LakeFormation

Best Practices

Issue	01
Date	2024-01-31

HUAWEI

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1 Configuring Interconnection Between Open Source Spark and LakeFormation

1.1 Preparing Environment

Before interconnecting open source component Spark with LakeFormation, complete the following operations:

Step 1 Prepare an available open source Spark environment and Hive environment. Install the Git environment.

Currently, only Spark 3.1.1 and Spark 3.3.1 are supported. The Hive kernel version is 2.3.

Step 2 Prepare a LakeFormation instance. For details, see **Creating an Instance**.

Step 3 Create a LakeFormation access client in the same VPC and subnet as Spark. For details, see **Managing Clients**.

Step 4 Prepare the development environment. For details, see the part "Preparing the Environment" in**Preparing the Development Environment**. You can choose whether to install and configure IntelliJ IDEA.

Step 5 Download the LakeFormation client.

- **Method 1: downloading the release version**

  Download link: **https://gitee.com/HuaweiCloudDeveloper/huaweicloud-lake-formation-lakecat-sdk-java/releases**

  Download the corresponding client based on the Spark and Hive versions. For example, if the versions of Spark and Hive are 3.1.1 and 2.3.7, download **lakeformation-lakecat-client-hive2.3-spark3.1-1.0.0.jar**.

- **Method 2: compiling the client locally**

  a. Obtain the client code.

  Download link: **https://gitee.com/HuaweiCloudDeveloper/huaweicloud-lake-formation-lakecat-sdk-java**

  b. Run the following command in Git to switch the branch to **master_dev**:

  git checkout master_dev

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

c. Configure the Maven source. For details, see **Obtaining the SDK and Configuring Maven**.

d. Obtain the following JAR package and corresponding POM file and save the file to the local Maven repository.

For example, if the local repository directory is **D:\maven\repository**, save the file in the **D:\maven\repository\com\huaweicloud\hadoop-huaweicloud\3.1.1-hw-53.8** path.

- JAR package: **https://github.com/huaweicloud/obsa-hdfs/blob/master/release/hadoop-huaweicloud-3.1.1-hw-53.8.jar**
- POM file: **https://github.com/huaweicloud/obsa-hdfs/blob/master/hadoop-huaweicloud/pom.xml** (rename the downloaded file **hadoop-huaweicloud-3.1.1-hw-53.8.pom**)

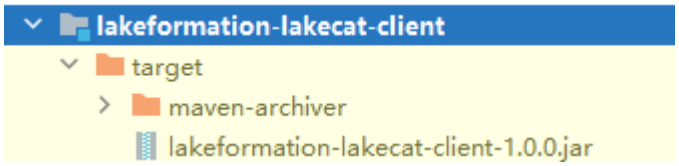
e. Package and upload the client code.

f. Go to the client project directory and run the following command to package the client code:

mvn clean install -DskipTests=true -P"*${SPARK_PROFILE}***" -P"***${HIVE_PROFILE}***"**

- **SPARK_PROFILE**: Enter **spark-3.1** or **spark-3.3** based on the Spark version.
- **HIVE_PROFILE**: Enter **hive-2.3**.

g. After the packaging is complete, obtain the **lakeformation-lakecat-client-1.0.0.jar** package from the **target** directory of **lakeformation-lakecat-client**.



Step 6 Prepare and replace JAR packages required by the Hive kernel.

NOTE

If only **SparkCatalogPlugin** is used for interconnection (MetastoreClient is not used), skip this step.

- **Method 1: downloading the JAR packages needed for Hive pre-building.**

  Download link: **https://gitee.com/HuaweiCloudDeveloper/huaweicloud-lake-formation-lakecat-sdk-java/releases**

  Download the corresponding client based on the Spark and Hive versions. For example, if the versions of Spark and Hive are 3.1.1 and 2.3.7, download **hive-exec-2.3.7-core.jar** and **hive-common-2.3.7.jar**.

- **Method 2: modifying Hive-related JAR packages locally**

  If the connected environment is Spark 3.1.1, use Hive 2.3.7. If the interconnected environment is Spark 3.3.1, use Hive 2.3.9.

  In Windows, you need to perform Maven operations in the WSL development environment.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

a. Download the Hive source code based on the Hive version.

   For example, if the Hive kernel version is 2.3.9, the download link is **https://github.com/apache/hive/tree/rel/release-2.3.9**.

b. Apply the patch in the LakeFormation client code to the Hive source code.

