
.
.
.
JDBC-Driver: postgresql
New ConnectionURL: jdbc:postgresql://postgres-service-default-warehouse:
5432/hive1?createDatabaseIfNotExist=true&sslmode=disable
Applying: kubectl --kubeconfig "[***KUBECONFIG-FILENAME***]" apply -f u
pdated.configmap.hive-conf-metastore.yaml
```

8. Note the “New ConnectionURL” from the logs of the disable-sslmode.sh script.
9. If the Database Catalog does not start automatically after the script completes running, then restart the metastore-0 and metastore-1 pods as follows:

```
kubectl --kubeconfig [***KUBECONFIG-FILENAME***].yaml delete pod metasto
re-0 -n [***DBC-NAMESPACE***]
```

Wait for the pods to restart and be in the running state.

10. Go to Database Catalog ⋮ Edit CONFIGURATIONS Metastore and select hive-site from the Configuration files drop-down menu.
11. Add the value of the New ConnectionURL parameter in the javax.jdo.option.ConnectionURL field.

    This ensures that a correct connection URL is delegated to the Virtual Warehouses from the Database Catalog.
12. Click Apply Changes.

Debugging Impala Virtual Warehouses using Web UIs

You can use the Catalog Web UI, Coordinator Web UI, and the StateStore Web UI to debug Impala Virtual Warehouses in Cloudera Data Warehouse (CDW).

About this task

The Impala daemons (impalad, statestored, and catalogd) debug Web UIs, which can be used in CDP Runtime by using Cloudera Manager, is also available in the CDW service. In CDW service, the following Web UIs are provided:

- Impala Catalog Web UI

  This UI provides the same type of information as the Catalog Server Web UI in Cloudera Manager. It includes information about the objects managed by the Impala Virtual Warehouse. For more information about this debug Web UI, see .
- Impala Coordinator Web UI

  This UI provides the same type of information as the Impala Daemon Web UI in Cloudera Manager. It includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries. For information about this debug Web UI, see .
- Impala StateStore Web UI

  This UI provides the same type of information as the StateStore Web UI in Cloudera Manager. It includes information about memory usage, configuration settings, and ongoing health checks that are performed by the Impala statestored daemon. For information about this debug Web UI, see
- Impala Autoscaler Web UI

  This UI gives you insight into Autoscaler operations, accessing log messages, and resetting the log level. The autoscaler Web UI includes information about the queries queued and running, executor groups, suspended calls, scale up/down calls, the autoscaler config, and the autoscaler logs.

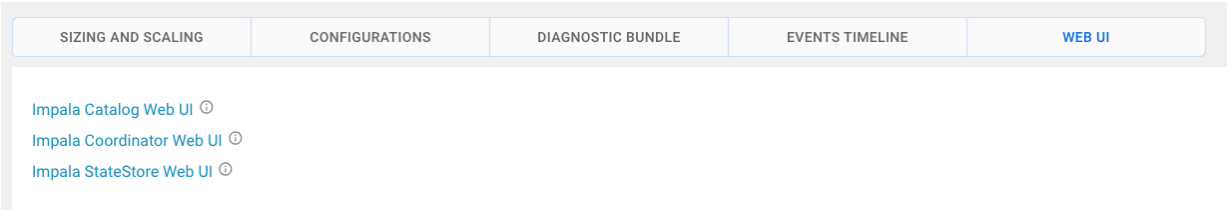
Required role: EnvironmentAdmin

Before you begin

Make sure that you note your CDP workload user name and have set a password for it in the User Management module of Management Console. You need to use your workload user name and its associated password to log into the debug Web UIs.

Procedure

1. In the CDW UI on the Overview page, locate the Impala Virtual Warehouse for which you want to view the debug UIs, and select Edit from the options menu on the tile. This launches the details page for this Virtual Warehouse.
2. In the **Virtual Warehouse** details page, select the WEB UI tab on the right. The list of debug Web UI links are displayed as shown in the following image:



3. Click a Web UI link corresponding to an Impala daemon that you want to debug.

   You are prompted to enter your workload user name and password.

