

MapReduce Service

Best Practices

Issue 01
Date 2024-05-06



Copyright © Huawei 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 Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks 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 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.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:

<https://www.huawei.com/en/psirt/vul-response-process>

For vulnerability information, enterprise customers can visit the following web page:

<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

Contents

1 Data Analytics.....	1
1.1 Using Spark2x to Analyze IoV Drivers' Driving Behavior.....	1
1.2 Using Hive to Load HDFS Data and Analyze Book Scores.....	9
1.3 Using Hive to Load OBS Data and Analyze Enterprise Employee Information.....	15
1.4 Using Flink Jobs to Process OBS Data.....	24
1.5 Consuming Kafka Data Using Spark Streaming Jobs.....	30
1.6 Using Flume to Collect Log Files from a Specified Directory to HDFS.....	37
1.7 Kafka-based WordCount Data Flow Statistics Case.....	45
2 Data Migration.....	51
2.1 Data Migration Solution.....	51
2.1.1 Making Preparations.....	51
2.1.2 Exporting Metadata.....	52
2.1.3 Copying Data.....	53
2.1.4 Restoring Data.....	54
2.2 Using BulkLoad to Import Data to HBase in Batches.....	55
2.3 Migrating Data from MySQL to an MRS Hive Partitioned Table.....	60
2.4 Migrating Data from MRS HDFS to OBS.....	69
3 Data Backup and Restoration.....	73
3.1 HDFS Data.....	73
3.2 Hive Metadata.....	75
3.3 Hive Data.....	76
3.4 HBase Data.....	76
3.5 Kafka Data.....	82
4 System Interconnection.....	84
4.1 Using DBeaver to Access Phoenix.....	84
4.2 Using DBeaver to Access HetuEngine.....	91
4.3 Using Tableau to Access HetuEngine.....	97
4.4 Using Yonghong BI to Access HetuEngine.....	98
4.5 Interconnecting Hive with External Self-Built Relational Databases.....	101
4.6 Interconnecting Hive with CSS.....	106
4.7 Interconnecting Hive with External LDAP.....	108

1 Data Analytics

1.1 Using Spark2x to Analyze IoV Drivers' Driving Behavior

The best practices for Huawei Cloud MapReduce Service (MRS) guides you through the basic functions of MRS. This case shows you how to use the Spark2x component of MRS to analyze and collect statistics on driver behaviors and obtain the analysis results.

NOTE

This practice applies only to MRS 3.1.0. Create a cluster as instructed.

You can get started by reading the following topics:

1. [Scenario](#)
2. [Creating a Cluster](#)
3. [Preparing a Spark2x Sample Program and Sample Data](#)
4. [Creating a Job](#)
5. [Viewing the Job Execution Results](#)

Scenario

In this case, raw data is driver behavior information, including abrupt acceleration, abrupt deceleration, neutral sliding, overspeed, and fatigue driving. With the powerful analysis capability of the Spark2x component, you can analyze the driver behavior information of a specified period and obtain result statistics on the information.

Creating a Cluster

Step 1 Go to the [Buy Cluster](#) page.

Step 2 Click the **Custom Config** tab.

Configure cluster software information according to [Table 1-1](#).

Table 1-1 Software configurations

Parameter	Configuration
Region	EU-Dublin NOTE This document uses EU-Dublin as an example. If you want to perform operations in other regions, ensure that all operations are performed in the same region.
Billing Mode	Pay-per-use
Cluster Name	mrs_demo
Cluster Type	Analysis cluster (for offline data analysis)
Version Type	Normal
Cluster Version	MRS 3.1.0 NOTE This practice applies only to MRS 3.1.0.
Component	All components
Metadata	Local

Figure 1-1 Software configurations

The screenshot shows the MRS configuration page with the following settings:

- Region:** A dropdown menu with a location pin icon.
- Billing Mode:** Two buttons: "Yearly/Monthly" and "Pay-per-use" (selected).
- Cluster Name:** A text input field containing "mrs".
- Cluster Type:** Two buttons: "Custom" and "Hybrid ..." (selected).
- Hybrid cluster description:** A light blue box containing text: "Hybrid cluster", "This type is suitable for both offline data analysis and stream processing.", and "You can select analysis components such as Hadoop, Spark, HBase, and Hive, and stream processing components such as Kafka and Flume."
- Version Type:** Two buttons: "LTS" and "Normal" (selected).
- Cluster Version:** A dropdown menu showing "MRS 3.1.0".
- Component:** A section titled "Mandatory components and their dependent components are automatically selected. You can change components based on your needs. For some clusters, components cannot be added after creation. [Learn more](#)". Below it is a table:

<input checked="" type="checkbox"/>	Name	Version	Description
<input checked="" type="checkbox"/>	Hadoop	3.1.1	A framework that allows for the distributed processing of large data sets across clusters.
<input checked="" type="checkbox"/>	Spark2x	2.4.5	Apache Spark2x is a fast and general engine based on open source Spark2.x for large-scale data processing.
<input type="checkbox"/>	HBase	2.2.3	HBase - distributed, versioned, non-relational database.
<input checked="" type="checkbox"/>	Hive	3.1.0	Data warehouse software that facilitates query and management of large datasets stored in distributed storage systems.
<input type="checkbox"/>	Hue	4.7.0	The UI for Apache Hadoop.

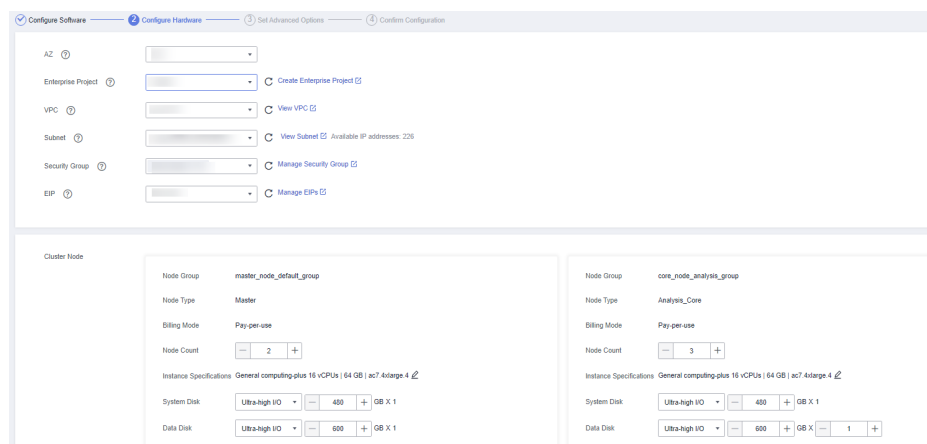
Step 3 Click **Next** to configure hardware.

Configure cluster hardware information according to [Table 1-2](#).

Table 1-2 Hardware configurations

Parameter	Configuration
AZ	AZ2
Enterprise Project	default
VPC	Select the VPC for which you want to create a cluster and click View VPC to view the name and ID of the VPC. If no VPC is available, create one.
Subnet	Select the subnet for which you want to create a cluster to enter the VPC and view the name and ID of the subnet. If no subnet is created under the VPC, click Create Subnet to create one.
Security Group	Auto create
EIP	Bind later
Cluster Node	Default settings

Figure 1-2 Hardware configurations



Step 4 Click **Next**. On the **Set Advanced Options** page, set the following parameters by referring to **Table 1-3** and retain the default settings for other parameters.

Table 1-3 Advanced configurations

Parameter	Configuration
Kerberos Authentication	Disable Kerberos authentication.
Username	Name of the administrator of MRS Manager. admin is used by default.
Password	Password of the MRS Manager administrator.

Parameter	Configuration
Confirm Password	Enter the password of the Manager administrator again.
Login Mode	Select Password .
Username	Name of the user for logging in to ECSs. root is used by default.
Password	Password for logging in to ECSs.
Confirm Password	Enter the password for logging in to ECSs again.

Figure 1-3 Advanced configurations

Configure Software — Configure Hardware — **3 Set Advanced Options** — 4 Confirm Configuration

Kerberos Authentication ?

Username admin

Password
The password will be required to log in to the MRS Manager.

Confirm Password

Login Mode **Password** Key Pair


Username root

Password
This password is required when you remotely log in to the ECS or BMS.

Confirm Password

Hostname Prefix ?
Enter the prefix for the computer hostname of an ECS or BMS in the cluster.

Set Advanced Options Configure

Step 5 Click **Next**. On the **Confirm Configuration** page, check the cluster configuration information. If you need to adjust the configuration, click  to go to the corresponding tab page and configure parameters again.

Step 6 Select **Secure Communications** and click **Buy Now**.

Step 7 Click **Back to Cluster List** to view the cluster status.

Cluster creation takes some time. The initial status of the cluster is **Starting**. After the cluster has been created successfully, the cluster status becomes **Running**.

----End

Preparing a Spark2x Sample Program and Sample Data

Step 1 Create an OBS parallel file system to store the Spark sample program, sample data, job execution results, and logs.

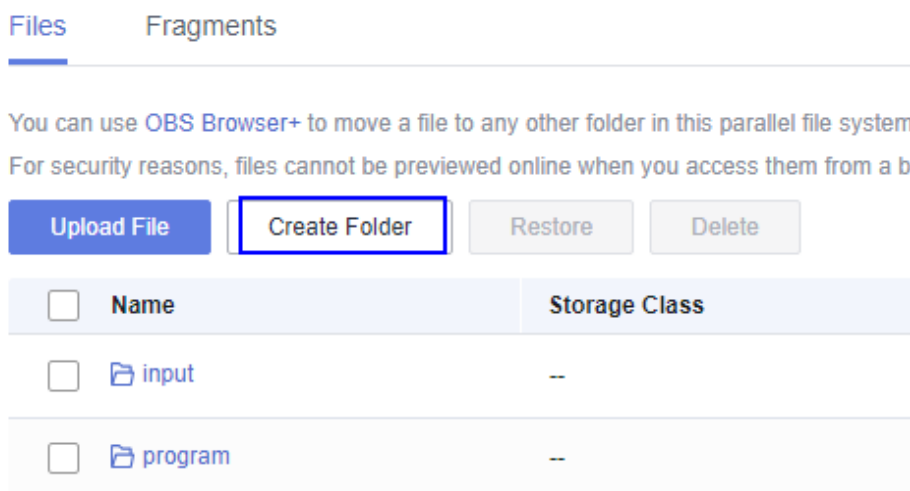
1. Log in to the HUAWEI CLOUD management console.
2. In the **Service List**, choose **Storage > Object Storage Service**.
3. In the navigation pane on the left, choose **Parallel File System** and click **Create Parallel File System** to create a file system named **obs-demo-analysis-hwt4**. Retain the default values for parameters such as **Policy**.

Figure 1-4 Creating a parallel file system



Step 2 Click the name of the file system. In the navigation pane on the left, choose **Files**. On the displayed page, click **Create Folder** to create the **program** and **input** folders, as shown in **Figure 1-5**.

Figure 1-5 Creating a folder



Step 3 Download the sample program **driver_behavior.jar** from https://mrs-obs-ap-southeast-1.obs.ap-southeast-1.myhuaweicloud.com/mrs-demon-samples/demon/driver_behavior.jar to the local PC.

Step 4 Go to the **program** folder. Click **Upload File** and select the local **driver_behavior.jar** sample program.

- Step 5** Click **Upload** to upload the sample program to the OBS bucket.
- Step 6** Obtain Spark sample data from <https://mrs-obs-ap-southeast-1.obs.ap-southeast-1.myhuaweicloud.com/mrs-demon-samples/demon/detail-records.zip>.
- Step 7** Decompress the downloaded **detail-records.zip** package to obtain the sample data.

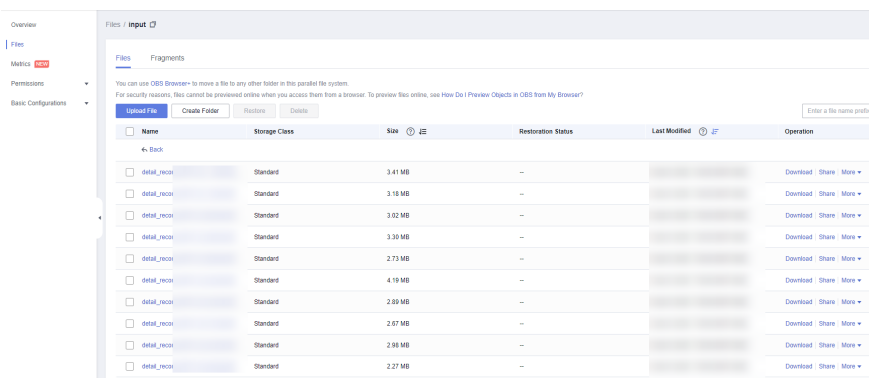
Figure 1-6 Sample data

detail_record_2017_01_02_08_00_00	3,056 KB
detail_record_2017_01_03_08_00_00	2,955 KB
detail_record_2017_01_04_08_00_00	4,291 KB
detail_record_2017_01_05_08_00_00	2,324 KB
detail_record_2017_01_06_08_00_00	3,088 KB
detail_record_2017_01_07_08_00_00	2,739 KB
detail_record_2017_01_08_08_00_00	2,797 KB
detail_record_2017_01_09_08_00_00	3,383 KB
detail_record_2017_01_10_08_00_00	3,253 KB
detail_record_2017_01_11_08_00_00	3,497 KB

- Step 8** Go to the **input** folder. Click **Upload File** and select the local Spark sample data. Click **Upload** to upload the sample data to the OBS bucket.

NOTE

Upload the data decompressed in **Step 7** to the **input** folder.



----End

Creating a Job

- Step 1** Log in to the MRS console, click the **mrs_demo** cluster on the displayed **Active Clusters** page.
- Step 2** Click the **Jobs** tab and then **Create** to create a job.

Figure 1-7 Creating a job

This is a program execution platform where you can process and analyze big data. [Learn more](#)



Step 3 Set job parameters by referring to [Figure 1-8](#).

Table 1-4 Configuring job parameters

Parameter	Configuration
Type	Select SparkSubmit .
Name	Enter driver_behavior_task .
Program Path	Click OBS and select the driver_behavior.jar package uploaded in Preparing a Spark2x Sample Program and Sample Data .
Program Parameter	Select --class in Parameter , and enter com.huawei.bigdata.spark.examples.DriverBehavior in Value .
Parameters	<p>Enter AK SK 1 Input path Output path.</p> <ul style="list-style-type: none"> For details about how to obtain the AK/SK, see the steps described in NOTE. 1 is a fixed input that is used to specify the program function invoked during job execution. <i>Input path</i> is the path you selected for the Program Path parameter. <i>Output path</i> should be a directory that does not exist, for example, obs://obs-demo-analysis-hwt4/output/. <p>NOTE To obtain the AK/SK, perform the following steps:</p> <ol style="list-style-type: none"> Log in to the Huawei Cloud management console. Click the username in the upper right corner and choose My Credentials. In the navigation pane on the left, choose Access Keys. Click Create Access Key to add a key. Enter the password and verification code as prompted. The browser automatically downloads the credentials.csv file. The file is in CSV format and separated by commas (.). In the file, the middle part is AK and the last part is SK.
Service Parameter	This parameter is left blank by default. Retain the default settings.

Figure 1-8 Creating a job

Create Job

* Type: SparkSubmit

* Name: driver_behavior_task

* Program Path: obs://.../program/driver_behavior.jar [HDFS] [OBS]

Program Parameter: --class com.huawei.bigdata.spark.examples.DriverBehavior

Parameters: obs://obs-demo-analysis-hwt4/input | obs://obs-demo-analysis-hwt4/output [HDFS] [OBS]

Service Parameter: 参数 值

Command Reference: spark-submit --class com.huawei.bigdata.spark.examples.DriverBehavior --master yarn-cluster obs://.../program/driver_behavior.jar ... 1 obs://obs-demo-analysis-hwt4/input obs://obs-demo-analysis-hwt4/output

[OK] [Cancel]

Step 4 Click **OK** to start executing the program.

----End

Viewing the Job Execution Results

Step 1 Go to the **Jobs** page to view the job execution status.

Figure 1-9 Execution status

This is a program execution platform where you can process and analyze big data. Learn more

Create Delete

Sep 20, 2021 - Oct 20, 2021 X All statuses All types

Name/ID	Username	Type	Status	Result	Queue	Submitted	Ended
driver_behavior_task b454d92c-6028-4d92-0001-1609b6e6a060		SparkSubmit	Completed	Successful	default		Oct 20, 2021 11:33:31 GMT+08:00

Step 2 Wait 1 to 2 minutes and log in to OBS console. Go to the output path of the **obs-demo-analysis-hwt4** file system to view the execution result. Click **Download** in the **Operation** column of the generated CSV file to download the file to your local PC.

Figure 1-10 Viewing the job execution results

Files Fragments

You can use OBS Browser+ to move a file to any other folder in this parallel file system. For security reasons, files cannot be previewed online when you access them from a browser. To preview files online, see How Do I Preview Objects in OBS from My Browser?

Upload File Create Folder Restore Delete

Enter a file name prefix. Q C

Name	Storage Class	Size	Restoration Status	Last Modified	Operation
._SUCCESS	Standard	0 byte	--		Download Share More
	Standard	28.32 MB	--		Download Share More

Step 3 Open the downloaded CSV file using Excel and classify the data in each column according to the fields defined in the program. The job execution results are obtained.

Figure 1-11 Execution result

Driver ID	License Plate Number	Abrupt Acceleration Times	Abrupt Brake Times	Neutral Sliding Times	Total Neutral Sliding Time	Overspeed Times	Total Overspeed Time	Fatigue Driving on the Accelerator Times	Times of Stepping on the Accelerator While Stopping	Oil Leakage Times
shenxian1000004	ADJ750	374	355	297	2810	3126	31484	3767	363	366
xiesiao1000001	AEB132	264	261	248	2525	2324	23434	2720	314	253
xiezhi1000009	A6CU11	255	310	254	2074	2535	23842	2631	312	279
duxu1000009	AT75H8	238	284	247	2632	2301	22338	2814	264	248
hanhui1000002	AZI419	401	444	327	2844	3349	31813	3997	433	371
panxian1000005	AX542C	395	434	330	2930	3531	33946	4307	417	441
zouan1000007	A58M83	360	385	315	2997	3181	31248	3594	389	385
likun1000003	AVM936	341	354	291	3043	3044	28728	3552	347	376
zengpeng1000000	AZQ110	340	344	272	2894	2763	25479	3274	284	337
haoweir1000008	A709GB	321	314	255	2659	2639	25522	3204	312	318

----End

1.2 Using Hive to Load HDFS Data and Analyze Book Scores

MRS offline processing clusters enable you to analyze and process massive amount of data as well as provide the results for later use.

Offline processing has low requirements on processing time. However, a large amount of data needs to be processed, which occupies a large number of compute and storage resources. Generally, offline processing is implemented through Hive/SparkSQL or MapReduce/Spark2x.

This practice describes how to import and analyze raw data using Hive after you create an MRS cluster and how to implement elastic and low-cost offline big data analysis.

You can get started by reading the following topics:

1. [Creating an MRS Offline Query Cluster](#)
2. [Importing Local Data to HDFS](#)
3. [Creating a Hive Table](#)
4. [Importing Raw Data to Hive for Analysis](#)

Scenario

Hive is a data warehouse built on Hadoop. It provides batch computing capability for the big data platform and is able to batch analyze and summarize structured and semi-structured data for data calculation. Hive operates structured data using Hive Query Language (HQL), a SQL-like language. HQL is automatically converted into MapReduce tasks for the query and analysis of massive data in the Hadoop cluster.

Hive is able to:

- Analyze massive structured data and summarizes analysis results.
- Allow complex MapReduce jobs to be compiled in SQL languages.
- Support flexible data storage formats, including JavaScript object notation (JSON), comma separated values (CSV), TextFile, RCFile, SequenceFile, and Optimized Row Columnar (ORC).

In this practice, user comments from the background of a book website are used as the raw data. After the data is imported to a Hive table, you can run SQL commands to query the most popular best-selling books.