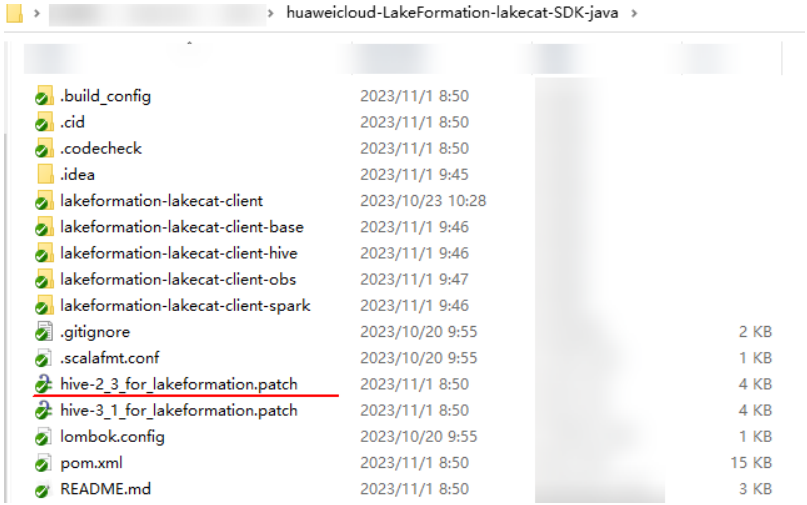
   i. Switch the Hive source code branch as required. For example, if the Hive kernel version is 2.3.9, run the following command:

      git checkout rel/release-*2.3.9*

   ii. Run the following command to apply the **patch** file to the Hive source code project after the branch is switched:

      mvn patch:apply -DpatchFile=${*your patch file location***}**

      In the command, **your patch file location** indicates the storage path of the **hive-2_3_for_lakeformation.patch** file. The **patch** file can be obtained from the client project, as shown in the following figure.



   iii. Run the following command to recompile the Hive kernel source code:

      mvn clean install -DskipTests=true

Step 7 Add the JAR packages required by the Spark environment.

Obtain the JAR packages listed in the following table and supplement or replace them in the **jars** directory in Spark.

Table 1-1 JAR packages required

No.	JAR Package	How to Obtain
1	spring-web-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-web/5.3.24/**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

No.	JAR Package	How to Obtain
2	spring-core-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-core/5.3.24/**
3	spring-context-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-context/5.3.24/**
4	spring-beans-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-beans/5.3.24/**
5	caffeine-2.9.3.jar	**https://mirrors.huaweicloud.com/repository/maven/com/github/ben-manes/caffeine/caffeine/2.9.3/**
6	mapstruct-1.5.3.Final.jar	**https://mirrors.huaweicloud.com/repository/maven/org/mapstruct/mapstruct/1.5.3.Final/**
7	log4j-api-2.19.0.jar	**https://mirrors.huaweicloud.com/repository/maven/org/apache/logging/log4j/log4j-api/2.19.0/**
8	java-sdk-core-3.2.4.jar (If only **Custom Authentication Information Obtaining Class** is used for token authentication, this JAR package is not required.)	**https://mirrors.huaweicloud.com/repository/maven/huaweicloudsdk/com/huawei/apigateway/java-sdk-core/3.2.4/**
9	bcprov-jdk15to18-1.70.jar	**https://mirrors.huaweicloud.com/repository/maven/org/bouncycastle/bcprov-jdk15to18/1.70/**
10	jca-1.0.4.jar	**https://mirrors.huaweicloud.com/repository/maven/org/openeuler/jca/1.0.4/**
11	hadoop-huaweicloud-3.1.1-hw-53.8.jar	**https://github.com/huaweicloud/obsa-hdfs/blob/master/release/hadoop-huaweicloud-3.1.1-hw-53.8.jar**
12	lakeformation-lakecat-client-1.0.0.jar	Obtain the package by referring to the operation in **Step 5**.
13	hive-exec-${version}-core.jar	Obtain the package by referring to the operation in **Step 6**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

No.	JAR Package	How to Obtain
14	hive-common-${version}.jar	Obtain the package by referring to the operation in **Step 6**.

----End

1.2 Interconnecting Spark with LakeFormation

NOTE

When using PySpark, trim off the **spark.hadoop** prefix from each parameter, but keep the rest of these parameters and add them to the **hive-site.xml** configuration file.

Adding Interconnection Configuration Items

Add the following configuration items to the **spark/conf/spark-defaults.conf** file:

```
# Project ID. This parameter is mandatory. The value is for reference only.
spark.hadoop.lakeformation.project.id=Project ID
# LakeFormation instance ID. This parameter is optional. You can obtain the value from the LakeFormation instance page. If this parameter is not specified, the default instance is connected. The value configured here is for reference only.
spark.hadoop.lakeformation.instance.id=LakeFormation Instance ID
#AK information for lakeformation IAM authentication. This parameter is optional. Ignore it if you plan to use the custom authentication information obtaining class.
spark.hadoop.lakeformation.authentication.access.key=AK
#SK information for lakeformation IAM authentication. This parameter is optional. Ignore it if you plan to use the custom authentication information obtaining class.
spark.hadoop.lakeformation.authentication.secret.key=SK
# IAM authentication information securitytoken for accessing lakeformation. This parameter is optional and is used together with a temporary AK/SK. If a permanent AK/SK or the custom authentication information obtaining class is used, ignore this parameter.
spark.hadoop.lakeformation.authentication.security.token=securitytoken information
```

NOTE

The project ID must be configured and other parameters are optional. Set them based on the site requirements.