Results

After you are authenticated, you can view the debug Web UI and use the information to help you troubleshoot issues with your Impala Virtual Warehouse.

Generating and downloading diagnostic bundles

Cloudera Data Warehouse (CDW) collects diagnostic data on workload logs, such as Impala Coordinator, Statefulset, CatalogD logs and stores it in the tmp directory on HDFS. You can download the logs using the Hue File Browser from the base cluster.

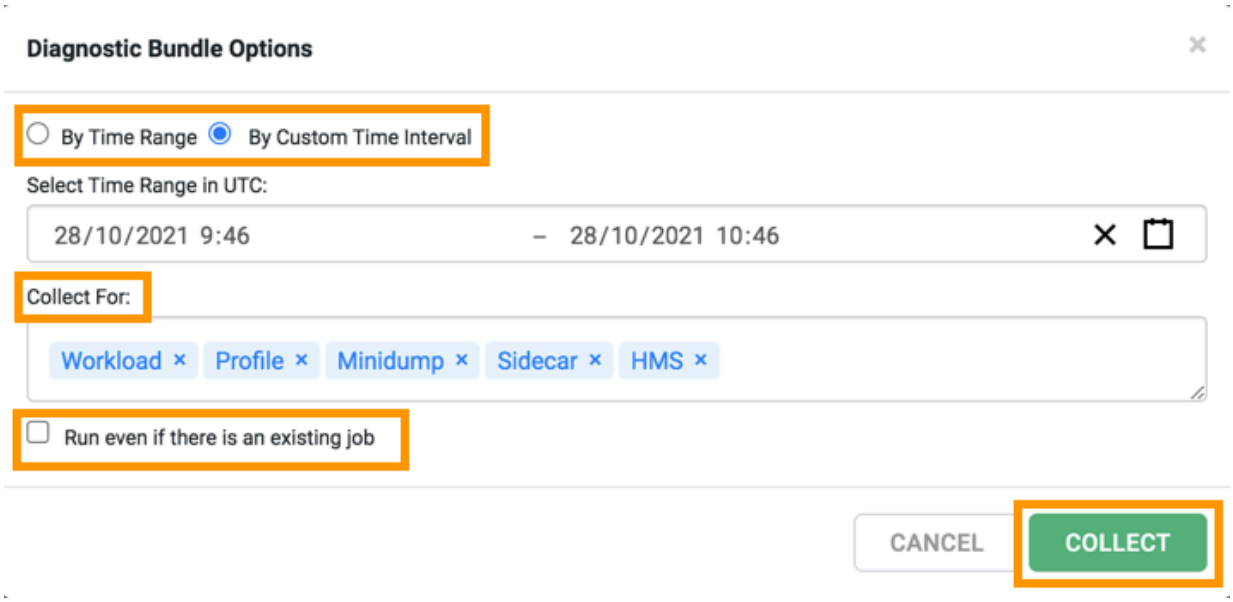
About this task

During the lifetime of a cluster, logs are continously written to the following directory on HDFS: [**WAREHOUSE-DIR**]/warehouse/tablespace/external/hive/sys.db/. When you click Collect Diagnostic Bundle from the CDW web interface, CDW collects the logs for the specified time interval and for the services that you select. These logs are compressed in a ZIP file format and stored in the tmp directory.

Attention: In 1.3.2 release of CDW Private Cloud, you can generate and download diagnostic bundles only for Impala.