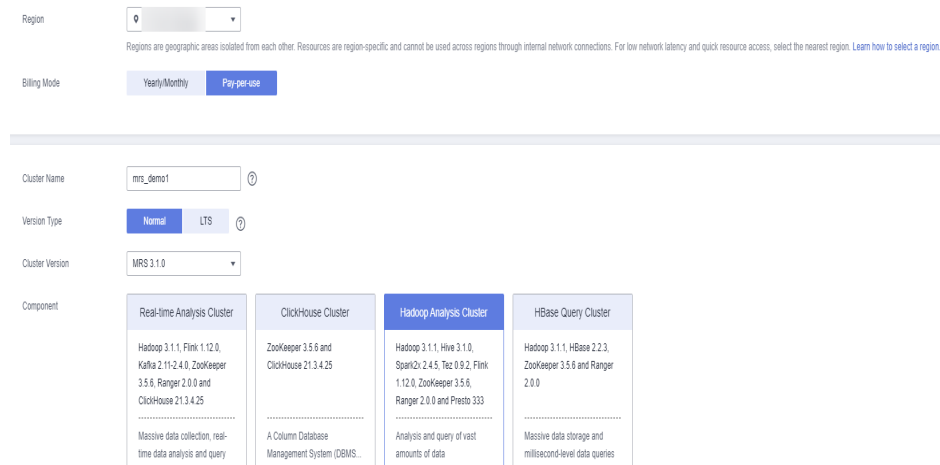
Creating an MRS Offline Query Cluster

1. Go to the [Buy Cluster](#) page.
2. Click the **Quick Config** tab and set configuration parameters.

Table 1-5 Software parameters (for reference only)

Parameter	Value
Region	EU-Dublin
Billing Mode	Pay-per-use
Cluster Name	MRS_demo
Version Type	Normal
Cluster Version	MRS 3.1.0
Component	Hadoop Analysis Cluster
AZ	AZ1
VPC	vpc-01
Subnet	subnet-01
Enterprise Project	default
Kerberos Authentication	Disabled
Username	root/admin
Password	Set the password for logging in to the cluster management page and ECS node, for example, Test!@12345 .
Confirm Password	Enter the password again.
Secure Communications	Select Enable .

Figure 1-12 Buying a Hadoop analysis cluster



3. Click **Buy Now** and wait until the MRS cluster is created.

Figure 1-13 Cluster purchased

Name/ID	Cluster Version	Cluster Type	Nodes	Status
mrs_7beac1fb-c54f-4769-bc3f-8b09583c9293	MRS 3.1.0	Analysis Cluster	5	Running

Importing Local Data to HDFS

1. Obtain the book comments file **book_score.txt** from the background of the book website and save it on the local host.

The file contains the following fields: user ID, book ID, book score, and remarks.

Some data is as follows:

```
202001,242,3,Good!
202002,302,3,Test.
202003,377,1,Bad!
220204,51,2,Bad!
202005,346,1,aaa
202006,474,4,None
202007,265,2,Bad!
202008,465,5,Good!
202009,451,3,Bad!
202010,86,3,Bad!
202011,257,2,Bad!
202012,465,4,Good!
202013,465,4,Good!
202014,465,4,Good!
202015,302,5,Good!
202016,302,3,Good!
...
```

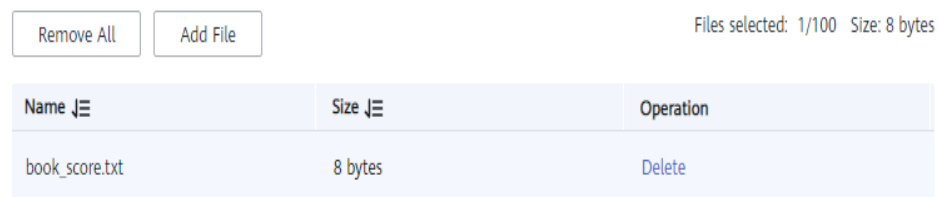
2. Log in to the OBS console, click **Create Bucket**, set the following parameters, and click **Create Now**.

Table 1-6 Bucket parameters

Parameter	Value
Region	EU-Dublin
Bucket Name	mrs-hive
Default Storage Class	Standard
Bucket Policy	Private
Direct Reading	Disable
Enterprise Project	default
Tags	-

After the bucket is created, click the bucket name. In the navigation pane on the left, choose **Objects** and click **Upload Object** to upload the data file.

Figure 1-14 Uploading an object



- Switch back to the MRS console and click the name of the created MRS cluster. On the **Dashboard** page, click **Synchronize** next to **IAM User Sync**. The synchronization takes about five minutes.

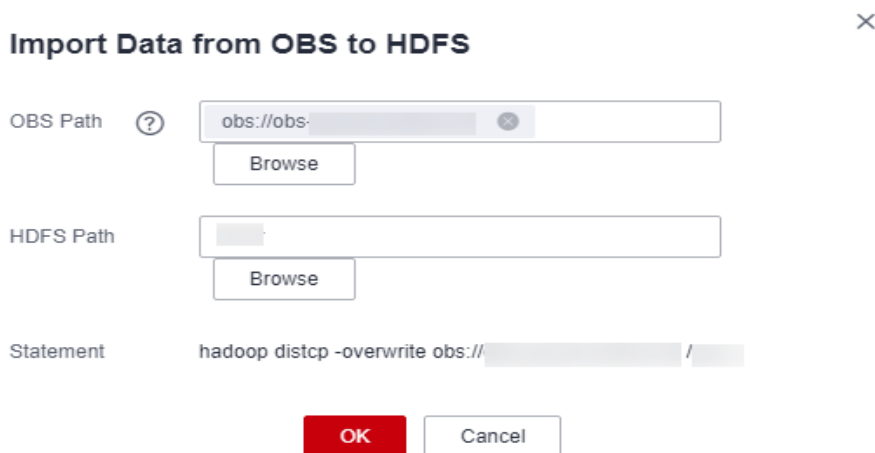
Figure 1-15 Synchronizing IAM users



- Upload the data file to the HDFS.
 - On the **Files** page, click the **HDFS File List** and go to the data storage directory, for example, **/tmp/test**.
The **/tmp/test** directory is only an example. You can use any directory on the page or create a new one.
 - Click **Import Data**.
 - OBS Path:** Select the created OBS bucket name, find the **book_score.txt** file, select **I confirm that the selected script is secure, and I understand the potential risks and accept the possible exceptions or impacts on the cluster**, and click **OK**.

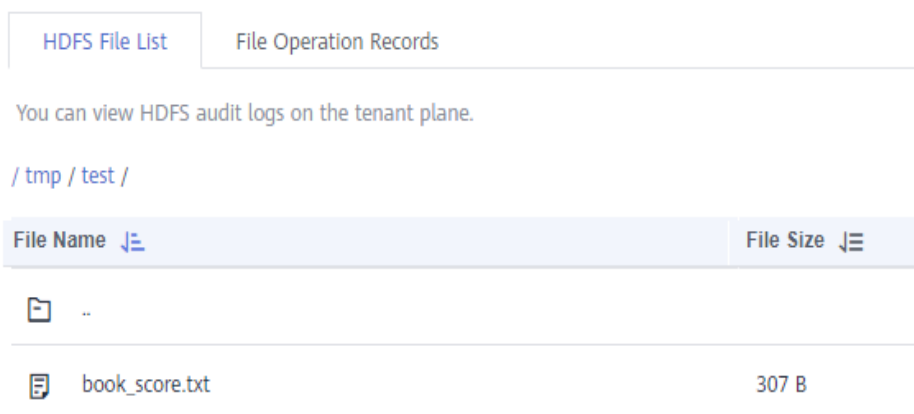
- **HDFS Path:** Select the `/tmp/test` directory and click **OK**.

Figure 1-16 Importing data from OBS to HDFS



- c. Click **OK**. After the data is imported, the data file has been uploaded to HDFS of the MRS cluster.

Figure 1-17 Data imported



Creating a Hive Table

1. Download the cluster client, and install it, for example, in the `/opt/client` directory of the active master node.
You can also use the cluster client provided in the `/opt/Bigdata/client` directory of the master node.
2. Bind an EIP to the active master node and enable port 22 in the security group. Then, log in to the active master node as user **root**, go to the directory where the client is located, and load variables.
cd /opt/client
source bigdata_env
3. Run the **beeline -n'hdfs'** command to go to the Hive Beeline page.
Run the following command to create a Hive table whose fields match the raw data fields:

create table bookscore (userid int,bookid int,score int,remarks string) row format delimited fields terminated by ',' stored as textfile;

- Run the following command to check whether the table is successfully created:

show tables;

```
+-----+
| tab_name |
+-----+
| bookscore |
+-----+
```

Importing Raw Data to Hive for Analysis

- Run the following command on Hive Beeline to import the raw data that has been imported to HDFS to the Hive table:

load data inpath '/tmp/test/book_score.txt' into table bookscore;

- After data is imported, run the following command to view content in the Hive table:

select * from bookscore;

```
+-----+-----+-----+-----+
| bookscore.userid | bookscore.bookid | bookscore.score | bookscore.remarks |
+-----+-----+-----+-----+
| 202001           | 242              | 3               | Good!              |
| 202002           | 302              | 3               | Test.              |
| 202003           | 377              | 1               | Bad!               |
| 220204           | 51               | 2               | Bad!               |
| 202005           | 346              | 1               | aaa                |
| 202006           | 474              | 4               | None               |
| 202007           | 265              | 2               | Bad!               |
| 202008           | 465              | 5               | Good!              |
| 202009           | 451              | 3               | Bad!               |
| 202010           | 86               | 3               | Bad!               |
| 202011           | 257              | 2               | Bad!               |
| 202012           | 465              | 4               | Good!              |
| 202013           | 465              | 4               | Good!              |
| 202014           | 465              | 4               | Good!              |
| 202015           | 302              | 5               | Good!              |
| 202016           | 302              | 3               | Good!              |
...

```

Run the following command to count the number of rows in the table:

select count(*) from bookscore;

```
+-----+
| _c0 |
+-----+
| 32 |
+-----+
```

- Run the following command to filter the top 3 books with the highest scores in the raw data after the MapReduce task is complete:

select bookid,sum(score) as summarize from bookscore group by bookid order by summarize desc limit 3;

Finally, the following information is displayed:

```
...
INFO : 2021-10-14 19:53:42,427 Stage-2 map = 0%, reduce = 0%
INFO : 2021-10-14 19:53:49,572 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.15 sec
INFO : 2021-10-14 19:53:56,713 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 4.19 sec
INFO : MapReduce Total cumulative CPU time: 4 seconds 190 msec
INFO : Ended Job = job_1634197207682_0025
INFO : MapReduce Jobs Launched:
```

```
INFO : Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.24 sec HDFS Read: 7872 HDFS Write:
322 SUCCESS
INFO : Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.19 sec HDFS Read: 5965 HDFS Write:
143 SUCCESS
INFO : Total MapReduce CPU Time Spent: 8 seconds 430 msec
INFO : Completed executing
command(queryId=omm_20211014195310_cf669633-5b58-4bd5-9837-73286ea83409); Time taken:
47.388 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+-----+
| bookid | summarize |
+-----+-----+
| 465    | 170        |
| 302    | 110        |
| 474    | 88         |
+-----+-----+
3 rows selected (47.469 seconds)
```

The books whose IDs are 456, 302, and 474 are the top 3 books with the highest scores.

1.3 Using Hive to Load OBS Data and Analyze Enterprise Employee Information

MRS Hadoop analysis cluster provides Hive and Spark for storing, computing, and querying massive amounts of offline as well as distributed data.

This practice describes how to import and analyze raw data stored in OBS using Hive after you create an MRS cluster and how to implement elastic and low-cost big data analysis based on storage-compute decoupling.

You can get started by reading the following topics:

1. [Creating an MRS Offline Query Cluster](#)
2. [Creating an OBS Agency and Binding It to an MRS Cluster](#)
3. [Creating a Hive Table and Loading Data from OBS](#)
4. [Analyzing Data Based on HQL](#)

Scenario

Hive is a data warehouse built on Hadoop. It provides batch computing capability for the big data platform and is able to batch analyze and summarize structured and semi-structured data for data calculation. Hive operates structured data using HQL, a SQL-like language. HQL is automatically converted into MapReduce tasks for the query and analysis of massive data in the Hadoop cluster.

Hive is able to:

- Analyze massive structured data and summarizes analysis results.
- Allow complex MapReduce jobs to be compiled in SQL languages.
- Support flexible data storage formats, including JSON, CSV, TextFile, RCFile, SequenceFile, and ORC.

This practice describes how to develop a Hive data analysis application and how to run HQL statements to access Hive data stored in OBS after you connect to Hive through the client. For example, manage and query enterprise employee

information. If you need to develop and build your application based on the sample code project provided by MRS, see [Application Development Overview](#).

In this practice, the raw data of employee information includes the following two tables:

Table 1-7 Employee information

ID	Name	Salary Currency	Salary	Tax Category	Work Place	Hire Date
1	Wang	R	8000.01	personal income tax&0.05	China:Shenzhen	2014
3	Tom	D	12000.02	personal income tax&0.09	America:NewYork	2014
4	Jack	D	24000.03	personal income tax&0.09	America:Manhattan	2015
6	Linda	D	36000.04	personal income tax&0.09	America:NewYork	2014
8	Zhang	R	9000.05	personal income tax&0.05	China:Shanghai	2014

Table 1-8 Employee contact information

ID	Mobile Number	Email Addresses
1	135 XXXX XXXX	xxxx@xx.com
3	159 XXXX XXXX	xxxxx@xx.com.cn
4	186 XXXX XXXX	xxxx@xx.org
6	189 XXXX XXXX	xxxx@xxx.cn
8	134 XXXX XXXX	xxxx@xxxx.cn

You can perform the following analysis through a data application:

- Query contact information of employees whose salaries are paid in USD.
- Query the IDs and names of employees who were hired in 2014, and load the query results to a new table.
- Collect the number of employee information records.
- Query information about employees whose email addresses end with "cn".

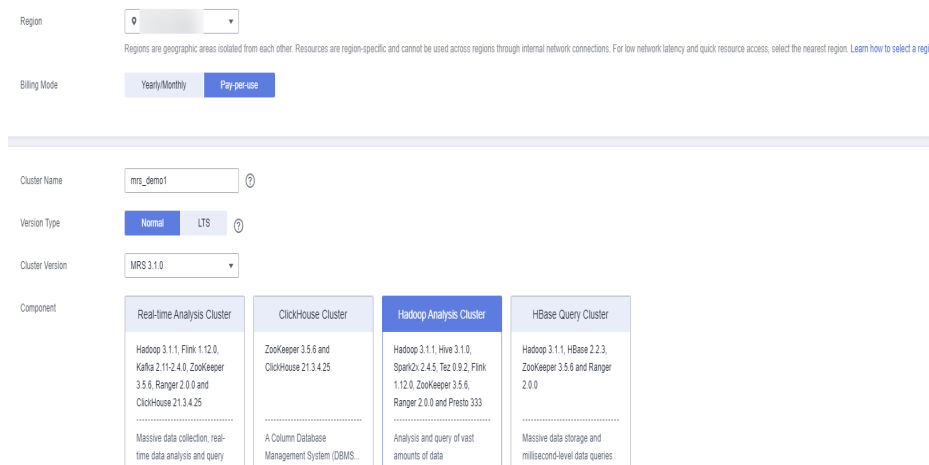
Creating an MRS Offline Query Cluster

1. Go to the [Buy Cluster](#) page.
2. Click the **Quick Config** tab and set configuration parameters.

Table 1-9 Software parameters (for reference only)

Parameter	Value
Region	EU-Dublin
Billing Mode	Pay-per-use
Cluster Name	MRS_demo
Version Type	Normal
Cluster Version	MRS 3.1.0
Component	Hadoop Analysis Cluster
AZ	AZ1
VPC	vpc-01
Subnet	subnet-01
Enterprise Project	default
Kerberos Authentication	Disabled
Username	root/admin
Password	Set the password for logging in to the cluster management page and ECS node, for example, Test!@12345 .
Confirm Password	Enter the password again.
Secure Communications	Select Enable .

Figure 1-18 Buying a Hadoop analysis cluster



3. Click **Buy Now** and wait until the MRS cluster is created.

Figure 1-19 Cluster created

Name/ID	Cluster Version	Cluster Type	Nodes	Status
mrs_7beac1fb-c54f-4769-bc3f-8b09583c9293	MRS 3.1.0	Analysis Cluster	5	Running

Creating an OBS Agency and Binding It to an MRS Cluster

NOTE

- MRS presets **MRS_ECS_DEFAULT_AGENCY** in the agency list of IAM so that you can select this agency when creating a custom cluster. This agency has the **OBSOperateAccess** permissions and the **CESFullAccess** (only available for users who have enabled fine-grained policies), **CES Administrator**, and **KMS Administrator** permissions in the region where the cluster resides.
 - If you want to use a custom agency, perform the following steps to create an agency. (To create or modify an agency, you must have the **Security Administrator** permission.)
1. Log in to the HUAWEI CLOUD management console.
 2. Choose **Service List > Management & Deployment > Identity and Access Management**.
 3. In the navigation pane on the left, choose **Agencies**. On the displayed page, click **Create Agency**.
 4. Set **Agency Name**, select **Cloud service** for **Agency Type**, and select **ECS BMS** for **Cloud Service** to authorize ECS or BMS to invoke OBS.
 5. Set **Validity Period** to **Unlimited** and click **Next**.

Figure 1-20 Creating an agency

* Agency Name:

* Agency Type: Account
 Delegate another HUAWEI CLOUD account to perform operations on your resources.
 Cloud service
 Delegate a cloud service to access your resources in other cloud services.

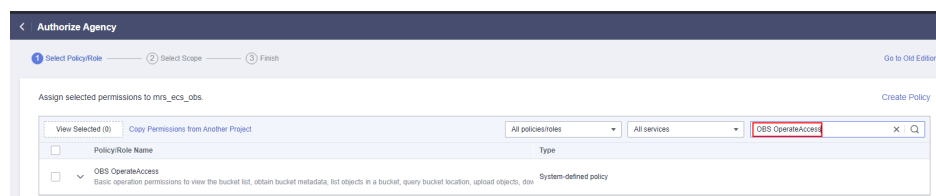
* Cloud Service:

* Validity Period:

Description: 0/255

- On the displayed page, search for the **OBS OperateAccess** policy and select it.

Figure 1-21 Assigning permissions



- Click **Next**. On the displayed page, select the desired scope for permissions you selected. By default, **All resources** is selected. Click **Show More** and select **Global resources**.
- In the dialog box that is displayed, click **OK** to start authorization. After the message "**Authorization successful.**" is displayed, click **Finish**. The agency is successfully created.
- Switch back to the MRS console and click the name of the created MRS cluster. On the **Dashboard** page, click **Manage Agency**, select the created OBS agency, and click **OK**.

Figure 1-22 Dashboard tab page of the MRS cluster

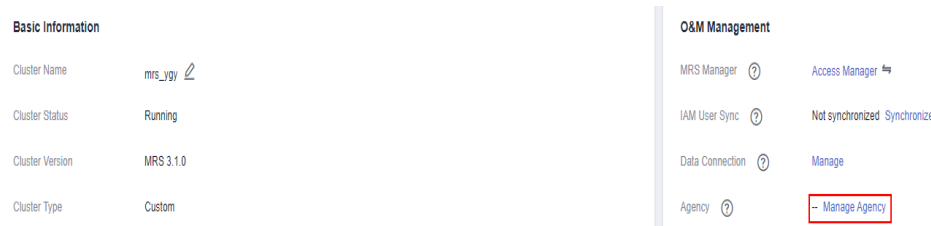
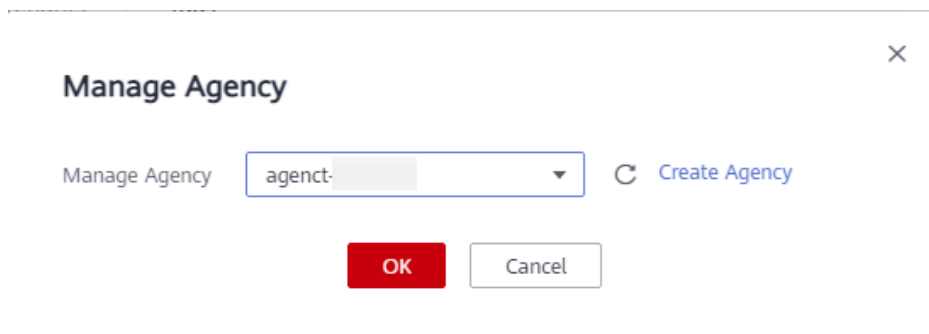


Figure 1-23 Binding an agency to an MRS cluster



Creating a Hive Table and Loading Data from OBS

1. Choose **Service List > Object Storage Service**. In the navigation pane on the left, choose **Parallel File Systems** and click **Create Parallel File System**, set the following parameters, and click **Create Now**.