- For how to obtain a project ID, see **Obtaining a Project ID**.
- For how to obtain the ID of a LakeFormation instance, see **How Do I Obtain the ID of a LakeFormation Instance?**.
- For how to obtain an AK/SK, see **How Do I Obtain the AK/SK?**.
- For how to obtain a securityToken, see **Obtaining a Temporary Access Key and SecurityToken Through a Token**.

These configuration items can also take effect after being added to **hive-site.xml** or **core-site.xml**. Remember to trim off the **spark.hadoop** prefix when adding them.

Interconnecting with OBS

Add the following configuration items to the **spark/conf/spark-defaults.conf** file:

```
# Fixed configuration for interconnecting with OBS. The endpoint needs to be configured based on the region.
spark.hadoop.fs.obs.impl=org.apache.hadoop.fs.obs.OBSFileSystem
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
spark.hadoop.fs.AbstractFileSystem.obs.impl=org.apache.hadoop.fs.obs.OBS
spark.hadoop.fs.obs.endpoint=obs.xxx.huawei.com

# Specify LakeFormationObsCredentialProvider as the class for obtaining OBS credentials.
spark.hadoop.fs.obs.credentials.provider=com.huawei.cloud.dalf.lakecat.client.obs.LakeFormationObsCredentialProvider

# Optional parameter. Disable the OBS file system cache. This configuration needs to be added for long tasks to prevent the temporary AK/SK in the cache from becoming invalid.
spark.hadoop.fs.obs.impl.disable.cache=true
```

NOTE

Endpoint: Endpoints vary in different services and regions. Obtain the value of this parameter from **Regions and Endpoints**.

These configuration items can also take effect after being added to **core-site.xml**. Remember to trim off the **spark.hadoop** prefix when adding them.

Interconnecting with LakeFormation Metadata

You can use either of the following methods to connect Spark to LakeFormation. You are advised to use either method as required.

- Interconnection using SparkCatalogPlugin: Spark SessionCatalogV2 allows you to connect to different catalogs in the same session. This feature is still experimental and does not support some SQL commands.
- Interconnection using MetastoreClient: MetastoreClient relies on Spark HiveExternalCatalog and Hive MetastoreClient mechanisms to execute most Hive SQL commands. However, it does not allow connecting to different catalogs simultaneously.

Interconnection using SparkCatalogPlugin:

1. Add the following configuration items to the **spark/conf/spark-defaults.conf** file. If multiple catalogs need to be interconnected at the same time, configure the following configuration in multiple lines:

   ```
   # Specify the catalog implementation class. This parameter is mandatory. spark_catalog_name indicates the catalog name in Spark. Replace it as required.
   spark.sql.catalog.$
   {spark_catalog_name}=com.huawei.cloud.dalf.lakecat.client.spark.LakeFormationSparkCatalog
   # Name of the catalog to be connected (lakeformation_catalog_name is the catalog in lakeFormation). This parameter is optional. If it is not set, the Hive catalog is connected instead. The value here is for reference only.
   spark.sql.catalog.${spark_catalog_name}.lakecat.catalogname.default=${lakeformation_catalog_name}
   ```

2. Verify the interconnection.

   After the interconnection, you can access LakeFormation through spark-shell, spark-submit, or spark-sql. The following uses spark-sql as an example.

   - Switch the database. (You need to specify the catalog name during the switchover. The database corresponding to *database_name* must exist in LakeFormation.)

     use *spark_catalog_name.database_name*;

   - View the table information.

     show tables;

   - Create a database. (You cannot directly create a database with the same name as the catalog. You need to specify the catalog.)

     create database *catalog_name.test*;

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Interconnection using MetastoreClient:

1. Add the following configuration items to **spark-defaults.conf**:
   ```
   spark.sql.catalogImplementation=hive
   ```
2. Add the **hive-site.xml** file to the **spark/conf/** folder (edit this file if it already exists) and add the following configurations to the **hive-site.xml** file:
   ```
   <configuration>
   <!--Fixed configuration. Enable the custom metastore client.-->
   <property>
   <name>hive.metastore.session.client.class</name>
   <value>com.huawei.cloud.dalf.lakecat.client.hiveclient.LakeCatMetaStoreClient</value>
   </property>
   <!--Name of the Lakeformation catalog to be connected. This parameter is optional. If it is not set, the Hive catalog is connected instead. The value of this parameter is for reference only.
   <property>
   <name>lakecat.catalogname.default</name>
   <value>hive</value>
   </property>
   <!--Hive execution path. This parameter is optional. If the HDFS is not connected, local path /tmp/hive is used by default. The value here is for reference only.
   <property>
   <name>hive.exec.scratchdir</name>
   <value>/tmp/hive</value>
   </property>
   </configuration>
   ```

   In addition to adding configurations to **hive-site.xml**, you can also add configurations starting with **spark.hadoop** in the **spark-defaults.conf** configuration file, for example, add **spark.hadoop.hive.metastore.session.client.class=com.huawei.cloud.dalf.lakecat.client.hiveclient.LakeCatMetaStoreClient**.