Procedure

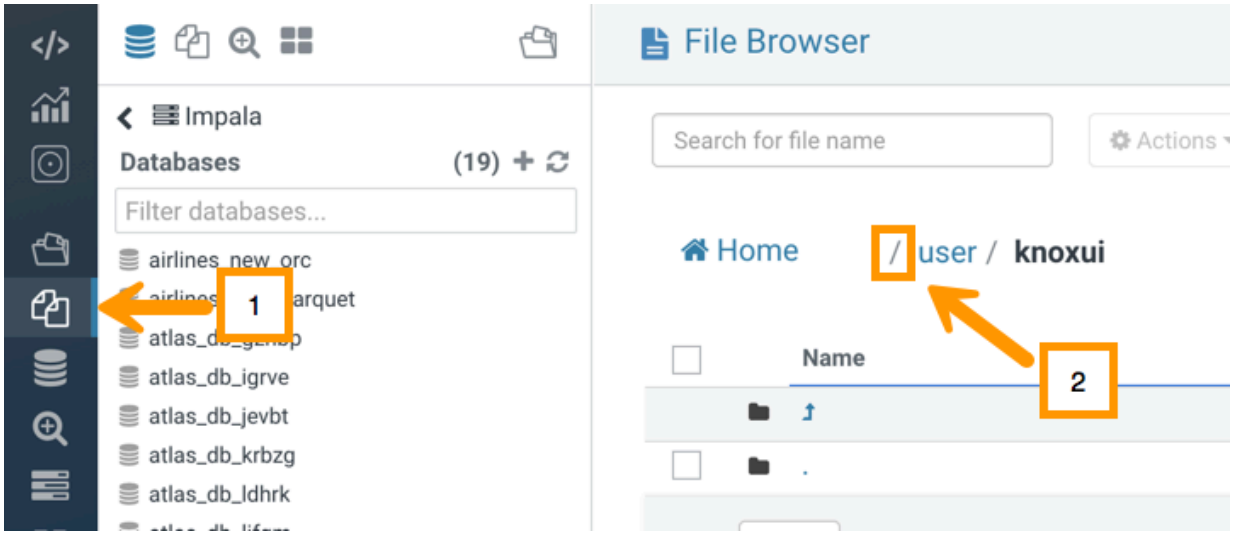
1. Log in to the CDW service as a DWAdmin.
2. Click the options drop-down menu on the Virtual Warehouse for which you want to collect the logs and click Collect Diagnostic Bundle.
3. On the **Diagnostic Bundle Options** dialog box, select the time interval and the type of logs you want to collect and click COLLECT.



4. To view the status of the job and to obtain the HDFS location where the logs are stored, select Edit from the Virtual Warehouse options menu and go to the DIAGNOSTIC BUNDLE tab.

   The logs are collected and bundled under the /tmp/[***VIRTUAL-WAREHOUSE-ID-TIMESTAMP***].zip directory.
5. To access and download the logs, open the Hue service from the base cluster.

6. Go to the Hue File Browser and click the forward slash (/) before the user directory as shown in the following image:



The tmp directory is displayed. You can access and download the logs to your computer by clicking Download.

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*.

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 Cloudera Data Warehouse 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 Cloudera 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.

Creating custom pod configurations for Impala Virtual Warehouses

You can configure the resources used by Impala Virtual Warehouses in Cloudera Data Warehouse (CDW) Private Cloud environments to optimize Impala performance or to control resource usage in the environment.

About this task

When you create a Virtual Warehouse, CDW allocates standard resources to the Warehouses that are suitable for most workloads. You can control the size of the Virtual Warehouse at the time of creation by choosing the number of

nodes to be used. By using custom pod configurations, you can also change the resources used by the critical Impala components, such as the coordinators, executors, and catalog daemons to pack a particular number of pods into a Kubernetes node or to create extra-large daemons to handle specific workloads.

Procedure

1. Log in to the Data Warehouse service as a DWAdmin.
2. Go to your environment and click ⋮ Edit .

   The **Environment Details** page is displayed.
3. Click the EDIT POD CONFIGURATIONS tab.

   A pod configuration is a named resource that is configured at the environment level.
4. Select one of the following two pod configuration options from the Select Pod Configuration section:
   - The Cdw Defaults option is selected by default. CDW uses default values for the pods if a specific pod configuration is not used.
   - Select the 1 x Node option for the allocation of most node resources found in the environment, to the Impala executors and coordinators.

   Cdw Defaults and 1 x Node are read-only options.