Table 1-10 Parallel file system parameters

Parameter	Value
Region	EU-Dublin
File System Name	hiveobs
Policy	Private
Direct Reading	Disable
Enterprise Project	default
Tags	-

2. Download the MRS cluster client, and install it, for example, in the **/opt/client** directory of the active master node.
You can also use the cluster client provided in the **/opt/Bigdata/client** directory of the master node.
3. Bind an EIP to the active master node and enable port 22 in the security group. Then, log in to the active master node as user **root**, go to the directory where the client is located, and load variables.

```
cd /opt/client
source bigdata_env
```


4. Run the **beeline** command to go to the Hive Beeline page.

Run the following command to create an employee information data table **employees_info** that matches the raw data fields:

```
create external table if not exists employees_info
```

```
(
```

```
id INT,
```

```
name STRING,
```

```
usd_flag STRING,
```

```
salary DOUBLE,
```

```
deductions MAP<STRING, DOUBLE>,
```

```
address STRING,
```

```
entrytime STRING
```

```
)
```

```
row format delimited fields terminated by ',' map keys terminated by '&'
```

```
stored as textfile
```

```
location 'obs://hiveobs/employees_info';
```

Run the following command to create an employee contact information table **employees_contact** that matches the raw data fields:

```
create external table if not exists employees_contact
```

```
(
```

```
id INT,
```

```
phone STRING,
```

```
email STRING
```

```
)
```

```
row format delimited fields terminated by ','
```

```
stored as textfile
```

```
location 'obs://hiveobs/employees_contact';
```

5. Run the following command to check whether the table is successfully created:

```
show tables;
```

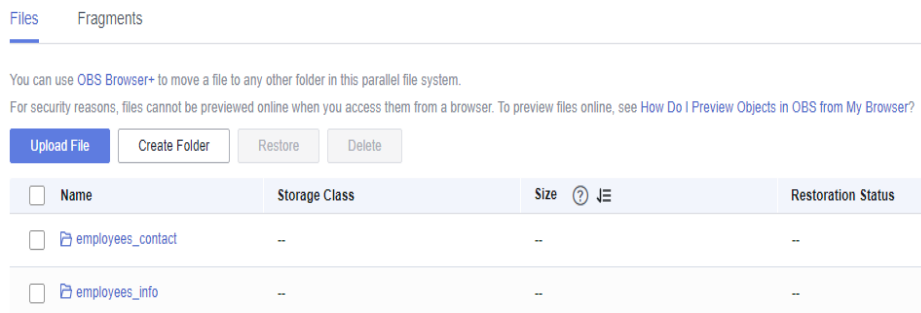
```
+-----+
|  tab_name  |
+-----+
| employees_contact |
| employees_info   |
+-----+
```

6. Import data to the corresponding OBS table directory.

By default, a folder is created in the specified storage space for a Hive internal table. The table can read data that matches the table structure as long as the file is stored in the folder.

Log in to the OBS Console. On the **Files** page of the created file system, upload the local raw data to the **employees_info** and **employees_contact** folders.

Figure 1-24 Uploading data



The following is an example of the raw data format:

info.txt:

```
1,Wang,R,8000.01,personal income tax&0.05,China:Shenzhen,2014
3,Tom,D,12000.02,personal income tax&0.09,America:NewYork,2014
4,Jack,D,24000.03,personal income tax&0.09,America:Manhattan,2015
6,Linda,D,36000.04,personal income tax&0.09,America:NewYork,2014
8,Zhang,R,9000.05,personal income tax&0.05,China:Shanghai,2014
```

contact.txt:

```
1,135 XXXX XXXX,xxxx@xx.com
3,159 XXXX XXXX,xxxx@xx.com.cn
4,189 XXXX XXXX,xxxx@xx.org
6,189 XXXX XXXX,xxxx@xx.cn
8,134 XXXX XXXX,xxxx@xxx.cn
```

- Run the following command on the Hive Beeline client to check whether the source data is correctly loaded:

select * from employees_info;

```
+-----+-----+-----+-----+
| employees_info.id | employees_info.name | employees_info.usd_flag | employees_info.salary |
employees_info.deductions | employees_info.address | employees_info.entrytime |
+-----+-----+-----+-----+
| 1 | Wang | R | 8000.01 | {"personal income
tax":0.05} | China:Shenzhen | 2014 |
| 3 | Tom | D | 12000.02 | {"personal income
tax":0.09} | America:NewYork | 2014 |
| 4 | Jack | D | 24000.03 | {"personal income tax":0.09}
| America:Manhattan | 2015 |
| 6 | Linda | D | 36000.04 | {"personal income
tax":0.09} | America:NewYork | 2014 |
| 8 | Zhang | R | 9000.05 | {"personal income
tax":0.05} | China:Shanghai | 2014 |
+-----+-----+-----+-----+
```

select * from employees_contact;

```
+-----+-----+-----+
| employees_contact.id | employees_contact.phone | employees_contact.email |
+-----+-----+-----+
| 1 | 135 XXXX XXXX | xxx@xx.com |
| 3 | 159 XXXX XXXX | xxx@xx.com.cn |
| 4 | 186 XXXX XXXX | xxx@xx.org |
| 6 | 189 XXXX XXXX | xxx@xx.cn |
| 8 | 134 XXXX XXXX | xxx@xxx.cn |
+-----+-----+-----+
```

Analyzing Data Based on HQL

On the Hive Beeline client, run the HQL statements to analyze the raw data.

1. Query contact information of employees whose salaries are paid in USD.

Run the following command to create a data table for data cleansing:

```
create table employees_info_v2 as select id, name,
regexp_replace(usr_flag, '\s+', '') as usr_flag, salary, deductions, address,
entrytime from employees_info;
```

After the Map task is complete, run the following command:

```
select a.* from employees_info_v2 a inner join employees_contact b on
a.id = b.id where a.usr_flag='D';
```

```
INFO : MapReduce Jobs Launched:
INFO : Stage-Stage-3: Map: 1 Cumulative CPU: 2.95 sec HDFS Read: 8483 HDFS Write: 317
SUCCESS
INFO : Total MapReduce CPU Time Spent: 2 seconds 950 msec
INFO : Completed executing command(queryId=omm_20211022162303_c26d4f1b-
a577-4d6c-919c-6cb96095b24b); Time taken: 26.259 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+-----+-----+-----+-----+-----+
| a.id | a.name | a.usr_flag | a.salary | a.deductions | a.address | a.entrytime |
+-----+-----+-----+-----+-----+-----+
| 3 | Tom | D | 12000.02 | {"personal income tax":0.09} | America:NewYork | 2014 |
| 4 | Jack | D | 24000.03 | {"personal income tax":0.09} | America:Manhattan | 2015 |
| 6 | Linda | D | 36000.04 | {"personal income tax":0.09} | America:NewYork | 2014 |
+-----+-----+-----+-----+-----+-----+
3 rows selected (26.439 seconds)
```

2. Query the IDs and names of employees who were hired in 2014, and load the query results to the partition with the hire date of 2014 in the **employees_info_extended** table.

Run the following to create a table:

```
create table if not exists employees_info_extended (id int, name string,
usr_flag string, salary double, deductions map<string, double>, address
string) partitioned by (entrytime string) stored as textfile;
```

Run the following command to write data into the table:

```
insert into employees_info_extended partition(entrytime='2014') select
id,name,usr_flag,salary,deductions,address from employees_info_v2
where entrytime = '2014';
```

After data is extracted, run the following command to query the data:

```
select * from employees_info_extended;
```

```
+-----+-----+-----+-----+-----+-----+
| employees_info_extended.id | employees_info_extended.name | employees_info_extended.usr_flag |
employees_info_extended.salary | employees_info_extended.deductions |
employees_info_extended.address | employees_info_extended.entrytime |
+-----+-----+-----+-----+-----+-----+
| 1 | Wang | R | 8000.01 | | |
{"personal income tax":0.05} | China:Shenzhen | 2014 | |
| 3 | Tom | D | 12000.02 | | |
{"personal income tax":0.09} | America:NewYork | 2014 | |
| 6 | Linda | D | 36000.04 | | |
{"personal income tax":0.09} | America:NewYork | 2014 | |
| 8 | Zhang | R | 9000.05 | | |
{"personal income tax":0.05} | China:Shanghai | 2014 | |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

3. Run the following command to collect the number of employee information records:

```
select count(1) from employees_info_v2;
```

```
+-----+  
|_c0 |  
+-----+  
| 5 |  
+-----+
```

4. Run the following command to query information about employees whose email addresses end with "cn":

```
select a.*, b.email from employees_info_v2 a inner join employees_contact  
b on a.id = b.id where b.email rlike '.*cn$';
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| a.id | a.name | a.usd_flag | a.salary | a.deductions | a.address | a.entrytime |  
b.email |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
+-----+  
| 3 | Tom | D | 12000.02 | {"personal income tax":0.09} | America:NewYork | 2014 |  
xxxx@xx.com.cn |  
| 6 | Linda | D | 36000.04 | {"personal income tax":0.09} | America:NewYork | 2014 |  
xxxx@xx.cn |  
| 8 | Zhang | R | 9000.05 | {"personal income tax":0.05} | China:Shanghai | 2014 |  
xxxx@xxx.cn |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
+-----+
```

1.4 Using Flink Jobs to Process OBS Data

Application Scenarios

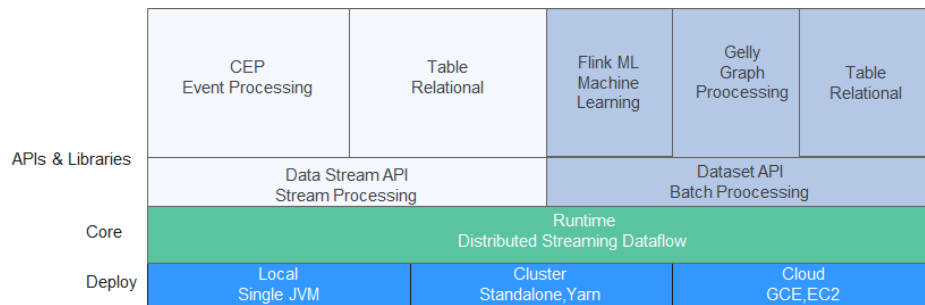
MRS supports decoupled storage and compute in scenarios where a large storage capacity is required and compute resources need to be scaled on demand. This allows you to store your data in OBS and use an MRS cluster only for data computing.

This practice instructs you on how to run Flink jobs in an MRS cluster to process data stored in OBS.

Solution Architecture

Flink is a unified computing framework that supports both batch processing and stream processing. It provides a stream data processing engine that supports data distribution and parallel computing. Flink features stream processing and is a top open-source stream processing engine in the industry.

Flink provides high-concurrency pipeline data processing, millisecond-level latency, and high reliability, making it suitable for low-latency data processing.

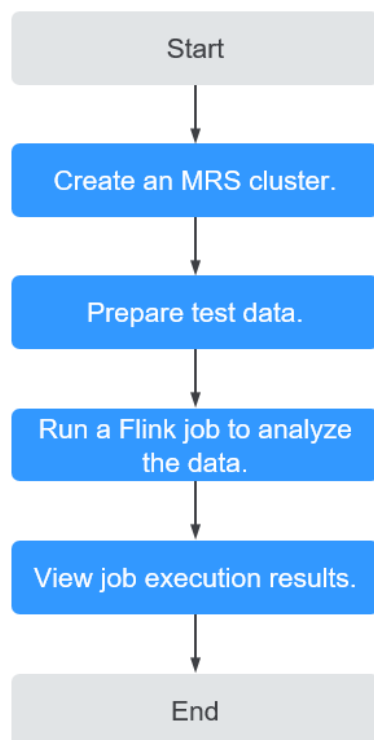


In this example, the Flink WordCount job program built in the MRS cluster is used to analyze the source data stored in the OBS file system and compute the frequency of words in the source data. For details about the program code, see <https://github.com/apache/flink/tree/master/flink-examples/flink-examples-batch/src/main/java/org/apache/flink/examples/java/wordcount>.

You can also obtain [MRS sample code project](#) and develop other Flink stream job programs by referring to [Flink Development Guide](#).

Procedure

The operation process is as follows:



Step 1: Creating an MRS Cluster

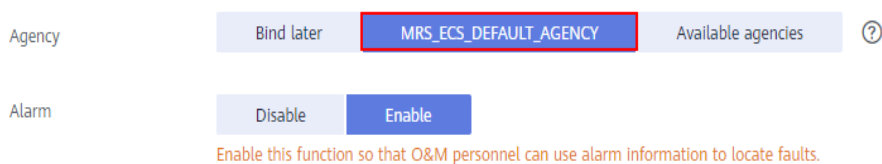
Create and purchase an MRS cluster that contains the Flink component. For details, see [Buying a Custom Cluster](#).

NOTE

In this practice, an MRS 3.1.0 cluster with Kerberos authentication disabled is used as an example.

In this example, before you analyze data stored in OBS, bind an IAM agency to the MRS cluster so that cluster components can connect to the OBS file system and have operation permissions on file system directories.

You can select the default **MRS_ECS_DEFAULT_AGENCY** agency or create a custom agency that has the permission to operate the OBS file system.



After the cluster is purchased, install the cluster client on any node of the cluster as user **omm**. For details, see [Installing and Using the Cluster Client](#).

Assume that the client is installed in **/opt/client**.

Step 2: Preparing Test Data

Before you create a Flink job for data analysis, prepare test data to be analyzed and upload the data to OBS.

Step 1 Create a file named **mrs_flink_test.txt** on your local PC. For example, the file content is as follows:

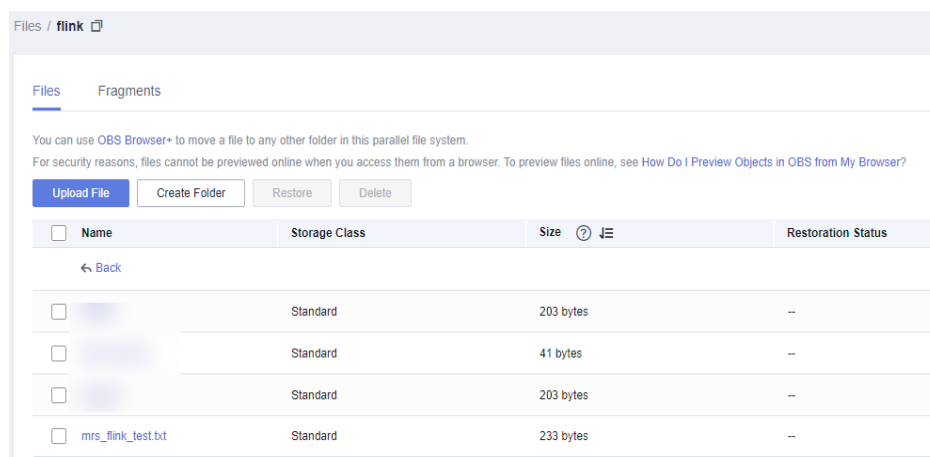
This is a test demo for MRS Flink. Flink is a unified computing framework that supports both batch processing and stream processing. It provides a stream data processing engine that supports data distribution and parallel computing.

Step 2 Choose **Service List > Storage > Object Storage Service**.

Step 3 On the OBS management console that is displayed, choose **Parallel File Systems** in the navigation pane on the left. On the page displayed, click **Create Parallel File System** and set required parameters to create a parallel file system. After the system is created, upload the test data to it.

For example, if the created file system is named **mrs-demo-data**, click the system name, and click the **Files** tab. On this tab page, click **Create Folder** to create a folder named **flink** and upload the test data to the folder.

In this example, the complete path of the test data is **obs://mrs-demo-data/flink/mrs_flink_test.txt**.

Figure 1-25 Uploading test data

Step 4 (Optional) Uploading Data Analysis Applications

You can upload the JAR files of the Flink applications developed by yourself to OBS or HDFS of the MRS cluster.

In this example, the Flink WordCount sample program built in the MRS cluster is used. You can obtain the sample program from the MRS cluster client installation directory, that is, `/opt/client/Flink/flink/examples/batch/WordCount.jar`.

Upload **WordCount.jar** to the `mrs-demo-data/program` directory.

----End

Step 3: Creating and Running a Flink Job

Method 1: Submit a job online on the console.

- Step 1** Log in to the MRS management console and click the cluster name to go to the cluster details page.
- Step 2** On the **Dashboard** tab page, click **Synchronize** next to **IAM User Sync** to synchronize IAM users.
- Step 3** Click the **Jobs** tab.
- Step 4** Click **Create**. In the **Create Job** dialog box that is displayed, set the following parameters to create a Flink job.
 - **Type:** Select **Flink**.
 - **Name:** Customize a job name, for example, `flink_obs_test`.
 - **Program Path:** In this example, the WordCount program of the Flink client is used.
 - **Program Parameter:** Use the default value.
 - **Parameters:** Enter the input and output parameters of the application. The **input** parameter indicates the test data to be analyzed, and the **output** parameter indicates the result output file.
In this example, set this parameter to `--input obs://mrs-demo-data/flink/mrs_flink_test.txt --output obs://mrs-demo-data/flink/output`.

- **Service Parameter:** Use the default values. For details about how to manually configure job parameters, see [Running a Flink Job](#).

Create Job

* Type:

* Name:

* Program Path:

Program Parameter:

Parameters:

Service Parameter:

Command Reference:

```
flink run -d -m yarn-cluster obs://mrs-demo-data/program/WordCount.jar
--input obs://mrs-demo-data/flink/mrs_flink_test.txt --output
obs://mrs-demo-data/flink/output
```

Step 5 Confirm the job configuration information and click **OK**.

----End

Method 2: Submit a job using the cluster client.

Step 1 Log in to the node where the cluster client is installed as user **root** and go to the client installation directory.

```
su - omm
```

```
cd /opt/client
```

```
source bigdata_env
```

Step 2 Run the following command to check whether the cluster can access OBS:

```
hdfs dfs -ls obs://mrs-demo-data/flink
```

Step 3 Submit a Flink job and specify the source file data for consumption.

```
flink run -m yarn-cluster /opt/client/Flink/flink/examples/batch/
WordCount.jar --input obs://mrs-demo-data/flink/mrs_flink_test.txt --output
obs://mrs-demo/data/flink/output2
```

```
...
Cluster started: Yarn cluster with application id application_1654672374562_0011
Job has been submitted with JobID a89b561de5d0298cb2ba01fbc30338bc
Program execution finished
Job with JobID a89b561de5d0298cb2ba01fbc30338bc has finished.
Job Runtime: 1200 ms
```

----End

Step 4: Viewing Job Execution Results

- Step 1** After the job is submitted, log in to FusionInsight Manager of the MRS cluster and choose **Cluster > Services > Yarn**.
- Step 2** Click the link next to **ResourceManager WebUI** to access the native Yarn web UI. On the **All Applications** page that is displayed, choose **Applications** on the left, and view the job running status and run logs.



▼ Cluster

- [About](#)
- [Nodes](#)
- [Node Labels](#)
- [Applications](#)
- [NEW](#)
- [NEW SAVING](#)
- [SUBMITTED](#)
- [ACCEPTED](#)
- [RUNNING](#)
- [FINISHED](#)
- [FAILED](#)
- [KILLED](#)
- [Scheduler](#)

► Tools

Cluster Metrics

Apps Submitted	Apps Pending
5	0

Cluster Nodes Metrics

Active Nodes	Decommissioned Nodes
3	0

User Metrics for developer

Apps Submitted	Apps Pending	Apps Running
5	0	0

Scheduler Metrics

Scheduler Type	Scheduling Request
SuperiorYarnScheduler	[yarn.io/gpu, memory-r...

- Step 3** After the job execution is complete, you can view the data analysis result in the specified result output file in the OBS file system.

File Name	Format	Size	Permissions	Created Time	Actions
mrs_flink_test.txt	Standard	233 bytes	-	Jun 10, 2022 14:39:08 GMT+08:00	Download Share More
output	Standard	203 bytes	-	Jun 16, 2022 16:04:47 GMT+08:00	Download Share More

Download the output file to your local PC and open the file to view the analysis result.

```
a 3
and 2
batch 1
both 1
computing 2
data 2
demo 1
distribution 1
engine 1
flink 2
for 1
framework 1
is 2
it 1
mrs 1
parallel 1
processing 3
provides 1
stream 2
```

```
supports 2
test 1
that 2
this 1
unified 1
```

If you do not specify the output directory when submitting a job using the cluster client CLI, you can view the data analysis result on the job running page.