   NOTE

   - The permission on the **hive.exec.scratchdir** path must be changed to **777**. Otherwise, the Hive client initialization will be abnormal.
   - You need to create a database named **default** in the catalog corresponding to **lakecat.catalogname.default**. (If the database has been created, ignore it.) Otherwise, spark-sql initialization will be abnormal or spark-shell cannot be connected.
3. Verify the interconnection.

   After the interconnection, you can use spark-shell or execute SQL statements to access LakeFormation. The following uses spark-sql as an example.

   - Switch the database. (You do not need to specify the catalog name during the switchover.)

     use database_name;
   - View the table information.

     show tables;

Integrating the SQL Authentication Plug-in

Step 1 To use the authentication plug-in, you must implement and specify a custom user information obtaining class. For details, see **Custom User Information Obtaining Class**.

Step 2 Add the following configuration to the **spark-default.conf** configuration file:

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
com.huawei.cloud.dalf.lakecat.client.spark.v31.authorizer.LakeFormationSparkSQLExtension
spark.sql.extensions=com.huawei.cloud.dalf.lakecat.client.spark.authorizer.LakeFormationSparkSQLExtension
```

----End

NOTE

- After the permission plug-in is integrated, if the current user (specified by **Custom User Information Obtaining Class**) does not have the corresponding metadata permission, an exception is thrown when the SQL statement is executed.
- If the current user has the **IAM LakeFormation:policy:create** permission and the current user (specified by **Custom User Information Obtaining Class**) and authentication information (specified by **Custom Authentication Information Obtaining Class**) are unified users, SQL authentication will be skipped.
- Currently, filtering functions are not supported. Databases, tables, and rows cannot be filtered, and columns cannot be masked.

Log Printing

You can add **log4j.logger.org.apache=WARN** to the **log4j.properties** file to disable the HttpClient request logging function of the LakeFormation client.

1.3 Performing Secondary Development After Interconnection

You can perform secondary development as required. Currently, the following examples are provided.

- Custom authentication information obtaining class: used to obtain IAM authentication information for accessing LakeFormation.
- Custom user information obtaining class: used to obtain the information of the user who accesses LakeFormation.

Custom Authentication Information Obtaining Class

The IdentityGenerator class is used to obtain IAM authentication information (token, permanent AK/SK, and temporary AK/SK and securityToken) for accessing LakeFormation.

LakeFormation provides a default class for obtaining authentication information. The AK/SK is obtained from the configuration file to generate authentication information.

In addition to the default authentication information obtaining class provided by LakeFormation, you can implement the default authentication information obtaining class.

1. Develop code.

   The implementation project is as follows. Add the lakeformation-lakecat-client dependency to the POM file of the Maven project.

   ```
   <dependency>
   <groupId>com.huawei.lakeformation</groupId>
   <artifactId>lakeformation-lakecat-client</artifactId>
   <version>${lakeformation.version}</version>
   </dependency>
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
Add a class for obtaining authentication information to implement the IdentityGenerator API.

/*
* Copyright (c) Huawei Technologies Co., Ltd. 2023-2023. All rights reserved.
*/

package com.huawei.cloud.dalf.lakecat.examples;

import com.huawei.cloud.dalf.lakecat.client.ConfigCenter;
import com.huawei.cloud.dalf.lakecat.client.identity.Identity;
import com.huawei.cloud.dalf.lakecat.client.identity.IdentityGenerator;

import java.util.Collections;

/**
* Identity information generator example
*
*/
public class LakeFormationExampleIdentityGenerator implements IdentityGenerator {
public String token;

@Override
public void initialize(ConfigCenter configCenter) {
//Perform initialization.
}