5. Click Copy Config to create and edit a new configuration with the option that you selected earlier as the basis.
   a) Specify the name for your configuration in the Cloned Config Name field.
   b) Enter a description for the new configuration in the Description for new config field.
   c) Click Create New Config.

      A new pod configuration is created, which you can now customize.
6. Specify the values for the following parameters under the Coordinator section:
   - Memory
   - Cpus
   - Xmx (maximum memory allocation pool for a Java Virtual Machine)
   - Xms (initial memory allocation pool for a Java Virtual Machine)
   - AC Slots (admission_control_slots flag)
   - Cache size (size of the data cache)
   - Scratch size (limit of Impala scratch space)
   - Overhead size (size for resources used by tools run by the containers)

   Note: By default, the total space allocated for scratch, cache, and overhead is 600GiB, with scratch=280GiB, cache=280GiB, and overhead=40GiB. You can allocate space for these components in the following format: [***QUANTITY***][***UNIT***].

   The quantity can be a decimal number, separated by a period (.). For example, 240.5GiB. The unit must be in GiB (gigibytes).

   1GiB (gibibyte) equals 1.074GB (gigabytes).

7. Specify the values for the following parameters under the Executor section:
   - Memory
   - Cpus
   - Xmx (maximum memory allocation pool for a Java Virtual Machine)
   - Xms (initial memory allocation pool for a Java Virtual Machine)
   - AC Slots (admission_control_slots flag)
   - Cache (size of the data cache)
   - Cache size (size of the data cache)
   - Scratch size (limit of Impala scratch space)
   - Overhead size (size for resources used by tools run by the containers)

   Note: By default, the total space allocated for scratch, cache, and overhead is 600GiB, with scratch=280GiB, cache=280GiB, and overhead=40GiB. You can allocate space for these components in the following format: [***QUANTITY***][***UNIT***].

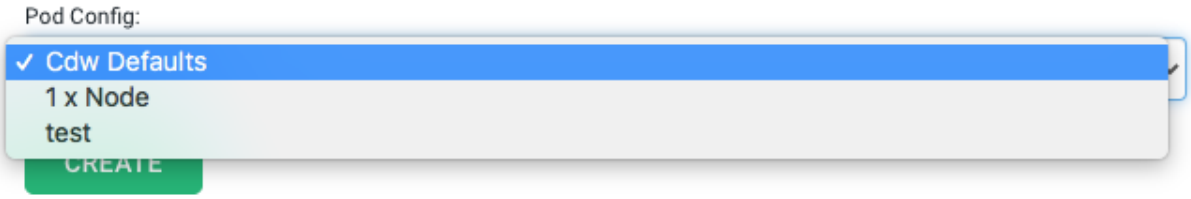
   The quantity can be a decimal number, separated by a period (.). For example, 240.5GiB. The unit must be in GiB (gigibytes).

   1GiB (gibibyte) equals 1.074GB (gigabytes).
8. Specify the values for the following parameters under the Catalog section:
   - Memory
   - Cpus
   - Xmx (maximum memory allocation pool for a Java Virtual Machine)
   - Xms (initial memory allocation pool for a Java Virtual Machine)
9. Specify the values for the following parameters under the Default Settings section:
   - MaxQueryMemLimit
   - MinQueryMemLimit
   - mt_dop
10. Click Apply under the **EDIT POD CONFIGURATION** tab to save the custom settings.

    The “Configuration update initiated” message is displayed.
11. Click the Set as default configuration toggle button to make this a default pod configuration.

    This makes a pod configuration the default configuration at the environment level.
12. Click APPLY at the top of the Environment Details page.

    The new pod configuration becomes available in the Pod Config drop-down menu as shown in the following image. You can select this Impala pod configuration while creating a new Impala Virtual Warehouse:



Results

While adding a new Impala Virtual Warehouse, you can select the Pod Configuration to be used for resource allocation. The default value is "Cdw Defaults", but you can select other configurations available in your environment that you created using these steps.