```
Job with JobID xxx has finished.
Job Runtime: xxx ms
Accumulator Results:
- e6209f96ffa423974f8c7043821814e9 (java.util.ArrayList) [31 elements]

(a,3)
(and,2)
(batch,1)
(both,1)
(computing,2)
(data,2)
(demo,1)
(distribution,1)
(engine,1)
(flink,2)
(for,1)
/framework,1)
(is,2)
(it,1)
(mrs,1)
(parallel,1)
(processing,3)
(provides,1)
(stream,2)
(supports,2)
(test,1)
(that,2)
(this,1)
(unified,1)
```

----End

1.5 Consuming Kafka Data Using Spark Streaming Jobs

Application Scenarios

Use an MRS cluster to run Spark Streaming jobs to consume Kafka data.

Assume that Kafka receives one word record every second in a service. The Spark applications developed based on service requirements implements the function of accumulating the total number of records of each word in real time.

Spark Streaming sample projects store data in and send data to Kafka.

Solution Architecture

Spark is a distributed batch processing framework. It provides analysis and mining and iterative memory computing capabilities and supports application development in multiple programming languages, including Scala, Java, and Python. Spark applies to the following scenarios:

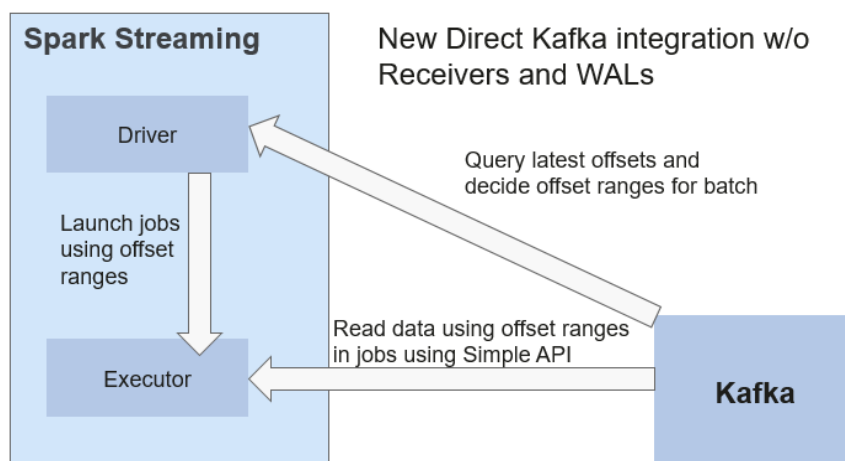
- Data processing: Spark can process data quickly and has fault tolerance and scalability.

- Iterative computation: Spark supports iterative computation to meet the requirements of multi-step data processing logic.
- Data mining: Based on massive data, Spark can handle complex data mining and analysis and support multiple data mining and machine learning algorithms.
- Streaming processing: Spark supports streaming processing with a seconds-level delay and supports multiple external data sources.
- Query analysis: Spark supports standard SQL query analysis, provides the DSL (DataFrame), and supports multiple external inputs.

Spark Streaming is a real-time computing framework built on the Spark, which expands the capability for processing massive streaming data. Spark supports two data processing approaches: Direct Streaming and Receiver.

In Direct Streaming approach, Direct API is used to process data. Take Kafka Direct API as an example. Direct API provides offset location that each batch range will read from, which is much simpler than starting a receiver to continuously receive data from Kafka and written data to WALs. Then, each batch job is running and the corresponding offset data is ready in Kafka. These offset information can be securely stored in the checkpoint file and read by applications that failed to start.

Figure 1-26 Data transmission through Direct Kafka API

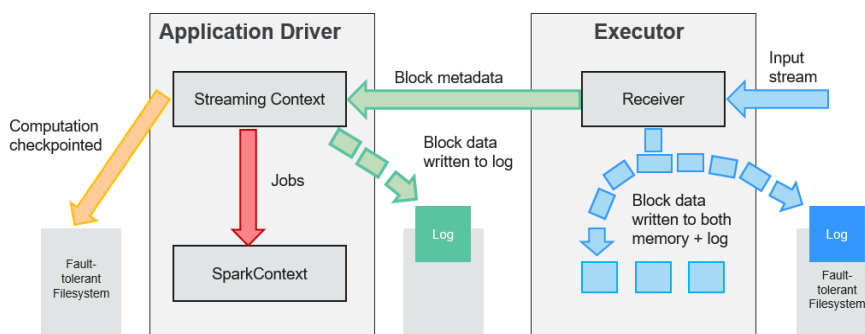


After the failure, Spark Streaming can read data from Kafka again and process the data segment. The processing result is the same no matter Spark Streaming fails or not, because the semantic is processed only once.

Direct API does not need to use the WAL and Receivers, and ensures that each Kafka record is received only once, which is more efficient. In this way, the Spark Streaming and Kafka can be well integrated, making streaming channels be featured with high fault-tolerance, high efficiency, and ease-of-use. Therefore, you are advised to use Direct Streaming to process data.

When a Spark Streaming application starts (that is, when the driver starts), the related StreamingContext (the basis of all streaming functions) uses SparkContext to start the receiver to become a long-term running task. Receiver receives and stores streaming data to the Spark memory for processing. [Figure 1-27](#) shows the data transfer lifecycle.

Figure 1-27 Data transfer lifecycle



1. Receive data (blue arrow).
Receiver divides a data stream into a series of blocks and stores them in the executor memory. In addition, after WAL is enabled, it writes data to the WAL of the fault-tolerant file system.
2. Notify the driver (green arrow).
The metadata in the received block is sent to StreamingContext in the driver. The metadata includes:
 - Block reference ID used to locate the data position in the Executor memory.
 - Block data offset information in logs (if the WAL function is enabled).
3. Process data (red arrow).
For each batch of data, StreamingContext uses block information to generate resilient distributed datasets (RDDs) and jobs. StreamingContext executes jobs by running tasks to process blocks in the executor memory.
4. Periodically set checkpoints (orange arrows).
5. For fault tolerance, StreamingContext periodically sets checkpoints and saves them to external file systems.

Procedure

Huawei Cloud MRS provides sample development projects for Spark in multiple scenarios. The development guideline for the scenario in this practice is as follows:

1. Receive data from Kafka and generate the corresponding DStream.
2. Classify word records.
3. Compute the result and print it.

Step 1: Creating an MRS Cluster

Step 1 Create and purchase an MRS cluster that contains the Spark2x and Kafka components. For details, see [Buying a Custom Cluster](#).

NOTE

In this practice, an MRS 3.1.0 cluster with Kerberos authentication disabled is used as an example.

Step 2 After the cluster is purchased, install the cluster client on any node of the cluster. For details, see [Installing and Using the Cluster Client](#).

Assume that the client is installed in `/opt/client`.

----End

Step 2: Preparing Applications

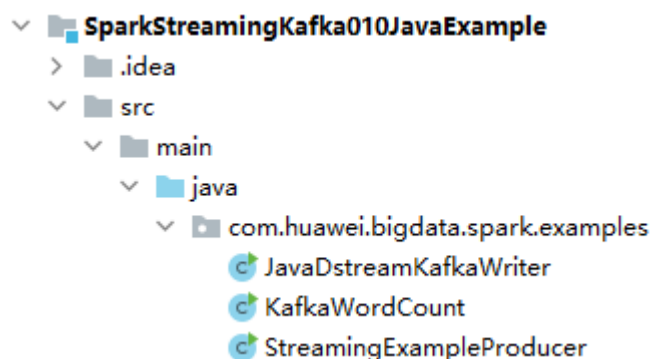
Step 1 Obtain the sample project from Huawei Mirrors.

Download the Maven project source code and configuration files of the sample project, and configure related development tools on the local host. For details, see [Obtaining Sample Projects from Huawei Mirrors](#).

Select a sample project based on the cluster version and download the sample project.

For example, to obtain `SparkStreamingKafka010JavaExample`, visit <https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.0/src/spark-examples/sparknormal-examples/SparkStreamingKafka010JavaExample>.

Step 2 Use the IDEA tool to import the sample project and wait for the Maven project to download the dependency package. For details, see [Configuring and Importing Sample Projects](#).



In this example project, Streaming is used to call the Kafka API to obtain word records, and word records are classified to obtain the number of records of each word. The key code snippets are as follows:

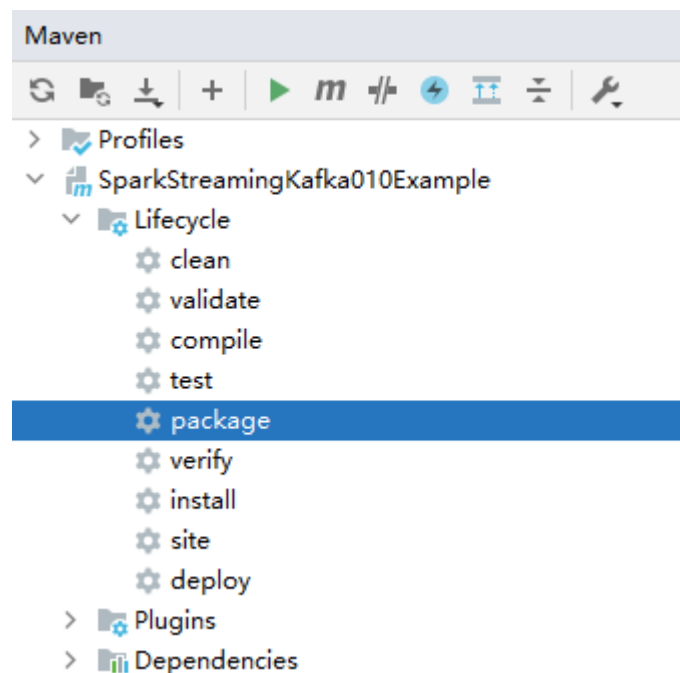
```
public class StreamingExampleProducer {
    public static void main(String[] args) throws IOException {
        if (args.length < 2) {
            printUsage();
        }
        String brokerList = args[0];
        String topic = args[1];
        String filePath = "/home/data/"; //Path for obtaining the source data
        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, brokerList);
        props.put(ProducerConfig.CLIENT_ID_CONFIG, "DemoProducer");
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
        Producer<String, String> producer = new KafkaProducer<String, String>(props);

        for (int m = 0; m < Integer.MAX_VALUE / 2; m++) {
            File dir = new File(filePath);
            File[] files = dir.listFiles();
            if (files != null) {
                for (File file : files) {
                    if (file.isDirectory()) {
                        System.out.println(file.getName() + "This is a directory!");
                    }
                }
            }
        }
    }
}
```

```
    } else {
        BufferedReader reader = null;
        reader = new BufferedReader(new FileReader(filePath + file.getName()));
        String tempString = null;
        while ((tempString = reader.readLine()) != null) {
            // Blank line judgment
            if (!tempString.isEmpty()) {
                producer.send(new ProducerRecord<String, String>(topic, tempString));
            }
        }
        // make sure the streams are closed finally.
        reader.close();
    }
}
}
try {
    Thread.sleep(3);
} catch (InterruptedException e) {
    e.printStackTrace();
}
}
}

private static void printUsage() {
    System.out.println("Usage: {brokerList} {topic}");
}
}
```

Step 3 After Maven and SDK parameters are configured on the local host, the sample project automatically loads related dependency packages. After the loading is complete, double-click **package** to obtain the JAR file.



For example, the packaged JAR file is **SparkStreamingKafka010JavaExample-1.0.jar**.

----End

Step 3: Uploading the JAR Package and Source Data

- Step 1** Prepare the source data to be sent to Kafka, for example, the following **input_data.txt** file. Upload the file to the **/home/data** directory on the client node.

```
ZhangSan  
LiSi  
WangwWU  
Tom  
Jemmy  
LinDa
```

- Step 2** Upload the compiled JAR package to a directory, for example, **/opt**, on the client node.

 **NOTE**

If you cannot directly connect to the client node to upload files through the local network, upload the JAR file or source data to OBS, import the file to HDFS on the **Files** tab page of the MRS cluster, and run the **hdfs dfs -get** command on the HDFS client to download the file to the client node.

----End

Step 4: Running the Job and Viewing the Result

- Step 1** Log in to the node where the cluster client is installed as user **root**.

```
cd /opt/client  
source bigdata_env
```

- Step 2** Create a Kafka topic for receiving data.

```
kafka-topics.sh --create --zookeeper IP address of the quorumpeer  
instance:ZooKeeper client connection port /kafka --replication-factor 2 --  
partitions 3 --topic Topic name
```

To query the IP address of the quorumpeer instance, log in to FusionInsight Manager of the cluster, choose **Cluster > Services > ZooKeeper**, and click the **Instance** tab. Use commas (,) to separate multiple IP addresses. You can query the ZooKeeper client connection port by querying the ZooKeeper service configuration parameter **clientPort**. The default value is **2181**.

For example, run the following command:

```
kafka-topics.sh --create --zookeeper 192.168.0.17:2181/kafka --replication-  
factor 2 --partitions 2 --topic sparkkafka
```

```
Created topic sparkkafka.
```

- Step 3** After the topic is created, execute the program to send data to Kafka.

```
java -cp /opt/SparkStreamingKafka010JavaExample-1.0.jar:/opt/client/  
Spark2x/spark/jars/*:/opt/client/Spark2x/spark/jars/streamingClient010/*  
com.huawei.bigdata.spark.examples.StreamingExampleProducer IP address of  
the Broker instance:Kafka connection port Topic name
```

To query the IP address of the Kafka Broker instance, log in to FusionInsight Manager of the cluster, choose **Cluster > Services > Kafka**, and click the **Instance** tab. Use commas (,) to separate multiple IP addresses. You can query the Kafka

connection port by querying the Kafka service configuration parameter **port**. The default value is **9092**.

For example, run the following command:

```
java -cp /opt/SparkStreamingKafka010JavaExample-1.0.jar:/opt/client/Spark2x/spark/jars/*:/opt/client/Spark2x/spark/jars/streamingClient010/* com.huawei.bigdata.spark.examples.StreamingExampleProducer 192.168.0.131:9092 sparkkafka
```

```
...
transactional.id = null
value.serializer = class org.apache.kafka.common.serialization.StringSerializer
2022-06-08 15:43:42 INFO  AppInfoParser:117 - Kafka version: xxx
2022-06-08 15:43:42 INFO  AppInfoParser:118 - Kafka commitId: xxx
2022-06-08 15:43:42 INFO  AppInfoParser:119 - Kafka startTimeMs: xxx
2022-06-08 15:43:42 INFO  Metadata:259 - [Producer clientId=DemoProducer] Cluster ID: d54RYHthSUishVb6nTHP0A
```

Step 4 Open a new client connection window and run the following commands to read data from the Kafka topic:

```
cd /opt/client/Spark2x/spark
```

```
source bigdata_env
```

```
bin/spark-submit --master yarn --deploy-mode client --jars $(files=($SPARK_HOME/jars/streamingClient010/*.jar); IFS=,; echo "${files[*]}") --class com.huawei.bigdata.spark.examples.KafkaWordCount /opt/SparkStreamingKafka010JavaExample-1.0.jar <checkpointDir> <brokers> <topic> <batchTime>
```

- **<checkpointDir>** indicates the HDFS path for backing up application results, for example, **/tmp**.
- **<brokers>** indicates the Kafka address for obtaining metadata, in the format of *IP address of the Broker instance:Kafka connection port*.
- **<topic>** indicates the topic name read from Kafka.
- **<batchTime>** indicates the interval for Streaming processing in batches, for example, **5**.

For example, run the following commands:

```
cd /opt/client/Spark2x/spark
```

```
source bigdata_env
```

```
bin/spark-submit --master yarn --deploy-mode client --jars $(files=($SPARK_HOME/jars/streamingClient010/*.jar); IFS=,; echo "${files[*]}") --class com.huawei.bigdata.spark.examples.KafkaWordCount /opt/SparkStreamingKafka010JavaExample-1.0.jar /tmp 192.168.0.131:9092 sparkkafka 5
```

After the program is executed, you can view the data statistics in Kafka.

```
.....
-----
Time: 1654674380000 ms
-----
(ZhangSan,6)
```



```
(Tom,6)
(LinDa,6)
(WangwWU,6)
(LiSi,6)
(Jemmmmy,6)
-----
Time: 1654674385000 ms
-----
(ZhangSan,717)
(Tom,717)
(LinDa,717)
(WangwWU,717)
(LiSi,717)
(Jemmmmy,717)
-----
Time: 1654674390000 ms
-----
(ZhangSan,2326)
(Tom,2326)
(LinDa,2326)
(WangwWU,2326)
(LiSi,2326)
(Jemmmmy,2326)
...
```

Step 5 Log in to FusionInsight Manager and choose **Cluster > Services > Spark2x**.

Step 6 On the **Dashboard** tab page that is displayed, click the link next to **Spark WebUI** to access the History Server web UI.

Click a job ID to view the status of the Spark Streaming job.

Spark Jobs ^(?)

User: root
Total Uptime: 7.4 min
Scheduling Mode: FIFO
Completed Jobs: 192
▶ Event Timeline
- Completed Jobs (192)

2 Pages: Jump to 1 Show 100 items in a page Go

Job ID	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
191	Streaming job from [output-operation-0, batch time: 15:52:40] print at KafkaWordCount.java:112	2022/06/08 15:53:24	9 ms	1/1 (1 skipped)	1/1 (0 skipped)
190	Streaming job from [output-operation-0, batch time: 15:52:40] print at KafkaWordCount.java:112	2022/06/08 15:53:24	19 ms	2/2	4/4
189	Streaming job from [output-operation-0, batch time: 15:52:35] print at KafkaWordCount.java:112	2022/06/08 15:53:24	8 ms	1/1	1/1
188	Streaming job from [output-operation-0, batch time: 15:52:30] print at KafkaWordCount.java:112	2022/06/08 15:53:24	67 ms	1/1 (2 skipped)	2/2 (0 skipped)
187	Streaming job from [output-operation-0, batch time: 15:52:35] print at KafkaWordCount.java:112	2022/06/08 15:53:24	23 ms	2/2 (1 skipped)	4/4 (0 skipped)
186	Streaming job from [output-operation-0, batch time: 15:52:30] print at KafkaWordCount.java:112	2022/06/08 15:52:30	15 ms	1/1 (1 skipped)	1/1 (0 skipped)

----End

1.6 Using Flume to Collect Log Files from a Specified Directory to HDFS

Application Scenarios

Flume is a distributed, reliable, and highly available system for aggregating massive logs. It can efficiently collect, aggregate, and move massive amounts of log data from different data sources and store the data in a centralized data storage system. Data senders can be customized in the system to collect data. In addition, Flume provides the capability of simply processing data and writing data to data receivers (customizable).

Flume consists of the client and server, both of which are FlumeAgents. The server corresponds to the FlumeServer instance and is directly deployed in a cluster. The

client can be deployed inside or outside the cluster. The client-side and service-side FlumeAgents work independently and provide the same functions.

The Flume client needs to be installed separately. It can be used to import data directly to components such as HDFS and Kafka of a cluster.

In this practice, the Flume component of a custom MRS cluster is used to automatically collect new files generated in the log directory of a specified node and store the files to HDFS.

Solution Architecture

A Flume-NG consists of agents. Each agent consists of three components (source, channel, and sink). A source is used for receiving data. A channel is used for transmitting data. A sink is used for sending data to the next end.

Figure 1-28 Flume-NG architecture

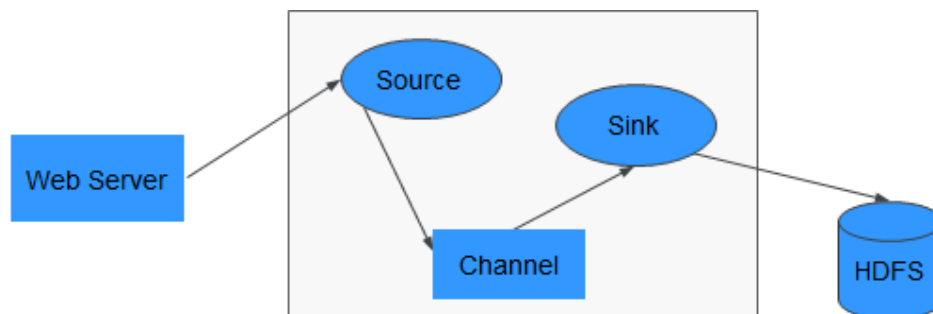


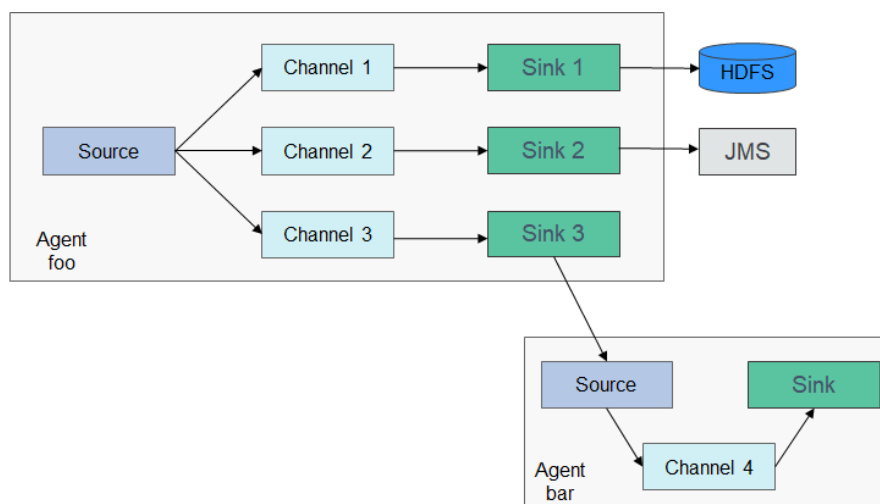
Table 1-11 Module description

Name	Description
Source	<p>A source receives data or generates data by using a special mechanism, and places the data in batches in one or more channels. The source can work in data-driven or polling mode.</p> <p>Typical source types are as follows:</p> <ul style="list-style-type: none"> • Sources that are integrated with the system, such as Syslog and Netcat • Sources that automatically generate events, such as Exec and SEQ • IPC sources that are used for communication between agents, such as Avro <p>A Source must associate with at least one channel.</p>

Name	Description
Channel	<p>A channel is used to buffer data between a source and a sink. The channel caches data from the source and deletes that data after the sink sends the data to the next-hop channel or final destination.</p> <p>Different channels provide different persistence levels.</p> <ul style="list-style-type: none"> • Memory channel: non-persistence • File channel: Write-Ahead Logging (WAL)-based persistence • JDBC channel: persistency implemented based on the embedded database <p>The channel supports the transaction feature to ensure simple sequential operations. A channel can work with sources and sinks of any quantity.</p>
Sink	<p>A sink sends data to the next-hop channel or final destination. Once completed, the transmitted data is removed from the channel.</p> <p>Typical sink types are as follows:</p> <ul style="list-style-type: none"> • Sinks that send storage data to the final destination, such as HDFS and HBase • Sinks that are consumed automatically, such as Null Sink • IPC sinks used for communication between Agents, such as Avro <p>A sink must be associated with a specific channel.</p>

As shown in [Figure 1-29](#), a Flume client can have multiple sources, channels, and sinks.

Figure 1-29 Flume structure



Step 1: Creating an MRS Cluster

Step 1 Create and purchase an MRS cluster that contains the Flume and HDFS components. For details, see [Buying a Custom Cluster](#).

 **NOTE**

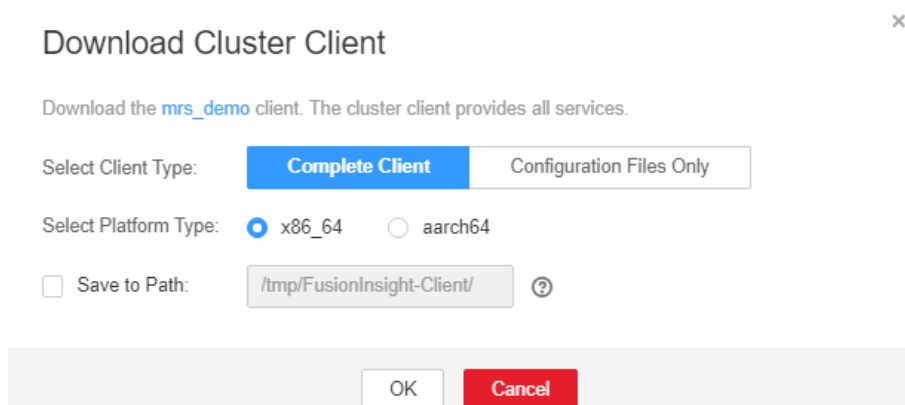
In this practice, an MRS 3.1.0 cluster with Kerberos authentication disabled is used as an example.

Step 2 After the cluster is purchased, log in to FusionInsight Manager of the cluster, download the cluster client, and decompress it.


The Flume client needs to be installed separately. You need to download the cluster client installation package to the node where the Flume client is to be installed and decompress the package.

1. On the **Homepage** page of FusionInsight Manager, click **...** next to the cluster name and click **Download Client** to download the cluster client.
2. On the **Download Cluster Client** page, enter the cluster client download information.

Figure 1-30 Downloading the cluster client



- Set **Select Client Type** to **Complete Client**.
 - Set **Select Platform Type** to the architecture of the node to install the client. **x86_64** is used as an example.
 - Select **Save to Path** and enter the download path, for example, **/tmp/FusionInsight-Client/**. Ensure that user **omm** has the operation permission on the path.
3. After the client software package is downloaded, log in to the active OMS node of the cluster as user **root** and copy the installation package to a specified node.

By default, the client software package is downloaded to the active OMS node of the cluster. You can view the node marked with  on the host page of FusionInsight Manager. If you need to install the client software package on another node in the cluster, run the following command to transfer the software package to the target node.