@Override
public Identity generateIdentity() {
//Return the IAM authentication information.
}
}
```

2. Configure integration.

   Use Maven to pack the code and place the JAR package in the **spark/jars** directory.

   Add the corresponding configurations for different interconnection methods:

   - If SparkCatalogPlugin is used for interconnection, add the following configurations to the **spark-default.conf** configuration file:

     ```
     # Authentication information obtaining class. Set this parameter based on the implementation class path. The value is for reference only.
     spark.sql.catalog.catalog_name.lakecat.auth.identity.util.class=com.huawei.cloud.dalf.lakecat.client.spark.v31.impl.SparkDefaultIdentityGenerator
     ```

   - You can use either of the following method to complete interconnection using MetastoreClient:

     Add the following configuration to **spark-default.conf**:

     ```
     # Authentication information obtaining class. Set this parameter based on the implementation class path. The value is for reference only.
     spark.hadoop.lakecat.auth.identity.util.class=com.huawei.cloud.dalf.lakecat.client.spark.v31.impl.SparkDefaultIdentityGenerator
     ```

     Alternatively, add the following configuration to **hive-site.xml**:

     ```
     <!--Authentication information obtaining class. The value is for reference only.-->
     <property>
     <name>lakecat.auth.identity.util.class</name>
     <value>com.huawei.cloud.dalf.lakecat.examples.LakeFormationExampleIdentityGenerator</value>
     </property>
     ```

Custom User Information Obtaining Class

The AuthenticationManager class is used to obtain the information of the user who accesses LakeFormation, which may be an IAM user or a local LDAP user. The default user information obtaining class obtains the user information using **UserGroupInformation.getCurrentUser()**.

In addition to the default user information obtaining class introduced here, you can implement other user information obtaining method.

NOTE

If user authentication information is used to access LakeFormation, the user information must be consistent with the user identity information (that is, the username and source must be consistent).

1. Develop code.

   The implementation project is as follows. Add the lakeformation-lakecat-client dependency to the POM file of the Maven project.

   ```
   <dependency>
   <groupId>com.huawei.lakeformation</groupId>
   <artifactId>lakeformation-lakecat-client</artifactId>
   <version>${lakeformation.version}</version>
   </dependency>
   User information obtaining class, which implements the AuthenticationManager API.

   /*
   * Copyright (c) Huawei Technologies Co., Ltd. 2023-2023. All rights reserved.
   */

   package com.huawei.cloud.dalf.lakecat.examples;

   import com.huawei.cloud.dalf.lakecat.client.ConfigCenter;
   import com.huawei.cloud.dalf.lakecat.client.identity.AuthenticationManager;
   import com.huawei.cloud.dalf.lakecat.client.model.Principal;

   public class ExampleAuthenticationManager implements AuthenticationManager {
   @Override
   public void initialize(ConfigCenter configCenter) {
   //Perform initialization.
   }

   @Override
   public Principal getCurrentUser() {
   //Return the information about the current user.
   }
   }
   ```

2. Configure integration.

   Use Maven to pack the code and place the JAR package in the **spark/jars** directory.

   Add the corresponding configurations for different interconnection methods:

   - If SparkCatalogPlugin is used for interconnection, add the following configurations to the **spark-default.conf** configuration file:

     ```
     # Optional parameter. Authentication manager implementation class, which is used to obtain the information of the current user. The value configured here is for reference only.
     spark.sql.catalog.catalog_name.lakeformation.authentication.manager.class=com.huawei.cloud.dalf.lakecat.examples.ExampleAuthenticationManager
     # Optional parameter, which specifies whether to specify the current user as the resource owner during resource creation. The default value is false.
     spark.sql.catalog.catalog_name.lakeformation.owner.designate=true
     ```

   - You can use either of the following method to complete interconnection using MetastoreClient:

     Add the following configuration to **spark-default.conf**:

     ```
     # Optional parameter. Authentication manager implementation class, which is used to obtain the information of the current user. The value configured here is for reference only.
     spark.hadoop.lakeformation.authentication.manager.class=com.huawei.cloud.dalf.lakecat.examples.ExampleAuthenticationManager
     # Optional parameter, which specifies whether to specify the current user as the resource owner
     ```

```
during resource creation. The default value is false.
spark.hadoop.lakeformation.owner.designate=true
```

Alternatively, add the following configuration to **hive-site.xml**:

```
<!--Optional parameter. Authentication manager implementation class, which is used to obtain the information of the current user. The value configured here is for reference only.-->
<property>
<name>lakeformation.authentication.manager.class</name>
<value>com.huawei.cloud.dalf.lakecat.examples.ExampleAuthenticationManager</value>
</property>
<!--Optional parameter, which specifies whether to specify the current user as the resource owner during resource creation. The default value is false.-->
<property>
<name>lakeformation.owner.designate</name>
<value>true</value>
</property>
</configuration>
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2 Configuring Interconnection Between Open Source Hive and LakeFormation

2.1 Preparing Environment

Before interconnecting open source component Hive with LakeFormation, complete the following operations:

Step 1 Prepare an available open-source Hive environment (Hive 2.3 and Hive 3.1 are supported) and install the Git environment.

Step 2 Prepare a LakeFormation instance. For details, see **Creating an Instance**.

Step 3 Create a LakeFormation access client in the same VPC and subnet as Hive. For details, see **Managing Clients**.

Step 4 Prepare the development environment. For details, see the part "Preparing the Environment" in**Preparing the Development Environment**. You can choose whether to install and configure IntelliJ IDEA.

Step 5 Download the LakeFormation client.

- **Method 1: downloading the release version**

  Download link: **https://gitee.com/HuaweiCloudDeveloper/huaweicloud-lake-formation-lakecat-sdk-java/releases**

  Download the corresponding client based on the Spark and Hive versions. For example, if the Hive version is 2.3.9, download **lakeformation-lakecat-client-hive2.3-spark3.1-1.0.0.jar**.

- **Method 2: compiling the client locally**

  a. Obtain the client code.

  Download link: **https://gitee.com/HuaweiCloudDeveloper/huaweicloud-lake-formation-lakecat-sdk-java**

  Run the following command in Git to switch the branch to **master_dev**:

  git checkout master_dev

  b. Configure the Maven source. For details, see **Obtaining the SDK and Configuring Maven**.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

c. Obtain the following JAR package and corresponding POM file and save the file to the local Maven repository.

For example, if the local repository directory is **D:\maven\repository**, save the file in the **D:\maven\repository\com\huaweicloud\hadoop-huaweicloud\3.1.1-hw-53.8** path.

- JAR package: **https://github.com/huaweicloud/obsa-hdfs/blob/master/release/hadoop-huaweicloud-3.1.1-hw-53.8.jar**
- POM file: **https://github.com/huaweicloud/obsa-hdfs/blob/master/hadoop-huaweicloud/pom.xml** (rename the downloaded file **hadoop-huaweicloud-3.1.1-hw-53.8.pom**)

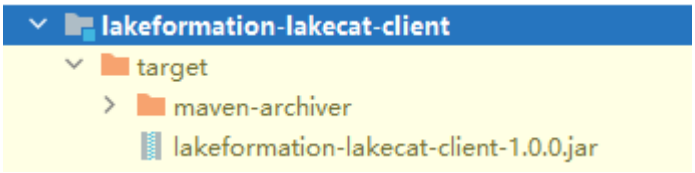
d. Package and upload the client code.

Go to the client project directory and run the following command to package the client code:

mvn clean install -DskipTests=true -P"*${HIVE_PROFILE}***"**

HIVE_PROFILE: Set this parameter to **hive-2.3** or **hive-3.1**.

e. After the packaging is complete, obtain the **lakeformation-lakecat-client-1.0.0.jar** package from the **target** directory of **lakeformation-lakecat-client**.



Step 6 Download the JAR packages required by the Hive kernel.

- **Method 1: downloading the JAR packages needed for Hive pre-building.**

  Download link: **https://gitee.com/HuaweiCloudDeveloper/huaweicloud-lake-formation-lakecat-sdk-java/releases**

  Download the corresponding client based on the Hive versions. For example, if the Hive version is 2.3.9, download **hive-exec-2.3.9.jar** and **hive-common-2.3.9.jar**.

- **Method 2: modifying Hive-related JAR packages locally**

  In Windows, you need to perform Maven operations in the WSL development environment.

  a. Download the Hive source code based on the Hive version.

  For example, if the Hive kernel version is 2.3.9, the download link is **https://github.com/apache/hive/tree/rel/release-2.3.9**.

  b. Apply the patch in the LakeFormation client code to the Hive source code.

  i. Switch the Hive source code branch as required. For example, if the Hive kernel version is 2.3.9, run the following command:

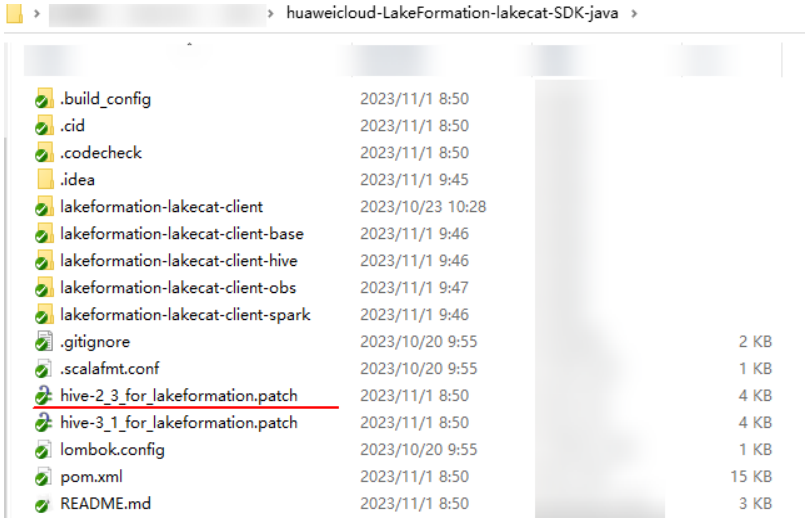
  git checkout rel/release-*2.3.9*

  ii. Run the following command to apply the **patch** file to the Hive source code project after the branch is switched:

  mvn patch:apply -DpatchFile=${*your patch file location***}**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

In the command, **your patch file location** indicates the storage path of **hive-2_3_for_lakeformation.patch** or **hive-3_1_for_lakeformation.patch**. The **patch** file can be obtained from the client project, as shown in the following figure.



iii. Run the following command to recompile the Hive kernel source code:

mvn clean install -DskipTests=true

Step 7 Prepare and replace JAR packages required by the Hive kernel.

Obtain the JAR packages listed in the following table and supplement or replace them in the **lib** directory in the Hive installation environment.