```
cd /tmp/FusionInsight-Client/
```

```
scp -p FusionInsight_Cluster_1_Services_Client.tar IP address of the node
where the Flume client is to be installed:/tmp
```

- Log in to the node where the Flume client is to be installed as user **root**, go to the directory where the client software package is stored, and run the following commands to decompress the software package:

```
tar -xvf FusionInsight_Cluster_1_Services_Client.tar
```

```
tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar
```

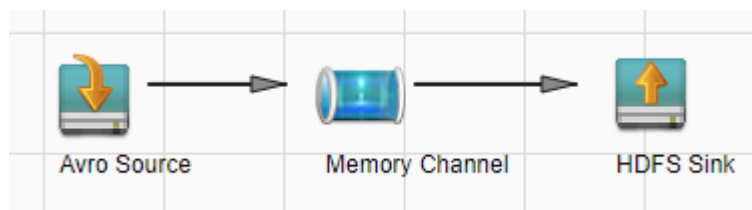
----End

Step 2: Generating the Flume Configuration File

Step 1 Log in to FusionInsight Manager and choose **Cluster > Services**. On the page that is displayed, choose **Flume**. On the displayed page, click the **Configuration Tool** tab.

Step 2 Configure and export the **properties.properties** file.

Set **Agent Name** to **server**, select **Avro Source**, **Memory Channel**, and **HDFS Sink**, and connect them.



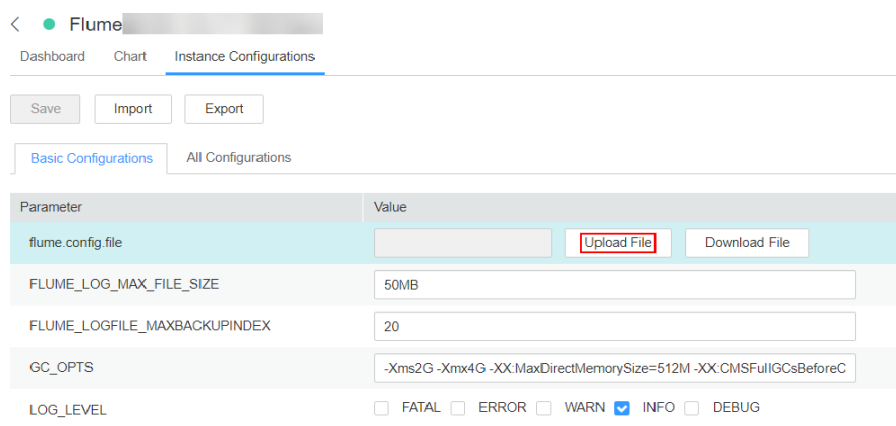
Double-click the module icon and set the parameters according to the following table. Retain the default values for the parameters not listed.

Type	Parameter	Description	Example Value
Avro Source	Name	Module name, which is customizable	test_source_1
	bind	IP address of the node where the Flume role resides. You can choose Cluster > Services > Flume > Instances to view the IP address of any Flume role instance.	192.168.10.192
	port	Connection port. The port number starts from 21154.	21154
Memory Channel	Name	Module name, which is customizable	test_channel_1
HDFS Sink	Name	Module name, which is customizable	test_sink_1
	hdfs.path	HDFS directory to which log files are written	hdfs://hacluster/flume/test

Type	Parameter	Description	Example Value
	hdfs.filePrefix	Prefix of the file name written to HDFS	over_% {basename}

Step 3 Click **Export** to download the **properties.properties** file to your local PC.

Step 4 On FusionInsight Manager, choose **Cluster > Services > Flume**, click the **Instance** tab, and click the Flume role in the row of the node where the configuration file is to be uploaded. The **Instance Configurations** tab page is displayed.



Step 5 Click **Upload File** and upload the **properties.properties** file.

Click **Save**. Then click **OK**.

Step 6 Choose **Cluster > Services > Flume**. On the page that is displayed, click the **Configuration Tool** tab.

Set **Agent Name** to **client**, select **SpoolDir Source**, **Memory Channel**, and **Avro Sink**, and connect them.



Double-click the module icon and set the parameters according to the following table. (Retain the default values for the parameters not listed.)

Type	Parameter	Description	Example Value
SpoolDir Source	Name	Module name, which is customizable	test_source_1

Type	Parameter	Description	Example Value
	spoolDir	Directory where logs need to be collected. The Flume running user must have the read and write permissions on the directory, and the permissions must be verified by storing files in the directory.	/var/log/Bigdata/audit/test
Memory Channel	Name	Module name, which is customizable	test_channel_1
HDFS Sink	Name	Module name, which is customizable	test_sink_1
	hostname	IP address of the node where the Flume role to be connected resides	192.168.10.192
	port	Connection port. The port number starts from 21154.	21154

Step 7 Click **Export** to download the **properties.properties** file to your local PC.

Step 8 Rename the **properties.properties** file as **client.properties.properties**, and upload the file to the *Path where the cluster client installation package is decompressed/Flume/FlumeClient/flume/conf* directory on the Flume client node.

----End

Step 3: Installing the Flume Client

Step 1 Log in to the node where the Flume client is to be installed as user **root**.

Step 2 Go to the path where the client installation package is decompressed. For example, the client installation package has been uploaded to **/tmp** and then decompressed.

Step 3 Run the following commands to install the Flume client. In the command, **/opt/FlumeClient** indicates the custom Flume client installation path.

```
cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_ClientConfig/Flume/FlumeClient
```

```
./install.sh -d /opt/FlumeClient -c flume/conf/client.properties.properties
```

```
CST ... [flume-client install]: install flume client successfully.
```

----End

Step 4: Viewing Log Collection Results

Step 1 After the Flume client is installed, write new log files to the log collection directory to check whether logs are transmitted.

For example, create several log files in the `/var/log/Bigdata/audit/test` directory.

```
cd /var/log/Bigdata/audit/test
```

```
vi log1.txt
```

```
Test log file 1!!!
```

```
vi log2.txt
```

```
Test log file 2!!!
```

Step 2 After the log files are written, run the `ll` command to view the file list. If the suffix `.COMPLETED` is automatically added to the file names, the log files have been collected.

```
-rw-----. 1 root root 75 Jun 9 19:59 log1.txt.COMPLETED
-rw-----. 1 root root 75 Jun 9 19:59 log2.txt.COMPLETED
```

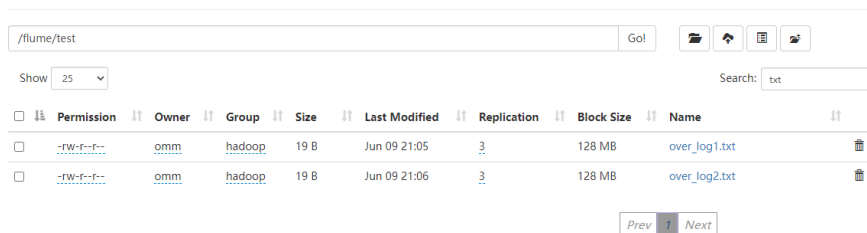
Step 3 Log in to FusionInsight Manager and choose **Cluster > Services > HDFS**. On the Dashboard tab page that is displayed, click the **NameNode(Node name,Active)** link next to **NameNode WebUI** to access the HDFS web UI.

Basic Information

Running Status:	● Normal
Configuration Status:	⊕ Synchronized
Version:	3.1.1
Read Rate:	0.00 MB/s
Write Rate:	0.00 MB/s
Safe Mode:	OFF
Disk Space:	0.19% 3GB/1.55TB <div style="width: 10%; height: 10px; background-color: #ccc; margin-top: 5px;"></div>
Missing Blocks:	0
Number of Blocks to be Replicated:	0
Damaged Blocks:	0
Normal DataNodes:	3
NameNode WebUI:	NameNode(node-master2pJgL.mrs-muix.com,Active) NameNode(node-master3pVHC.mrs-muix.com,Standby)
NameService Count:	1

Step 4 Choose **Utilities > Browse the file system** and check whether data is generated in the `/flume/test` directory in HDFS.

Browse Directory



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	omm	hadoop	19 B	Jun 09 21:05	3	128 MB	over_log1.txt
-rw-r--r--	omm	hadoop	19 B	Jun 09 21:06	3	128 MB	over_log2.txt

As shown above, log files are generated in the directory, and the prefix **over_** is added to the file names.

Download the log file **over_log1.txt** and check whether its content is the same as that of the log file **log1.txt**.

```
Test log file 1!!!
```

```
----End
```

1.7 Kafka-based WordCount Data Flow Statistics Case

Application Scenarios

Use an MRS cluster to run Kafka programs to process data.

Kafka Streams is a lightweight stream processing framework provided by Apache Kafka, where the input and output data are stored in Kafka clusters.

The following uses WordCount as an example.

Solution Architecture

Kafka is a distributed message publish-subscribe system. With features similar to JMS, Kafka processes active streaming data.

Kafka applies to many scenarios, such as message queuing, behavior tracing, O&M data monitoring, log collection, stream processing, event tracing, and log persistence.

Kafka has the following features:

- High throughput
- Message persistence to disks
- Scalable distributed system
- High fault tolerance

Procedure

Huawei Cloud MRS provides sample development projects for Kafka in multiple scenarios. The development guideline for the scenario in this practice is as follows:

1. Create two topics on the Kafka client to serve as the input and output topics.
2. Develop a Kafka Streams to implement the word count function. The system collects statistics on the number of words in each message by reading the

message in the input topic, consumes data from the output topic, and provides the statistical result in the form of a key-value pair.

Step 1: Creating an MRS Cluster

Step 1 Create and purchase an MRS cluster that contains the Kafka component. For details, see [Buying a Custom Cluster](#).

NOTE

In this practice, an MRS 3.1.0 cluster, with Hadoop and Kafka installed and with Kerberos authentication disabled, is used as an example.

Step 2 After the cluster is purchased, install the cluster client on any node of the cluster. For details, see [Installing and Using the Cluster Client](#).

For example, install the client in the `/opt/client` directory on the active management node.

Step 3 After the client is installed, create the `lib` directory on the client to store related JAR files.

Copy the Kafka JAR files in the directory decompressed during client installation to `lib`.

For example, if the download path of the client software package is `/tmp/FusionInsight-Client` on the active management node, run the following commands:

```
mkdir /opt/client/lib
```

```
cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Services_ClientConfig
```

```
scp Kafka/install_files/kafka/libs/* /opt/client/lib
```

```
----End
```

Step 2: Preparing Applications

Step 1 Obtain the sample project from Huawei Mirrors.

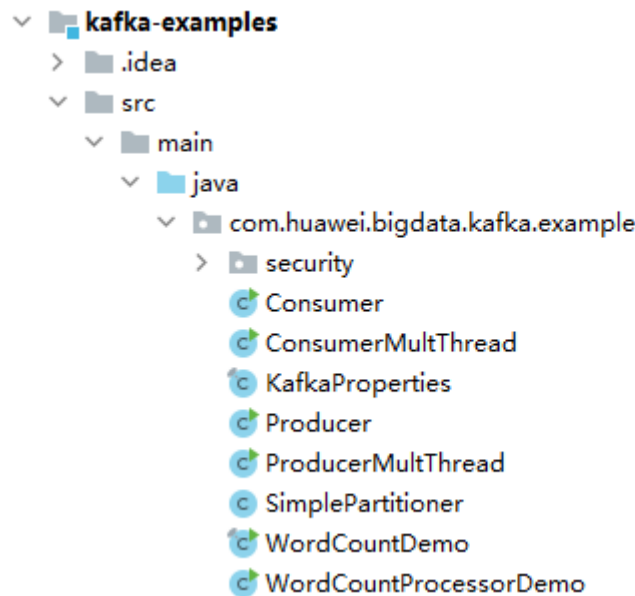
Download the Maven project source code and configuration files of the sample project, and configure related development tools on the local host. For details, see [Obtaining Sample Projects from Huawei Mirrors](#).

Select a sample project based on the cluster version and download the sample project.

For example, to obtain WordCountDemo, visit <https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.0/src/kafka-examples>.

Step 2 Use IntelliJ IDEA to import the sample project locally and wait for the Maven project to download related dependency packages.

After Maven and SDK parameters are configured on the local host, the sample project automatically loads related dependency packages. For details, see [Configuring and Importing a Sample Project](#).



In this sample program WordCountDemo, Kafka APIs are called to obtain word records, and word records are classified to obtain the number of records of each word. The key code snippets are as follows:

```

...
static Properties getStreamsConfig() {
    final Properties props = new Properties();
    KafkaProperties kafkaProc = KafkaProperties.getInstance();
    // Broker address list. Configure this parameter based on site requirements.
    props.put(BOOTSTRAP_SERVERS, kafkaProc.getValues(BOOTSTRAP_SERVERS, "node-
group-1kLfk.mrs-rbmq.com:9092"));
    props.put(SASL_KERBEROS_SERVICE_NAME, "kafka");
    props.put(KERBEROS_DOMAIN_NAME, kafkaProc.getValues(KERBEROS_DOMAIN_NAME,
"hadoop.hadoop.com"));
    props.put(APPLICATION_ID, kafkaProc.getValues(APPLICATION_ID, "streams-wordcount"));
    // Protocol type. The value can be SASL_PLAINTEXT or PLAINTEXT.
    props.put(SEcurity_PROTOCOL, kafkaProc.getValues(SEcurity_PROTOCOL, "PLAINTEXT"));
    props.put(CACHE_MAX_BYTES_BUFFERING, 0);
    props.put(DEFAULT_KEY_SERDE, Serdes.String().getClass().getName());
    props.put(DEFAULT_VALUE_SERDE, Serdes.String().getClass().getName());
    props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");
    return props;
}
static void createWordCountStream(final StreamsBuilder builder) {
    // Receives input records from the input topic.
    final KStream<String, String> source = builder.stream(INPUT_TOPIC_NAME);
    // Aggregates calculation results of the key-value pair.
    final KTable<String, Long> counts = source
        .flatMapValues(value ->
Arrays.asList(value.toLowerCase(Locale.getDefault()).split(REGEX_STRING)))
        .groupBy((key, value) -> value)
        .count();
    // Outputs the key-value pairs from the output topic.
    counts.toStream().to(OUTPUT_TOPIC_NAME, Produced.with(Serdes.String(), Serdes.Long()));
}
...

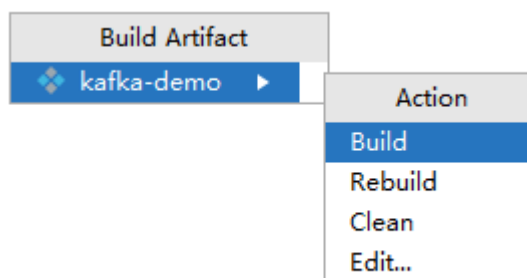
```

NOTE

- Set **BOOTSTRAP_SERVERS** to the host names and port numbers of Kafka broker nodes based on site requirements. For details about the broker information in [Commissioning an Application in Linux](#), log in to FusionInsight Manager, choose **Cluster > Services > Kafka**, and click the **Instance** tab.
- **SECURITY_PROTOCOL** indicates the protocol type for connecting to Kafka. In this example, set this parameter to **PLAINTEXT**.

Step 3 After confirming that the parameters in **WordCountDemo.java** are correct, compile the project and package it to obtain the JAR file.

For details about how to compile a JAR file, see [Commissioning an Application in Linux](#).



For example, the packaged JAR file is **kafka-demo.jar**.

----End

Step 3: Uploading the JAR File and Source Data

Step 1 Upload the compiled JAR file to a directory, for example, **/opt/client/lib**, on the client node.

NOTE

If you cannot directly connect to the client node to upload files through the local network, upload the JAR file or source data to OBS, import the file to HDFS on the **Files** tab page of the MRS cluster, and run the **hdfs dfs -get** command on the HDFS client to download the file to the client node.

----End

Step 4: Running the Job and Viewing the Result

Step 1 Log in to the node where the cluster client is installed as user **root**.

```
cd /opt/client
source bigdata_env
```

Step 2 Create an input topic and an output topic. Ensure that the topic names are the same as those specified in the sample code. Set the cleanup policy of the output topic to **compact**.

```
kafka-topics.sh --create --zookeeper IP address of the quorumpeer
instance:ZooKeeper client connection port /kafka --replication-factor 1 --
partitions 1 --topic Topic name
```

To query the IP address of the quorumpeer instance, log in to FusionInsight Manager of the cluster, choose **Cluster > Services > ZooKeeper**, and click the **Instance** tab. Use commas (,) to separate multiple IP addresses. You can query the ZooKeeper client connection port by querying the ZooKeeper service configuration parameter **clientPort**. The default value is **2181**.

For example, run the following commands:

```
kafka-topics.sh --create --zookeeper 192.168.0.17:2181/kafka --replication-factor 1 --partitions 1 --topic streams-wordcount-input
```

```
kafka-topics.sh --create --zookeeper 192.168.0.17:2181/kafka --replication-factor 1 --partitions 1 --topic streams-wordcount-output --config cleanup.policy=compact
```

Step 3 After the topics are created, run the following command to run the program:

```
java -cp ./opt/client/lib/*  
com.huawei.bigdata.kafka.example.WordCountDemo
```

Step 4 Open a new client connection window and run the following commands to use **kafka-console-producer.sh** to write messages to the input topic:

```
cd /opt/client
```

```
source bigdata_env
```

```
kafka-console-producer.sh --broker-list Broker instance IP address:Kafka connection port (For example, 192.168.0.13:9092) --topic streams-wordcount-input --producer.config /opt/client/Kafka/kafka/config/producer.properties
```

Step 5 Open a new client connection window and run the following commands to use **kafka-console-consumer.sh** to consume data from the output topic and view the statistics result:

```
cd /opt/client
```

```
source bigdata_env
```

```
kafka-console-consumer.sh --topic streams-wordcount-output --bootstrap-server Broker instance IP address:Kafka connection port --consumer.config /opt/client/Kafka/kafka/config/consumer.properties --from-beginning --property print.key=true --property print.value=true --property key.deserializer=org.apache.kafka.common.serialization.StringDeserializer --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer --formatter kafka.tools.DefaultMessageFormatter
```

Write a message to the input topic.