Table 2-1 JAR packages required by the Hive environment

No.	JAR Package	How to Obtain
1	spring-web-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-web/5.3.24/**
2	spring-core-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-core/5.3.24/**
3	spring-context-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-context/5.3.24/**
4	spring-beans-5.3.24.jar	**https://mirrors.huaweicloud.com/repository/maven/org/springframework/spring-beans/5.3.24/**

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

No.	JAR Package	How to Obtain
5	caffeine-2.9.3.jar	**https://mirrors.huaweicloud.com/repository/maven/com/github/ben-manes/caffeine/caffeine/2.9.3/**
6	mapstruct-1.5.3.Final.jar	**https://mirrors.huaweicloud.com/repository/maven/org/mapstruct/mapstruct/1.5.3.Final/**
7	http-core-4.4.13.jar (If the Hive kernel version is 3.1, this JAR package is not required.)	**https://mirrors.huaweicloud.com/repository/maven/org/apache/httpcomponents/httpcore/4.4.13/**
8	jca-1.0.4.jar	**https://mirrors.huaweicloud.com/repository/maven/org/openeuler/jca/1.0.4/**
9	Replace **commons-codec-1.4.jar** with **commons-codec-1.15.jar**. (If the Hive kernel version is 3.1, this JAR package is not required.)	**https://mirrors.huaweicloud.com/repository/maven/commons-codec/commons-codec/1.15/**
10	java-sdk-core-3.2.4.jar (If only **Custom Authentication Information Obtaining Class** is used for token authentication, this JAR package is not required.)	**https://mirrors.huaweicloud.com/repository/maven/huaweicloudsdk/com/huawei/apigateway/java-sdk-core/**
11	hadoop-huaweicloud-3.1.1-hw-53.8.jar	**https://github.com/huaweicloud/obsa-hdfs/blob/master/release/hadoop-huaweicloud-3.1.1-hw-53.8.jar**
12	lakeformation-lakecat-client-1.0.0.jar	Obtain the package by referring to the operation in **Step 5**.
13	hive-exec-${version}.jar	Obtain the package by referring to the operation in **Step 6**.
14	hive-common-${version}.jar	Obtain the package by referring to the operation in **Step 6**.

----End

2.2 Interconnecting Hive with LakeFormation

Step 1 Add the following content to the **hive-site.xml** file in the **conf** directory in the Hive server installation environment (replace some parameter values as prompted):

```
<property>
<name>hive.metastore.session.client.class</name>
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
<value>com.huawei.cloud.dalf.lakecat.client.hiveclient.LakeCatMetaStoreClient</value>
</property>
<property>
<name>lakeformation.project.id</name>
<value>****</value>
</property>
<property>
<name>lakeformation.instance.id</name>
<value>LakeFormation instance ID</value>
</property>
<!--AK information for lakeformation IAM authentication. This parameter is optional. Ignore it if you plan to use the custom authentication information obtaining class.-->
<property>
<name>lakeformation.authentication.access.key</name>
<value>AK</value>
</property>
<!--SK information for lakeformation IAM authentication. This parameter is optional. Ignore it if you plan to use the custom authentication information obtaining class.-->
<property>
<name>lakeformation.authentication.secret.key</name>
<value>SK</value>
</property>
<!--SecurityToken for accessing lakeformation IAM authentication information. This parameter is optional. If a permanent AK/SK or a custom authorizer information obtaining class is used, skip this parameter.-->
<property>
<name>lakeformation.authentication.security.token</name>
<value>SecurityToken information</value>
</property>
<property>
<name>fs.obs.impl</name>
<value>org.apache.hadoop.fs.obs.OBSFileSystem</value>
</property>
<property>
<name>fs.AbstractFileSystem.obs.impl</name>
<value>org.apache.hadoop.fs.obs.OBS</value>
</property>
<property>
<name>fs.obs.endpoint</name>
<value>Endpoint information</value>
</property>
<property>
<name>fs.obs.credentials.provider</name>
<value>com.huawei.cloud.dalf.lakecat.client.obs.LakeFormationObsCredentialProvider</value>
</property>
<property>
<name>fs.obs.impl.disable.cache</name>
<value>true</value>
</property>
<property>
<name>dfs.namenode.acls.enabled</name>
<value>false</value>
</property>
<!--Name of the LakeFormation catalog to be connected. This parameter is optional. If this parameter is not set, the Hive catalog is connected instead. The value of this parameter is for reference only.-->
<property>
<name>lakecat.catalogname.default</name>
<value>hive</value>
</property>
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

NOTE

- **lakeformation.project.id** indicates the project ID. For how to obtain a project ID, see **Obtaining a Project ID**.
- **lakeformation.instance.id** indicates the LakeFormation instance ID. For how to obtain the ID of a LakeFormation instance, see **How Do I Obtain the ID of a LakeFormation Instance?**.
- For how to obtain an AK/SK, see **How Do I Obtain the AK/SK?**.
- For how to obtain a securityToken, see **Obtaining a Temporary Access Key and SecurityToken Through a Token**.

Step 2 Restart the Hive service.

Step 3 Log in to the Hive client and run the following command:

show tables;

----End

2.3 Performing Secondary Development After Interconnection

You can perform secondary development as required. Currently, the following examples are provided.

- Custom authentication information obtaining class: used to obtain IAM authentication information for accessing LakeFormation.
- Custom user information obtaining class: used to obtain the information of the user who accesses LakeFormation.

Custom Authentication Information Obtaining Class

The IdentityGenerator class is used to obtain IAM authentication information (token, permanent AK/SK, and temporary AK/SK and securityToken) for accessing LakeFormation.

LakeFormation provides a default class for obtaining authentication information. The AK/SK is obtained from the configuration file to generate authentication information.

In addition to the default authentication information obtaining class provided by LakeFormation, you can implement the default authentication information obtaining class.

1. Develop code.

   The implementation project is as follows. Add the lakeformation-lakecat-client dependency to the POM file of the Maven project.