```
>This is Kafka Streams test  
>test starting  
>now Kafka Streams is running  
>test end
```

The message is output as follows:

```
this 1  
is 1  
kafka 1  
streams 1  
test 1
```

```
test 2
starting 1
now 1
kafka 2
streams 2
is 2
running 1
test 3
end 1
```

----End

2 Data Migration

2.1 Data Migration Solution

2.1.1 Making Preparations

This section describes how to migrate HDFS, HBase, and Hive data to an MRS cluster in different scenarios. During data migration, data may be overwritten, lost, or damaged. This document is for reference only. Please cooperate with Huawei Cloud technical personnel to formulate and implement a specific data migration solution.

Make preparations on a source cluster before data migration to prevent the source cluster from generating new data during data migration, thereby avoiding data inconsistency between the source and destination clusters after data migration. Before data migration is complete, the destination cluster must be in the initial state and cannot run any other services except data migration jobs.

Stopping Cluster Services and the Related Services

- If the Kafka service is involved in your cluster, stop all jobs that generate data in Kafka. Wait until the Kafka consumption tasks have consumed the inventory data in Kafka, and then perform the next step.
- Stop all services and jobs related to HDFS, HBase, and Hive, and stop the HBase and Hive services.

Establishing a Data Transmission Channel

- If the source cluster and destination cluster are deployed in different VPCs in the same region, create a network connection between the two VPCs to establish a data transmission channel at the network layer. For details, see [VPC Peering Connection Overview](#).
- If the source cluster and destination cluster are deployed in the same VPC but belong to different security groups, add security group rules to each security group on the VPC management console. In the security rules, **Protocol** is set to **ANY**, **Transfer Direction** is set to **Inbound**, and **Source** is set to **Security Group** (the security group of the peer cluster).

- To add an inbound rule to the security group of the source cluster, select the security group of the destination cluster in **Source**.
- To add an inbound rule to the security group of the destination cluster, select the security group of the source cluster in **Source**.
- If the source and destination clusters are deployed in the same security group of the same VPC and Kerberos authentication is enabled for both clusters, you need to configure mutual trust between the two clusters.

2.1.2 Exporting Metadata

To ensure that the data properties and permissions of the source cluster are consistent with those of the destination cluster, metadata of the source cluster needs to be exported to restore metadata after data migration. The metadata to be exported includes the owner, group, and permission information of the HDFS files and Hive table description.

Exporting HDFS Metadata

HDFS metadata information to be exported includes file and folder permissions and owner/group information. You can run the following command on the HDFS client to export the metadata:

```
$HADOOP_HOME/bin/hdfs dfs -ls -R <migrating_path> > /tmp/hdfs_meta.txt
```

The following provides description about the parameters in the preceding command.

- **\$HADOOP_HOME**: installation directory of the Hadoop client in the source cluster
- **<migrating_path>**: HDFS data directory to be migrated
- **/tmp/hdfs_meta.txt**: local path for storing the exported metadata

NOTE

If the source cluster can communicate with the destination cluster and you run the **hadoop distcp** command as a super administrator to copy data, you can add the **-p** parameter to enable DistCp to restore the metadata of the corresponding file in the destination cluster while copying data. In this case, skip this step.

Exporting Hive Metadata

Hive table data is stored in HDFS. Table data and the metadata of the table data is centrally migrated in directories by HDFS in a unified manner. Metadata of Hive tables can be stored in different types of relational databases (such as MySQL, PostgreSQL, and Oracle) based on cluster configurations. The exported metadata of the Hive tables in this document is the Hive table description stored in the relational database.

The mainstream big data release editions in the industry support Sqoop installation. For on-premises big data clusters of the community version, you can download the Sqoop of the community version for installation. Use Sqoop to decouple the strong dependency between the metadata to be exported and the relational database and export Hive metadata to HDFS and migrate it together with the table data for restoration. The procedure is as follows:

- Step 1** Download the Sqoop tool from the source cluster and install it. For details, see <http://sqoop.apache.org/>.
- Step 2** Download the JDBC driver of the relational database to the ``${Sqoop_Home}/lib` directory.
- Step 3** Run the following command to export all Hive metadata tables: All exported data is stored in the `/user/<user_name>/<table_name>` directory on HDFS.

```
`${Sqoop_Home}/bin/sqoop import --connect jdbc:<driver_type>://<ip>:<port>/<database> --table <table_name> --username <user> -password <passwd> -m 1
```

The following provides description about the parameters in the preceding command.

- ``${Sqoop_Home}`: Sqoop installation directory
- `<driver_type>`: Database type
- `<ip>`: IP address of the database in the source cluster
- `<port>`: Port number of the database in the source cluster
- `<table_name>`: Name of the table to be exported
- `<user>`: Username
- `<passwd>`: User password

NOTE

Commands carrying authentication passwords pose security risks. Disable historical command recording before running such commands to prevent information leakage.

----End

2.1.3 Copying Data

Based on the regions of and network connectivity between the source cluster and destination cluster, data copy scenarios are classified as follows:

Same Region

If the source cluster and destination cluster are in the same region, follow instructions in [Establishing a Data Transmission Channel](#) to configure the network and set up a network transmission channel. Use the DistCp tool to run the following command to copy the HDFS, HBase, Hive data files and Hive metadata backup files from the source cluster to the destination cluster.

```
`${HADOOP_HOME}/bin/hadoop distcp <src> <dist> -p
```

The following provides description about the parameters in the preceding command.

- ``${HADOOP_HOME}`: installation directory of the Hadoop client in the destination cluster
- `<src>`: HDFS directory of the source cluster
- `<dist>`: HDFS directory of the destination cluster

Migrating Data from an Offline Cluster to a Cloud

You can use the following way to migrate data from an offline cluster to the cloud.

- Direct Connect
Create a Direct Connect between the source cluster and target cluster, enable the network between the offline cluster egress gateway and the online VPC, and use DistCp to copy the data by referring to [Same Region](#).

2.1.4 Restoring Data

HDFS File Property Restoration

Based on the exported permission information, run the HDFS commands in the background of the destination cluster to restore the file permission and owner and group information.

```
$HADOOP_HOME/bin/hdfs dfs -chmod <MODE> <path>  
$HADOOP_HOME/bin/hdfs dfs -chown <OWNER> <path>
```

Hive Metadata Restoration

Install Sqoop and run the Sqoop command in the destination cluster to import the exported Hive metadata to DBService in the MRS cluster.

```
`${Sqoop_Home}/bin/sqoop export --connect jdbc:postgresql://<ip>.20051/hivemeta --table <table_name> --  
username hive -password <passwd> --export-dir <export_from>
```

The following provides description about the parameters in the preceding command.

- ``${Sqoop_Home}`: Sqoop installation directory in the destination cluster
- `<ip>`: IP address of the database in the destination cluster
- `<table_name>`: Name of the table to be restored
- `<passwd>`: Password of user **hive**
- `<export_from>`: HDFS address of the metadata in the destination cluster

NOTE

Commands carrying authentication passwords pose security risks. Disable historical command recording before running such commands to prevent information leakage.

HBase Table Reconstruction

Restart the HBase service of the destination cluster to make data migration take effect. During the restart, HBase loads the data in the current HDFS and regenerates metadata. After the restart is complete, run the following command on the Master node client to load the HBase table data:

```
`${HBase_Home}/bin/hbase hbck -fixMeta -fixAssignments
```

After the command is executed, run the following command repeatedly to check the health status of the HBase cluster until the health status is normal:

```
hbase hbck
```

2.2 Using BulkLoad to Import Data to HBase in Batches

When batch importing a large amount of data to HBase, you have many choices, for example, calling the **put** method of HBase to insert data or using MapReduce to load data from HDFS. However, the two methods cause high pressure on the RegionServer and consume a large number of CPU and network resources because of frequent flush, compact, and split operations of HBase, thereby resulting in low efficiency.

This practice describes how to import local data to HBase in batches using BulkLoad after you create an MRS cluster. This method greatly improves the write efficiency and reduces the write pressure on RegionServer nodes.

You can get started by reading the following topics:

1. [Creating an MRS Offline Query Cluster](#)
2. [Importing Local Data to HDFS](#)
3. [Creating an HBase Table](#)
4. [Generating an HFile and Importing It to HBase](#)

Scenario

BulkLoad uses MapReduce jobs to directly convert data into HFiles that comply with the internal data format of HBase, and then loads the generated StoreFiles to the corresponding nodes in a cluster. This method requires no flush, compact, or split operations, occupies no region resources, and generates little write requests. Fewer CPU and network resources are required.

Inapplicable scenarios of BulkLoad:

- Large amounts of data needs to be loaded to HBase in the one-off manner.
- When data is loaded to HBase, requirements on reliability are not high and WAL files do not need to be generated.
- When the **put** method is used to load large amounts of data to HBase, data loading and query will be slow.
- The size of an HFile generated after data loading is similar to the size of HDFS blocks.

Creating an MRS Offline Query Cluster

1. Go to the [Buy Cluster](#) page.
2. Click the **Quick Config** tab and set configuration parameters.

Table 2-1 Software configurations

Parameter	Value
Region	EU-Dublin
Billing Mode	Pay-per-use

Parameter	Value
Cluster Name	MRS_hbase
Version Type	Normal
Cluster Version	MRS 3.1.0
Component	HBase Query Cluster
AZ	AZ1
Enterprise Project	default
VPC	vpc-01
Subnet	subnet-01
Kerberos Authentication	Toggle the slider on.
Username	root/admin
Password	Set the password for logging in to the cluster management page and ECS node, for example, Test!@12345 .
Confirm Password	Enter the password again.
Secure Communications	Select Enable .

Figure 2-1 Creating an HBase query cluster

The screenshot shows the configuration steps for creating an HBase query cluster. The 'Component' section is expanded to show four options: Real-time Analysis Cluster, ClickHouse Cluster, Hadoop Analysis Cluster, and HBase Query Cluster. The 'HBase Query Cluster' option is selected, which includes Hadoop 3.1.1, HBase 2.2.3, ZooKeeper 3.5.6, and Ranger 2.0.0. Below this, other configuration fields are visible: AZ (AZ1), Enterprise Project (default), VPC (vpc-9999), and Subnet (subnet-33cf(192.168.0.0/...)).

3. Click **Buy Now** and wait until the MRS cluster is created.

Name/ID	Cluster Version	Cluster Type	Nodes	Status
mrs_7beac1fb-c54f-4769-bc3f-8b09583c9293	MRS 3.1.0	Analysis Cluster	5	Running

Importing Local Data to HDFS

1. Prepare a student information file **info.txt** on the local host.
The fields include student ID, name, birthday, gender, and address. An example file is as follows:

```
20200101245, Zhang xx, 20150324, Male, City 1
20200101246, Li xx, 20150202, Male, City 2
20200101247, Yang xx, 20151101, Female, City 3
20200101248, Chen xx, 20150218, Male, City 4
20200101249, Li xx, 20150801, Female, City 5
20200101250, Wang xx, 20150315, Male, City 6
20200101251, Li xx, 20151201, Male, City 7
20200101252, Sun xx, 20150916, Female, City 8
20200101253, Lin xx, 20150303, Male, City 9
```

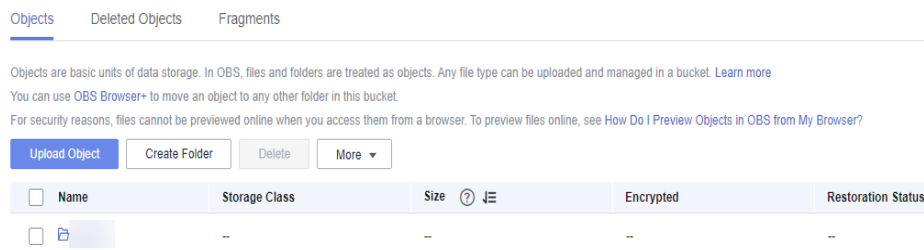
2. Log in to the OBS console, click **Create Bucket**, set the following parameters, and click **Create Now**.

Table 2-2 Bucket parameters

Parameter	Value
Region	EU-Dublin
Bucket Name	mrs-hbase
Data Redundancy Policy	Single-AZ storage
Default Storage Class	Standard
Bucket Policy	Private
Default Encryption	Disabled
Direct Reading	Disable
Enterprise Project	default
Tags	-

After the bucket is created, click the bucket name. In the navigation pane on the left, choose **Objects** and click **Upload Object** to upload the data file.

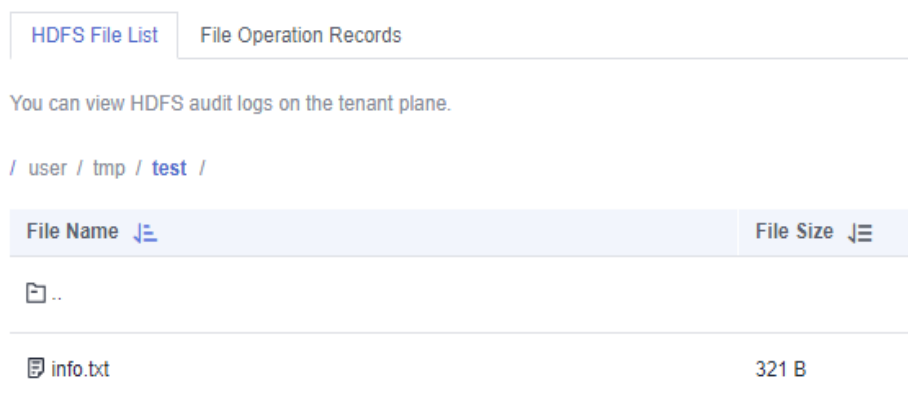
Figure 2-2 Uploading an object



3. Switch back to the MRS console and click the name of the created MRS cluster. On the **Dashboard** page, click **Synchronize** next to **IAM User Sync**. The synchronization takes about five minutes.

4. Upload the data file to the HDFS.
 - a. On the **Files** page, click the **HDFS File List** and go to the data storage directory, for example, **/tmp/test**.
The **/tmp/test** directory is only an example. You can use any directory on the page or create a new one.
 - b. Click **Import Data**.
 - **OBS Path:** Find the **info.txt** file in the created OBS bucket and click **Yes**.
 - **HDFS Path:** Select an HDFS path, for example, **/tmp/test**, and click **Yes**.
 - c. Click **OK** and wait until the data file is imported.

Figure 2-3 Importing data



Creating an HBase Table

1. Log in to FusionInsight Manager of the cluster (if no elastic IP address is available, purchase one), create a user named **hbasetest**, and bind it to the user group **supergroup** and role **System_administrator**.

Username	User Type	Description	Password Policy
hbasetest	Human-Ma...		default

Username:	hbasetest	User Group:	supergroup
User Type:	Human-Machine	Role:	System_administrator
Primary Group:	compcommon	Description:	
Created:	Jan 25, 2024		

2. Download the cluster client, and install it, for example, in the **/opt/client** directory of the active master node.
You can also use the cluster client provided by the Master node. The installation directory is **/opt/Bigdata/client**.
3. Run the following commands to bind an elastic IP address to the active Master node, log in to the active Master node as user **root**, go to the directory where the client is located, and authenticate the user.


```
cd /opt/client
source bigdata_env
kinit hbasetest
```

4. Run the **hbase shell** command to go to the HBase shell page.
Plan the table name, rowkey, column family, and column of the HBase data table based on the imported data. Ensure that the rowkey is pre-split during table creation.
Run the following command to create the **student_info** table:
create 'student_info', {NAME => 'base',COMPRESSION => 'SNAPPY', DATA_BLOCK_ENCODING => 'FAST_DIFF'},SPLITS => ['1','2','3','4','5','6','7','8']
 - **NAME => 'base':** Column family name of the HBase table
 - **COMPRESSION:** Compression mode
 - **DATA_BLOCK_ENCODING:** encoding algorithm
 - **SPLITS:** Region pre-splitting
5. Run the following command to check whether the table is created and exit the HBase shell page:
list

Generating an HFile and Importing It to HBase

1. Create a custom template file, for example, **/opt/configuration_index.xml**. You can obtain the template file example from *Client installation directory/HBase/hbase/conf/index_import.xml.template*.

vi /opt/configuration_index.xml

An example template file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<!--The value of column_num must be consistent with the number of columns in the data file: 5
columns -->
<import column_num="5" id="first">
<columns>
<column type="string" index="1">P_ID</column>
<column type="string" index="2">P_NAME</column>
<column type="string" index="3">P_BIRTH</column>
<column type="string" index="4">P_GENDER</column>
<column type="string" index="5">P_DISTRICT</column>
</columns>
<!--reverse(P_BIRTH): Reverse the birth date to avoid hotspotting. -->
<!--substring(P_NAME,0,1): Filter out the student information based on the last name. -->
<!--substring(P_ID,0,6): Filter out the student information based on the first six digits of a student ID.
-->
<rowkey>
reverse(P_BIRTH)+'_'+substring(P_NAME,0,1)+'_'+substring(P_ID,0,6)
</rowkey>
<qualifiers>
<!--The specified family must correspond to the column family of the table. -->
<normal family="base">
<qualifier column="P_ID">H_ID</qualifier>
<qualifier column="P_NAME">H_NAME</qualifier>
<qualifier column="P_BIRTH">H_BIRTH</qualifier>
<qualifier column="P_GENDER">H_GENDER</qualifier>
<qualifier column="P_DISTRICT">H_DISTRICT</qualifier>
</normal>
</qualifiers>
</import>
</configuration>
```

2. Run the following commands to generate an HFile file:

```
hbase com.huawei.hadoop.hbase.tools.bulkload.ImportData -  
Dimport.separator=',' -Dimport.hfile.output=/tmp/test/hfile /opt/  
configuration_index.xml student_info /tmp/test/info.txt
```

- **-Dimport.separator**: indicates a separator.
- **-Dimport.hfile.output**: indicates the output path of the execution result.
- **/opt/configuration_index.xml**: indicates a custom template file.
- **student_info**: indicates the name of the HBase table to be operated.
- **/tmp/test/info.txt**: indicates the HDFS data directory to which data is to be uploaded in batches.
- **com.huawei.hadoop.hbase.tools.bulkload.IndexImportData**: indicates **IndexImportData** used to create a secondary index during data import. If no secondary index needs to be created, **ImportData** is used.

After the MapReduce job is successfully executed, run the following command to an HFile file in the output path.

```
hdfs dfs -ls /tmp/test/hfile
```

```
Found 2 items  
-rw-r--r-- 3 hbasetest hadoop 0 2021-05-14 11:39 /tmp/test/hfile/_SUCCESS  
drwxr-xr-x - hbasetest hadoop 0 2021-05-14 11:39 /tmp/test/hfile/base
```

3. Run the following command to import the HFile to the HBase table:

```
hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /tmp/  
test/hfile student_info
```

4. Run the following commands to go to the HBase shell page and view the table content:

```
hbase shell
```

```
scan 'student_info', {FORMATTER => 'toString'}
```

```
ROW COLUMN+CELL  
10115102_Yang_202001 column=base:H_BIRTH, timestamp=2021-05-14T15:28:56.755,  
value=20151101  
10115102_Yang_202001 column=base:H_DISTRICT,  
timestamp=2021-05-14T15:28:56.755, value=City3  
10115102_Yang_202001 column=base:H_GENDER,  
timestamp=2021-05-14T15:28:56.755, value=female  
10115102_Yang_202001 column=base:H_ID, timestamp=2021-05-14T15:28:56.755,  
value=20200101247  
10115102_Yang_202001 column=base:H_NAME, timestamp=2021-05-14T15:28:56.755,  
value=Yang xx  
10215102_Li_202001 column=base:H_BIRTH, timestamp=2021-05-14T15:28:56.755,  
value=20151201  
10215102_Li_202001 column=base:H_DISTRICT, timestamp=2021-05-14T15:28:56.755,  
value=City7  
...
```

5. Analyze and process data based on the upper-layer applications of the big data platform after data is imported to the cluster.

2.3 Migrating Data from MySQL to an MRS Hive Partitioned Table

MRS provides enterprise-class on-cloud big data clusters. It contains components such as HDFS, Hive, and Spark, and is tailored to analyze massive amounts of enterprise data.

Hive supports SQL to help users perform extraction, transformation, and loading (ETL) operations on large-scale data sets. Query on large-scale data sets takes a long time. In many scenarios, you can create Hive partitions to reduce the total amount of data to be scanned each time. This significantly improves query performance.

Hive partitions are implemented by using the HDFS subdirectory function. Each subdirectory contains the column names and values of each partition. If there are multiple partitions, there are many HDFS subdirectories. It is not easy to load external data to each partition of the Hive table without using tools. With CDM, you can easily load data of the external data sources (relational databases, object storage services, and file system services) to Hive partition tables.

This practice demonstrates how to use CDM to import MySQL data to the Hive partition table in an MRS cluster.

Scenario

Suppose that there is a **trip_data** table in the MySQL database. The table stores cycling records such as the start time, end time, start sites, end sites, and rider IDs.

For details about the fields in the **trip_data** table, see [Figure 2-4](#).

Figure 2-4 MySQL table fields

Column Name	#	Data Type
TripID	1	int(11)
Duration	2	int(11)
StartDate	3	timestamp
StartStation	4	varchar(64)
StartTerminal	5	int(11)
EndDate	6	timestamp
EndStation	7	varchar(64)
EndTerminal	8	int(11)
Bike	9	int(11)
SubscriberType	10	varchar(32)
ZipCodev	11	varchar(10)

The following describes how to use CDM to import data in the **trip_data** table in the MySQL database to the MRS Hive partition table. The procedure includes five steps:

1. [Creating a Hive Partitioned Table on MRS Hive](#)
2. [Creating a CDM Cluster and Binding an EIP to the Cluster](#)
3. [Creating a MySQL Link](#)
4. [Creating an MRS Hive Link](#)
5. [Creating a Migration Job](#)

Prerequisites

- You have purchased an MRS cluster that contains the Hive service.
- You have obtained the IP address, port, database name, username, and password for connecting to the MySQL database. In addition, the user must have the read and write permissions on the MySQL database.
- You have uploaded the MySQL database driver by referring to [Managing Drivers](#).

Creating a Hive Partitioned Table on MRS Hive

On MRS Hive, run the following SQL statement to create a Hive partition table named **trip_data** with three new fields **y**, **ym**, and **ymd** used as partition fields.

The SQL statement is as follows:

```
create table trip_data(TripID int,Duration int,StartDate timestamp,StartStation varchar(64),StartTerminal int,EndDate timestamp,EndStation varchar(64),EndTerminal int,Bike int,SubscriberType varchar(32),ZipCodev varchar(10))partitioned by (y int,ym int,ymd int);
```

NOTE

The **trip_data** partition table has three partition fields: year, year and month, and year, month, and date of the start time of a ride.

For example, if the start time of a ride is **2018/5/11 9:40**, the record is saved in the **trip_data/2018/201805/20180511** partition.

When the records in the **trip_data** table are summarized, only part of the data needs to be scanned, greatly improving the performance.

Creating a CDM Cluster and Binding an EIP to the Cluster


- Step 1** If CDM is deployed as an independent service, create a CDM cluster by referring to [Creating a CDM Cluster](#). If it is deployed as a component of the DataArts Studio service, create a CDM cluster by referring to [Creating a CDM Cluster](#).

The key configurations are as follows:

- The flavor of the CDM cluster is selected based on the amount of data to be migrated. Generally, `cdm.medium` meets the requirements for most migration scenarios.
- The VPC, subnet, and security group of the CDM cluster must be the same as those of the MRS cluster.

- Step 2** After the CDM cluster is created, on the **Cluster Management** page, click **Bind Elastic IP** in the **Operation** column to bind an EIP to the cluster. The CDM cluster uses the EIP to access MySQL.

Figure 2-5 Cluster list



Clusters you can still create: 1						
Name	Status	Internal Network Address	Public Network Address	Enterprise Project	Operation	
[icon] [dropdown]	Running	192.168.1.5	-	default	Job Management	Bind EIP More

NOTE

If SSL encryption is configured for the access channel of a local data source, CDM cannot connect to the data source using the EIP.

----End

Creating a MySQL Link

Step 1 On the **Cluster Management** page of CDM, click **Job Management** in the **Operation** column of the CDM cluster. On the displayed page, click the **Links** tab and then **Create Link**.

Figure 2-6 Selecting a connector



Step 2 Select **MySQL** and click **Next**. On the page that is displayed, configure MySQL link parameters.

Click **Show Advanced Attributes** and configure optional parameters. Retain the default values of the optional parameters and configure the mandatory parameters according to [Table 2-3](#).

Table 2-3 MySQL link parameters

Parameter	Description	Example Value
Name	Enter a unique link name.	mysqllink
Database Server	IP address or domain name of the MySQL database	192.168.1.110
Port	MySQL database port	3306
Database Name	Name of the MySQL database	sqoop

Parameter	Description	Example Value
Username	User who has the read, write, and delete permissions on the MySQL database	admin
Password	Password of the user	-
Use Agent	Whether to extract data from the data source through an agent	Yes
Agent	Click Select to select the agent created in .	-

Step 3 Click **Save**. The **Links** page is displayed.

 **NOTE**

If an error occurs during the saving, the security settings of the MySQL database are incorrect. In this case, you need to enable the EIP of the CDM cluster to access the MySQL database.

----End

Creating an MRS Hive Link

Step 1 On the **Links** page, click **Create Link** and select **MRS Hive** to create an MRS Hive link.

Step 2 Click **Next** and configure the MRS Hive link parameters. See [Figure 2-7](#).

Figure 2-7 Creating an MRS Hive link

* Name

* Connector

* Hadoop Type

* Manager IP [Select](#)

Authentication Method

* HIVE Version

* Username

* Password

* OBS storage support

* Run Mode

Use Cluster Config

[Show Advanced Attributes](#)

Table 2-4 describes the parameters. You can configure the parameters as required.

Table 2-4 MRS Hive link parameters

Parameter	Description	Example Value
Name	Link name, which should be defined based on the data source type, so it is easier to remember what the link is for	hivelink
Manager IP	Floating IP address of MRS Manager. Click Select next to the Manager IP text box to select an MRS cluster. CDM automatically fills in the authentication information.	127.0.0.1

Parameter	Description	Example Value
Authentication Method	<p>Authentication method used for accessing MRS</p> <ul style="list-style-type: none"> • SIMPLE: Select this for non-security mode. • KERBEROS: Select this for security mode. 	SIMPLE
HIVE Version	Hive version. Set it to the Hive version on the server.	HIVE_3_X
Username	<p>If Authentication Method is set to KERBEROS, you must provide the username and password used for logging in to MRS Manager. If you need to create a snapshot when exporting a directory from HDFS, the user configured here must have the administrator permission on HDFS.</p> <p>To create a data connection for an MRS security cluster, do not use user admin. The admin user is the default management page user and cannot be used as the authentication user of the security cluster. You can create an MRS user and set Username and Password to the username and password of the created MRS user when creating an MRS data connection.</p> <p>NOTE</p> <ul style="list-style-type: none"> • If the CDM cluster version is 2.9.0 or later and the MRS cluster version is 3.1.0 or later, the created user must have the permissions of the Manager_viewer role to create links on CDM. To perform operations on databases, tables, and data of a component, you also need to add the user group permissions of the component to the user. • If the CDM cluster version is earlier than 2.9.0 or the MRS cluster version is earlier than 3.1.0, the created user must have the permissions of Manager_administrator or System_administrator to create links on CDM. • A user with only the Manager_tenant or Manager_auditor permission cannot create connections. 	cdm
Password	Password used for logging in to MRS Manager	-
OBS storage support	The server must support OBS storage. When creating a Hive table, you can store the table in OBS.	No

Parameter	Description	Example Value
Run Mode	<p>This parameter is used only when the Hive version is HIVE_3_X. Possible values are:</p> <ul style="list-style-type: none"> • EMBEDDED: The link instance runs with CDM. This mode delivers better performance. • Standalone: The link instance runs in an independent process. If CDM needs to connect to multiple Hadoop data sources (MRS, Hadoop, or CloudTable) with both Kerberos and Simple authentication modes, select STANDALONE or configure different agents. <p>Note: The STANDALONE mode is used to solve the version conflict problem. If the connector versions of the source and destination ends of the same data link are different, a JAR file conflict occurs. In this case, you need to place the source or destination end in the STANDALONE process to prevent the migration failure caused by the conflict.</p>	EMBEDDED
Use Cluster Config	You can use the cluster configuration to simplify parameter settings for the Hadoop connection.	No
Cluster Config Name	This parameter is valid only when Use Cluster Config is set to Yes . Select a cluster configuration that has been created.	hive_01

Step 3 Click **Save**. The **Links** page is displayed.

----End

Creating a Migration Job

Step 1 On the **Cluster Management** page, locate the row containing your desired cluster, and click **Job Management** in the **Operation** column. On the page that is displayed, click the **Table/File Migration** tab and then **Create Job** to create a data migration job. See [Figure 2-8](#).

Figure 2-8 Creating a job for migrating data from MySQL to Hive

Job Configuration

* Job Name: mysql2dws

Source Job Configuration

* Source Link Name: mysqlink

Use Sql: Yes No

* Schema/Table Space: sqoop

* Table Name: cdm

[Show Advanced Attributes](#)

Destination Job Configuration

* Destination Link Name: dwslink

* Schema/Table Space: public

Auto Table Creation: Auto Creation

* Table Name: date

isCompress: Yes No

Orientation: ROW

Clear data or Clear some data before import: none

[Show Advanced Attributes](#)

NOTE

Set **Clear Data Before Import** to **Yes** so that the data that has been imported to the Hive table is cleared each time before data is imported.

Step 2 After the parameters are configured, click **Next**. The **Map Field** page is displayed, as shown in **Figure 2-9**.

Map the fields of the MySQL table and Hive table. The Hive table has three more fields **y**, **ym**, and **ymd** than the MySQL table, which are the Hive partition fields. Because the fields of the source table cannot be directly mapped to the destination table, you need to configure an expression to extract data from the **StartDate** field in the source table.

Figure 2-9 Hive field mapping

Source Field				Destination Field			
Name	Example Value	Type	Operation	Name	Type	Distributed Columns	Operation
ID		DATETIME		ID	TIMESTAMP(19)	<input type="checkbox"/>	
NAME	2017-06-29 12:00:00	VARCHAR(20)		NAME	VARCHAR(20)	<input type="checkbox"/>	

Step 3 Click to display the **Converter List** dialog box, and then choose **Create Converter > Expression conversion**.

The expressions for the **y**, **ym**, and **ymd** fields are as follows:

DateUtils.format(DateUtils.parseDate(row[2],"yyyy-MM-dd HH:mm:ss.SSS"),"yyyy")

DateUtils.format(DateUtils.parseDate(row[2],"yyyy-MM-dd HH:mm:ss.SSS"),"yyyyMM")

DateUtils.format(DateUtils.parseDate(row[2],"yyyy-MM-dd HH:mm:ss.SSS"),"yyyyMMdd")

Step 4 Click **Next** to set task parameters. Generally, retain the default values of all parameters.

In this step, you can configure the following optional functions:

- **Retry Upon Failure:** If the job fails to be executed, you can determine whether to automatically retry. Retain the default value **Never**.
- **Group:** Select the group to which the job belongs. The default group is **DEFAULT**. On the **Job Management** page, jobs can be displayed, started, or exported by group.
- **Scheduled Execution:** Retain the default value **No**.
- **Concurrent Extractors:** Enter the number of extractors to be concurrently executed. Retain the default value **1**.
- **Write Dirty Data:** Specify this parameter if data that fails to be processed or filtered out during job execution needs to be written to OBS for future viewing. Before writing dirty data, create an OBS link. Retain the default value **No** so that dirty data is not recorded.
- **Delete Job After Completion:** Retain the default value **Do not delete**.

Step 5 Click **Save and Run**. The **Job Management** page is displayed, on which you can view the job execution progress and result.

Step 6 After the job is successfully executed, in the **Operation** column of the job, click **Historical Record** to view the job's historical execution records and read/write statistics.

On the **Historical Record** page, click **Log** to view the job logs.

----End

2.4 Migrating Data from MRS HDFS to OBS

Scenario

CDM supports file-to-file data migration. This section describes how to migrate data from MRS HDFS to OBS file system using CDM.

The process is as follows:

1. [Creating a CDM Cluster and Binding an EIP to the Cluster](#)
2. [Creating an MRS HDFS Link](#)
3. [Creating an OBS Link](#)
4. [Creating a Migration Job](#)

Prerequisites

- You have obtained the domain name, port number, AK, and SK for accessing OBS.
- You have created an MRS cluster that contains the Hadoop service.
- You have the EIP quota and have created an EIP.

Creating a CDM Cluster and Binding an EIP to the Cluster

- Step 1** If CDM is deployed as an independent service, create a CDM cluster by referring to [Creating a CDM Cluster](#). If it is deployed as a component of the DataArts Studio service, create a CDM cluster by referring to [Creating a CDM Cluster](#).

The key configurations are as follows:

- The flavor of the CDM cluster is selected based on the amount of data to be migrated. Generally, **cdm.medium** meets the requirements for most migration scenarios.
- The VPC, subnet, and security group of the CDM cluster must be the same as those of the MRS cluster.

- Step 2** After the CDM cluster is created, on the **Cluster Management** page, click **Bind Elastic IP** in the **Operation** column to bind an EIP to the cluster. The CDM cluster uses the EIP to access MRS HDFS.

NOTE

If SSL encryption is configured for the access channel of a local data source, CDM cannot connect to the data source using the EIP.

----End

Creating an MRS HDFS Link

- Step 1** On the **Cluster Management** page, click **Job Management** in the **Operation** column of the cluster. On the page that is displayed, click the **Links** tab then **Create Link**. On the **Select Connector** page that is displayed, select **MRS HDFS** for **Hadoop**, and click **Next** to set MRS HDFS link parameters.

- **Name:** Enter a custom link name, for example, **mrs_hdfs_link**.
- **Manager IP:** IP address of MRS Manager. Click **Select** next to the **Manager IP** text box to select a created MRS cluster. CDM automatically fills in the authentication information.
- **Username:** If KERBEROS is used for authentication, the username and password for logging in to MRS Manager is required.
If you need to create a snapshot when exporting a directory from HDFS, the user configured here must have the administrator permission on HDFS.
- **Password:** password for logging in to MRS Manager
- **Authentication Method:** authentication method for accessing MRS
- **Run Mode:** Select the running mode of the HDFS link.

----End

Creating an OBS Link

- Step 1** On the **Cluster Management** page, click **Job Management** in the **Operation** column of the cluster. On the page that is displayed, click the **Links** tab then **Create Link**. In the displayed dialog box, select **OBS** for **Connector**, and click **Next** to set OBS link parameters.

- **Name:** Enter a custom link name, for example, **obslink**.

- **OBS Endpoint and Port:** Enter the actual OBS address information.
- **OBS Bucket Type:** Use the default value.
- **AK and SK:** Enter the AK and SK used for logging in to OBS.

Step 2 Click **Save**. The **Links** page is displayed.

----End

Creating a Migration Job

Step 1 On the **Cluster Management** page, click **Job Management** in the **Operation** column of the cluster. On the page that is displayed, click the **Table/File Migration** tab then **Create Job** to create a job for exporting data from MRS HDFS to OBS.

Figure 2-10 Creating a job for migrating data from MRS HDFS to OBS

The screenshot shows the 'Job Configuration' page with the following details:

- Job Name:** hdfs2obs_004more
- Source Job Configuration:**
 - Source Link Name: hdfs_link
 - Source Directory/File: /interface/hdfsfrom/more
 - File Format: CSV
- Destination Job Configuration:**
 - Destination Link Name: obs_link
 - Bucket Name: cdm-autotest
 - Write Directory: /interface/obsto
 - File Format: CSV
 - Duplicate File Processing Method: Replace

- **Job Name:** Enter a unique name.
- **Source Job Configuration**
 - **Source Link Name:** Select the **hdfs_link** created in [Creating an MRS HDFS Link](#).
 - **Source Directory/File:** Enter the directory or file path of the data to be migrated.
 - **File Format:** Select the file format used for data transmission. Select **Binary**. If files are transferred without being parsed, the file format does not have to be **Binary**. This applies to file copy.
 - Retain the default values of other optional parameters.
- **Destination Job Configuration**
 - **Destination Link Name:** Select the **obs_link** created in [Creating an OBS Link](#).
 - **Bucket Name:** Select the bucket from which the data will be migrated.
 - **Write Directory:** Enter the directory to which data is to be written on the OBS server.
 - **File Format:** Select **Binary**.

- Retain the default values of the optional parameters in **Show Advanced Attributes**.

Step 2 Click **Next**. The **Map Field** page is displayed. CDM automatically matches the source and destination fields.

- If the field mapping is incorrect, you can drag the fields to adjust the mapping.
- CDM expressions have built-in ability to convert fields of common strings, dates, and numbers.

Step 3 Click **Next** to set task parameters. Typically, retain the default values for all parameters.

In this step, you can configure the following optional functions:

- **Retry Upon Failure:** If the job fails to be executed, you can determine whether to automatically retry. Retain the default value **Never**.
- **Group:** Select the group to which the job belongs. The default group is **DEFAULT**. On the **Job Management** page, jobs can be displayed, started, or exported by group.
- **Scheduled Execution:** The default value is **No**.
- **Concurrent Extractors:** Enter the number of extractors to be concurrently executed. CDM supports concurrent extraction of multiple files. Increasing the value of this parameter can improve migration efficiency.
- **Write Dirty Data:** Select **No**. The file-to-file migration is binary, and no dirty data will be generated.
- **Delete Job After Completion:** Retain the default value **Do not delete**. You can also set this parameter to **Delete** to prevent an accumulation of too many migration jobs.

Step 4 Click **Save and Run**. The **Job Management** page is displayed, on which you can view the job execution progress and result.

Step 5 After the job is successfully executed, in the **Operation** column of the job, click **Historical Record** to view the job's historical execution records and read/write statistics.

On the **Historical Record** page, click **Log** to view the job logs.

----End

3 Data Backup and Restoration

3.1 HDFS Data

Establishing a Data Transmission Channel

- If the source cluster and destination cluster are deployed in different VPCs in the same region, create a network connection between the two VPCs to establish a data transmission channel at the network layer. For details, see **Virtual Private Cloud > User Guide > VPC Peering Connection**.
- If the source cluster and destination cluster are deployed in the same VPC but belong to different security groups, add security group rules to each security group on the VPC management console. In the security rules, **Protocol** is set to **ANY**, **Transfer Direction** is set to **Inbound**, and **Source** is set to **Security Group** (the security group of the peer cluster).
 - To add an inbound rule to the security group of the source cluster, select the security group of the destination cluster in **Source**.
 - To add an inbound rule to the security group of the destination cluster, select the security group of the source cluster in **Source**.
- If the source and destination clusters are deployed in the same security group of the same VPC and Kerberos authentication is enabled for both clusters, configure mutual trust between the two clusters.

Backing Up HDFS Data

Based on the regions of and network connectivity between the source cluster and destination cluster, data backup scenarios are classified as follows:

- Same Region
If the source cluster and destination cluster are in the same region, set up a network transmission channel. Use the DistCp tool to run the following command to copy the HDFS, HBase, Hive data files and Hive metadata backup files from the source cluster to the destination cluster.

```
$HADOOP_HOME/bin/hadoop distcp <src> <dist> -p
```

The following provides description about the parameters in the preceding command.

- ***\$HADOOP_HOME***: installation directory of the Hadoop client in the destination cluster
- **<src>**: HDFS directory of the source cluster
- **<dist>**: HDFS directory of the destination cluster
- Different Regions
If the source cluster and destination cluster are in different regions, use the DistCp tool to copy the source cluster data to OBS, and use the OBS cross-region replication function to copy the data to OBS in the region where the destination cluster is deployed. If DistCp is used, permission, owner, and group information cannot be set for files on OBS. In this case, you need to export and copy the HDFS metadata while exporting data to prevent the loss of HDFS file property information.
- Migrating Data from an Offline Cluster to a Cloud
You can use the following way to migrate data from an offline cluster to the cloud.
 - Direct Connect
Create a Direct Connect between the source cluster and destination cluster, enable the network between the offline cluster egress gateway and the online VPC, and execute the DistCp to copy the data by referring to the method provided in [Same Region](#).

Backing Up HDFS Metadata

HDFS metadata information to be exported includes file and folder permissions and owner/group information. You can run the following command on the HDFS client to export the metadata:

```
$HADOOP_HOME/bin/hdfs dfs -ls -R <migrating_path> > /tmp/hdfs_meta.txt
```

The following provides description about the parameters in the preceding command.