```
<dependency>
<groupId>com.huawei.lakeformation</groupId>
<artifactId>lakeformation-lakecat-client</artifactId>
<version>${lakeformation.version}</version>
</dependency>
//Add a class for obtaining authentication information to implement the IdentityGenerator API.
/*
* Copyright (c) Huawei Technologies Co., Ltd. 2023-2023. All rights reserved.
*/
```

```
package com.huawei.cloud.dalf.lakecat.examples;

import com.huawei.cloud.dalf.lakecat.client.ConfigCenter;
import com.huawei.cloud.dalf.lakecat.client.identity.Identity;
import com.huawei.cloud.dalf.lakecat.client.identity.IdentityGenerator;

import java.util.Collections;

/**
* Identity information generator example
*
*/
public class LakeFormationExampleIdentityGenerator implements IdentityGenerator {
public String token;

@Override
public void initialize(ConfigCenter configCenter) {
//Perform initialization.
}

@Override
public Identity generateIdentity() {
//Return the IAM authentication information.
}
}
```

2. Configure integration.

   Use Maven to pack the code and place the JAR package in the **hive-*xxx*/lib** directory. *xxx* indicates the Hive kernel version.

   Add the following configuration to **hive-site.xml**:

   ```
   <!--Authentication information obtaining class. The value is for reference only.-->
   <property>
   <name>lakecat.auth.identity.util.class</name>
   <value>com.huawei.cloud.dalf.lakecat.examples.LakeFormationExampleIdentityGenerator</value>
   </property>
   ```

3. Restart the Hive service.

Custom User Information Obtaining Class

The AuthenticationManager class is used to obtain the information of the user who accesses LakeFormation, which may be an IAM user or a local LDAP user. The default user information obtaining class obtains the user information using **UserGroupInformation.getCurrentUser()**.

In addition to the default user information obtaining class introduced here, you can implement other user information obtaining method.

NOTE

If user authentication information is used to access LakeFormation, the user information must be consistent with the user identity information (that is, the username and source must be consistent).

1. Develop code.

   The implementation project is as follows. Add the lakeformation-lakecat-client dependency to the POM file of the Maven project.

   ```
   <dependency>
   <groupId>com.huawei.lakeformation</groupId>
   <artifactId>lakeformation-lakecat-client</artifactId>
   <version>${lakeformation.version}</version>
   </dependency>
   ```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
User information obtaining class, which implements the AuthenticationManager API.

/*
* Copyright (c) Huawei Technologies Co., Ltd. 2023-2023. All rights reserved.
*/

package com.huawei.cloud.dalf.lakecat.examples;

import com.huawei.cloud.dalf.lakecat.client.ConfigCenter;
import com.huawei.cloud.dalf.lakecat.client.identity.AuthenticationManager;
import com.huawei.cloud.dalf.lakecat.client.model.Principal;

public class ExampleAuthenticationManager implements AuthenticationManager {
@Override
public void initialize(ConfigCenter configCenter) {
//Perform initialization.
}

@Override
public Principal getCurrentUser() {
//Return the information about the current user.
}
}
```

2. Configure integration.

   Use Maven to pack the code and place the JAR package in the **hive-*xxx*/lib** directory. *xxx* indicates the Hive kernel version.

   Add the following configuration to **hive-site.xml**:

```
<!--Optional parameter. Authentication manager implementation class, which is used to obtain the information of the current user. The value configured here is for reference only.-->
<property>
<name>lakeformation.authentication.manager.class</name>
<value>com.huawei.cloud.dalf.lakecat.examples.ExampleAuthenticationManager</value>
</property>

<!--Optional parameter, which specifies whether to specify the current user as the resource owner during resource creation. The default value is false.-->
<property>
<name>lakeformation.owner.designate</name>
<value>true</value>
</property>
</configuration>
```

3. Restart the Hive service.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.