- ***\$HADOOP_HOME***: installation directory of the Hadoop client in the source cluster
- **<migrating_path>**: HDFS data directory to be migrated
- **/tmp/hdfs_meta.txt**: local path for storing the exported metadata

NOTE

If the source cluster can communicate with the destination cluster and you run the **hadoop distcp** command as a super administrator to copy data, you can add the **-p** parameter to enable DistCp to restore the metadata of the corresponding file in the destination cluster while copying data. In this case, skip this step.

HDFS File Property Restoration

Based on the exported permission information, run the HDFS commands in the background of the destination cluster to restore the file permission and owner and group information.

```
$HADOOP_HOME/bin/hdfs dfs -chmod <MODE> <path>  
$HADOOP_HOME/bin/hdfs dfs -chown <OWNER> <path>
```

3.2 Hive Metadata

Backing Up Hive Metadata

Hive table data is stored in HDFS. Table data and the metadata of the table data is centrally migrated in directories by HDFS in a unified manner. Metadata of Hive tables can be stored in different types of relational databases (such as MySQL, PostgreSQL, and Oracle) based on cluster configurations. The exported metadata of the Hive tables in this document is the Hive table description stored in the relational database.

The mainstream big data release editions in the industry support Sqoop installation. For on-premises big data clusters of the community version, you can download the Sqoop of the community version for installation. Use Sqoop to decouple the strong dependency between the metadata to be exported and the relational database and export Hive metadata to HDFS and migrate it together with the table data for restoration. The procedure is as follows:

- Step 1** Download the Sqoop tool from the source cluster and install it. For details, see <http://sqoop.apache.org/>.
- Step 2** Download the JDBC driver of the relational database to the `$Sqoop_Home/lib` directory.
- Step 3** Run the following command to export all Hive metadata tables: All exported data is stored in the `/user/<user_name>/<table_name>` directory on HDFS.

```
$Sqoop_Home/bin/sqoop import --connect jdbc:<driver_type>://<ip>:<port>/<database> --table <table_name> --username <user> -password <passwd> -m 1
```

The following provides description about the parameters in the preceding command.

- `$Sqoop_Home`: Sqoop installation directory
- `<driver_type>`: Database type
- `<ip>`: IP address of the database in the source cluster
- `<port>`: Port number of the database in the source cluster
- `<table_name>`: Name of the table to be exported
- `<user>`: Username
- `<passwd>`: User password

NOTE

Commands carrying authentication passwords pose security risks. Disable historical command recording before running such commands to prevent information leakage.

----End

Hive Metadata Restoration

Install Sqoop and run the Sqoop command in the destination cluster to import the exported Hive metadata to DBService in the MRS cluster.

```
$Sqoop_Home/bin/sqoop export --connect jdbc:postgresql://<ip>:20051/hivemeta --table <table_name> --username hive -password <passwd> --export-dir <export_from>
```

The following provides description about the parameters in the preceding command.

- ***\$Sqoop_Home***: Sqoop installation directory in the destination cluster
- ***<ip>***: IP address of the database in the destination cluster
- ***<table_name>***: Name of the table to be restored
- ***<passwd>***: Password of user **hive**
- ***<export_from>***: HDFS address of the metadata in the destination cluster

 **NOTE**

Commands carrying authentication passwords pose security risks. Disable historical command recording before running such commands to prevent information leakage.

3.3 Hive Data

Hive data is not backed up independently. For details, see [HDFS Data](#).

3.4 HBase Data

Currently, HBase data can be backed up in the following modes:

- Snapshots
- Replication
- Export
- CopyTable
- HTable API
- Offline backup of HDFS data

[Table 3-1](#) compares the impact of operations from six perspectives.

Table 3-1 Data backup mode comparison on HBase

Backup Mode	Performance Impact	Data Footprint	Downtime	Incremental Backup	Ease of Implementation	Mean Time to Repair (MTTR)
Snapshots	Minimal	Tiny	Brief (Only for Restore)	No	Easy	Seconds
Replication	Minimal	Large	None	Intrinsic	Medium	Seconds
Export	High	Large	None	Yes	Easy	High
CopyTable	High	Large	None	Yes	Easy	High

Backup Mode	Performance Impact	Data Footprint	Downtime	Incremental Backup	Ease of Implementation	Mean Time to Repair (MTTR)
HTable API	Medium	Large	None	Yes	Difficult	Up to you
Offline backup of HDFS data	-	Large	Long	No	Medium	High

Snapshots

You can perform the snapshot operation on a table to generate a snapshot for the table. The snapshot can be used to back up the original table, roll back the original table when the original table is faulty, as well as back up data cross clusters. After a snapshot is executed, the **.hbase-snapshot** directory is generated in the HBase root directory (**/hbase** by default) on HBase. The directory contains details about each snapshot. When the **ExportSnapshot** command is executed to export the snapshot, an MR task is submitted locally to copy the snapshot information and table's **HFile** to **/hbase/.hbase-snapshot** and **/hbase/archive** of the standby cluster respectively. For details, see <http://hbase.apache.org/2.2/book.html#ops.snapshots>.

- This backup mode has the following advantages:
The single table backup efficiency is high. Online data can be backed up locally or remotely without interrupting services of the active and standby clusters. The number of maps and traffic threshold can be flexibly configured. A MapReduce executor node does not need to be deployed in the active and standby clusters. Therefore, no resource is consumed.
- This backup mode has the following disadvantages and limitations:
Only a single table can be backed up. The name of the table to be backed up has been specified in the snapshot and cannot be changed. Incremental backup cannot be performed. Resources are consumed when an MR task runs.

Perform the following operations on the active cluster:

Step 1 Create a snapshot for a table. For example, create snapshot **member_snapshot** for the **member** table.

```
snapshot 'member','member_snapshot'
```

Step 2 Copy the snapshot to the standby cluster.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot member_snapshot -copy-to hdfs://IP address of the active NameNode of the HDFS service in the standby cluster:Port number/hbase -mappers 3
```

- The data directory of the standby cluster must be the HBase root directory (**/hbase**).

- **mappers** indicates the number of maps to be submitted for an MR task.

----End

Perform the following operations on the standby cluster:

Run the **restore** command to automatically create a table in the standby cluster and establish a link between HFile in **archive** and the table.

restore_snapshot 'member_snapshot'

NOTE

If only table data needs to be backed up, Snapshots is highly recommended. Use SnapshotExport to submit an MR task locally and copies Snapshot and HFile to the standby cluster. Then, data can be directly loaded to the standby cluster, more efficient than using other methods.

Replication

In Replication backup mode, a disaster recovery relationship is established between the active and standby clusters on HBase. When data is written to the active cluster, the active cluster pushes data to the standby cluster through WAL to implement real-time data synchronization between the active and standby clusters. For details, see http://hbase.apache.org/2.2/book.html#_cluster_replication.

- This backup mode has the following advantages:
 - Replication is different from other data backup modes. After the active/standby relationship between clusters is established, data can be synchronized in real time without manual operations.
 - The backup operation consumes few cluster resources and has little impact on cluster performance.
 - Data synchronization reliability is high. If the standby cluster is stopped for a while and then recovered, data generated during this period on the active cluster can be still synchronized to the standby cluster.
- This backup mode has the following disadvantages and limitations:
 - If WAL is not set for data written by clients, data cannot be backed up to the standby cluster.
 - The background synchronizes data in asynchronous mode, because only few resources can be occupied. Therefore, data is not synchronized in real time.
 - If the data exists in the active cluster before you use the replication mode to perform synchronization, you need to use other methods to import these data to the standby cluster.
 - If data is written to the active cluster in **bulkload** mode, it cannot be synchronized. (HBase on MRS enhances replication. Therefore, data written in the **bulkload** mode can be synchronized by replication.)

For details about how to use and configure HBase backup, see "MRS Component Operation Guide" > "Using HBase" > "Configuring HBase Replication" in *MapReduce Service User Guide* and "MRS Component Operation Guide" > "Using HBase" > "Using the ReplicationSyncUp Tool" in *MapReduce Service User Guide*.

Export/Import

Export/Import starts a MapReduce task to scan the data table and writes SequenceFile to the remote HDFS. Then, Import reads SequenceFile and puts it on HBase.

- This backup mode has the following advantages:
Online copy does not interrupt services. Because data is written to new tables in **scan- > put** mode, Export/Import is more flexible than CopyTable. Data to be obtained and used flexibly, and written incrementally.
- This backup mode has the following disadvantages and limitations:
Export writes SequenceFiles to the remote HDFS through a MapReduce task, and then Import reads SequenceFiles and puts them on HBase. Therefore, an MR task needs to be executed twice, thus being inefficient.

Perform the following operations on the active cluster:

Run the **Export** command to export the table.

hbase org.apache.hadoop.hbase.mapreduce.Export <tablename> <outputdir>

Example: **hbase org.apache.hadoop.hbase.mapreduce.Export member hdfs://IP address of the active NameNode of the HDFS service in the standby cluster:Port number/user/table/member**

In the command, **member** indicates the name of the table to be exported.

Perform the following operations on the standby cluster:

- Step 1** After operations are executed on the active cluster, you can view the generated directory data on the standby cluster, as shown in [Figure 3-1](#).

Figure 3-1 Directory data

```
Cvi:~ # hdfs dfs -ls -R /user/table/member
-rw-r--r--  3 root hadoop          0 2018-06-28 14:18 /user/table/member/_SUCCESS
-rw-r--r--  3 root hadoop    2937 2018-06-28 14:18 /user/table/member/part-m-00000
```

- Step 2** Run the **create** command to create a table in the standby cluster with the same structure as that of the active cluster, for example, **member_import**.
- Step 3** Run the **Import** command to generate the HFile data on HDFS.

hbase org.apache.hadoop.hbase.mapreduce.Import <tablename> <inputdir>

Example: **hbase org.apache.hadoop.hbase.mapreduce.Import member_import /user/table/member -Dimport.bulk.output=/tmp/member**

- **member_import** indicates a table in the standby cluster with the same table structure as that of the active cluster.
- **Dimport.bulk.output** indicates the output directory of the HFile data.
- **/user/table/member** indicates the directory for storing data exported from the active cluster.

- Step 4** Perform the **Load** operation to write the HFile data to HBase.

hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /tmp/member member

- **/tmp/member** indicates the output directory of the HFile data in [Step 3](#).
- **member** indicates the name of the table to which data is to be imported in the standby cluster.

----End

CopyTable

The function of CopyTable is similar to that of Export. Like Export, CopyTable uses HBase API to create a MapReduce task to read data from the source table. However, the difference is that the output of CopyTable is an HBase table that can be stored in a local or remote cluster. For details, see <http://hbase.apache.org/2.2/book.html#copy.table>

- This backup mode has the following advantages:
The operation is simple. Online copy does not interrupt services. You can specify the **startrow**, **endrow**, and **timestamp** parameters of the backup data.
- This backup mode has the following disadvantages and limitations:
Only a single table can be operated. The efficiency is low when a large amount of data is remotely copied. The MapReduce task consumes local resources. The number of maps of the MapReduce task is determined by the number of regions in the table.

Perform the following operations on the standby cluster:

Run the **create** command to create a table in the standby cluster with the same structure as that of the active cluster, for example, **member_copy**.

Perform the following operations on the active cluster:

Run the following CopyTable command to copy the table:

```
hbase org.apache.hadoop.hbase.mapreduce.CopyTable [--starttime=xxxxxx]
[--endtime=xxxxxx] --new.name=member_copy --
peer.adr=server1,server2,server3:2181:/hbase [--
families=myOldCf:myNewCf,cf2,cf3] TestTable
```

- **starttime/endtime** indicates the timestamp of the data to be copied.
- **new.name** indicates the name of the destination table in the standby cluster. The default name of the new table is the same as that of the original table.
- **peer.adr** indicates the information about the ZooKeeper node in the standby cluster. The format is **quorum:port:/hbase**.
- **families** indicates the family column of the table to be copied.

NOTE

If data is copied to a remote cluster, a MapReduce task is submitted on the host cluster to import the data. After the full or partial data in the original table is read, it is written to the remote cluster in **put** mode. Therefore, if the table contains a large amount of data (remote copy does not support **bulkload**), the efficiency is unsatisfactory.

HTable API

HTable API imports and exports data of the original HBase table in the code. You can use the public API to write customized client applications to directly query

tables, or design other methods based on the batch processing advantages of MapReduce tasks. This mode requires in-depth understanding of Hadoop development and the impact on the production cluster.

Offline backup of HDFS data

Offline backup of HDFS data means stopping the HBase service and allowing users to manually copy the HDFS data.

- This backup mode has the following advantages:
 - All data (including metadata) in the active cluster can be copied to the standby cluster at a time.
 - Data is directly copied by DistCp. Therefore, the data backup efficiency is relatively high.
 - You can copy data based on the site requirements. You can copy data of only one table or copy one HFile in a region.
- This backup mode has the following disadvantages and limitations:
 - This operation will overwrite the HDFS data directory in the standby cluster.
 - If the HBase versions of the active and standby clusters are different, an error may occur when the HDFS directory is directly copied. For example, if the system table **index** is added to the MRS **hbase1.3** and overwritten by the HDFS directory of the earlier version, the table cannot be found. Therefore, exercise caution when using this mode.
 - This operation has certain requirements on HBase capabilities. If an exception occurs, restore HBase based on the site requirements.

Perform the following operations on the active cluster:

Step 1 Run the following command to save the data in the current cluster to HDFS permanently:

```
flush 'tableName'
```

Step 2 Stop the HBase service.

Step 3 Run the following commands to copy the HDFS data of the current cluster to the standby cluster:

```
hadoop distcp -i /hbase/data hdfs://IP address of the active NameNode of the HDFS service in the standby cluster:Port number/hbase
```

```
hadoop distcp -update -append -delete /hbase/ hdfs://IP address of the active NameNode of the HDFS service in the standby cluster:Port number/hbase/
```

The second command is used to incrementally copy files except the data directory. For example, data in **archive** may be referenced by the data directory.

----End

Perform the following operations on the standby cluster:

Step 1 Restart the HBase service for the data migration to take effect. During the restart, HBase loads the data in the current HDFS and regenerates metadata.

Step 2 After the restart is complete, run the following command on the Master node client to load the HBase table data:

```
$HBase_Home/bin/hbase hbck -fixMeta -fixAssignments
```

Step 3 After the command is executed, run the following command repeatedly to check the health status of the HBase cluster until the health status is normal:

```
hbase hbck
```

NOTE

If the HBase coprocessor is used and custom JAR files are stored in the **regionserver/hmaster** of the active cluster, you need to copy the custom JAR files before restarting the HBase service on the standby cluster.

----End

3.5 Kafka Data

MirrorMaker is a powerful tool for Kafka data synchronization. It is used when data needs to be synchronized between two Kafka clusters or when data in the original Kafka cluster needs to be migrated to a new Kafka cluster. MirrorMaker is a built-in tool in Kafka. It actually integrates the functions of Kafka Consumer and Producer. MirrorMaker can read data from one Kafka cluster and write the data to another Kafka cluster to implement data synchronization between Kafka clusters.

This section describes how to use the MirrorMaker tool provided by MRS to synchronize and migrate Kafka cluster data. Before migrating Kafka data, ensure that the two clusters can communicate with each other by following the instructions provided in [Establishing a Data Transmission Channel](#).

Procedure

Versions earlier than MRS 3.x:

- Step 1** Enable the Kerberos authentication for clusters.
- Step 2** If you plan to use the MirrorMaker tool in a source cluster, go to the details page of a destination cluster and choose **Components**. If you plan to use the MirrorMaker tool in a destination cluster, go to the details page of a source cluster and choose **Components**.
- Step 3** Choose **Kafka > Service Configuration**, and change **Basic** to **All** in the parameter type drop-down box.
- Step 4** Click **Broker > Customization** and add the following rules on the displayed page:
- ```
sasl.kerberos.principal.to.local.rules = RULE:[1:$1@$0]
(*@XXXYYYZZZ.COM)s/@.*//,RULE:[2:$1@$0](.*@
XXXYYYZZZ.COM)s/@.*//,DEFAULT
```
- In the preceding rule, **XXXYYYZZZ.COM** indicates the domain name of the cluster (source cluster) where data resides. The domain name must be spelled in uppercase letters.
- Step 5** Click **Save Configuration** and select **Restart the affected services or instances**. Click **Yes** to restart the Kafka service.

 NOTE

For a security cluster with the Kerberos authentication enabled, perform [Step 1](#) to [Step 5](#).  
For a normal cluster with the Kerberos authentication disabled, skip [Step 1](#) to [Step 5](#) and go to [Step 6](#).

**Step 6** In the cluster that uses the MirrorMaker tool, go to the cluster details page and choose **Components**.

**Step 7** Choose **Kafka > Service Configuration**, change **Basic** to **All** in the parameter type drop-down box, and change **All Roles** to **MirrorMaker**.

Parameter description:

- The **bootstrap.servers** parameter in the **source** and **dest** tags indicates the **broker** node list and port information of the source and destination Kafka clusters respectively.
- Set parameter **security.protocol** in the **source** and **dest** tags based on the actual configurations of the source and destination Kafka clusters.
- If the source Kafka cluster or destination Kafka cluster is a security cluster, you need to set **kerberos.domain.name** and **sasl.kerberos.service.name** in the **source** and **dest** tags. If the local host is used, you do not need to set **kerberos.domain.name**. If the local host is not used, set **kerberos.domain.name** and **sasl.kerberos.service.name** based on the site requirements. The default value of **sasl.kerberos.service.name** is **kafka**.
- Set **whitelist** in the **mirror** tag, that is, the name of the topic to be synchronized.

**Step 8** Click **Save Configuration** and select **Restart the affected services or instances**. Click **Yes** to restart the MirrorMaker instance.

After MirrorMaker is restarted, the data migration task is started.

----End

# 4 System Interconnection

## 4.1 Using DBeaver to Access Phoenix

Use DBeaver 6.3.5 as an example to describe how to access MRS 3.1.0 clusters with Kerberos authentication disabled.

### Prerequisites

- DBeaver 6.3.5 has been installed. You can download the DBeaver installation package by clicking [https://dbeaver.io/files/6.3.5/dbeaver-ce-6.3.5-x86\\_64-setup.exe](https://dbeaver.io/files/6.3.5/dbeaver-ce-6.3.5-x86_64-setup.exe).
- An MRS 3.1.0 cluster, with HBase installed and Kerberos authentication disabled, has been created.
- The HBase client has been installed.
- JDK 1.8.0\_x has been installed.

### Procedure

- Step 1** Add the bin directory of JDK 1.8.0\_x, for example, `C:\Program Files\Java\jdk1.8.0_121\bin`, to the `dbeaver.ini` file in the DBeaver installation directory.

**Figure 4-1** Adding the bin directory of JDK

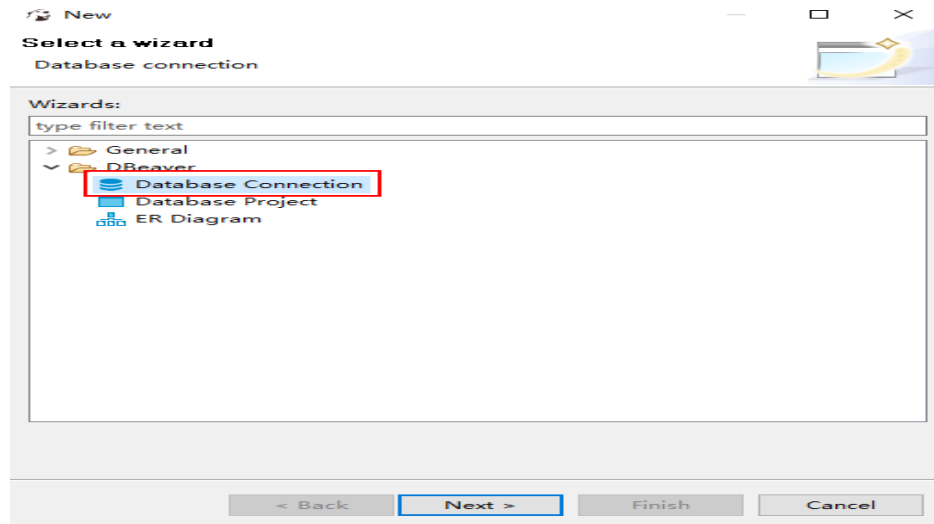
```
-vm
C:\Program Files\Java\jdk1.8.0_121\bin
```

- Step 2** Download the Phoenix software package from <https://archive.apache.org/dist/phoenix/apache-phoenix-5.0.0-HBase-2.0/bin/apache-phoenix-5.0.0-HBase-2.0-bin.tar.gz> and decompress it to obtain `phoenix-5.0.0-HBase-2.0-client.jar`.
- Step 3** Download the `hbase-site.xml` file from `Client installation directory/HBase/hbase/conf` on the node where the client is installed. Use the compression software to open the `phoenix-5.0.0-HBase-2.0-client.jar` file obtained in **Step 2** and drag `hbase-site.xml` to the JAR file.



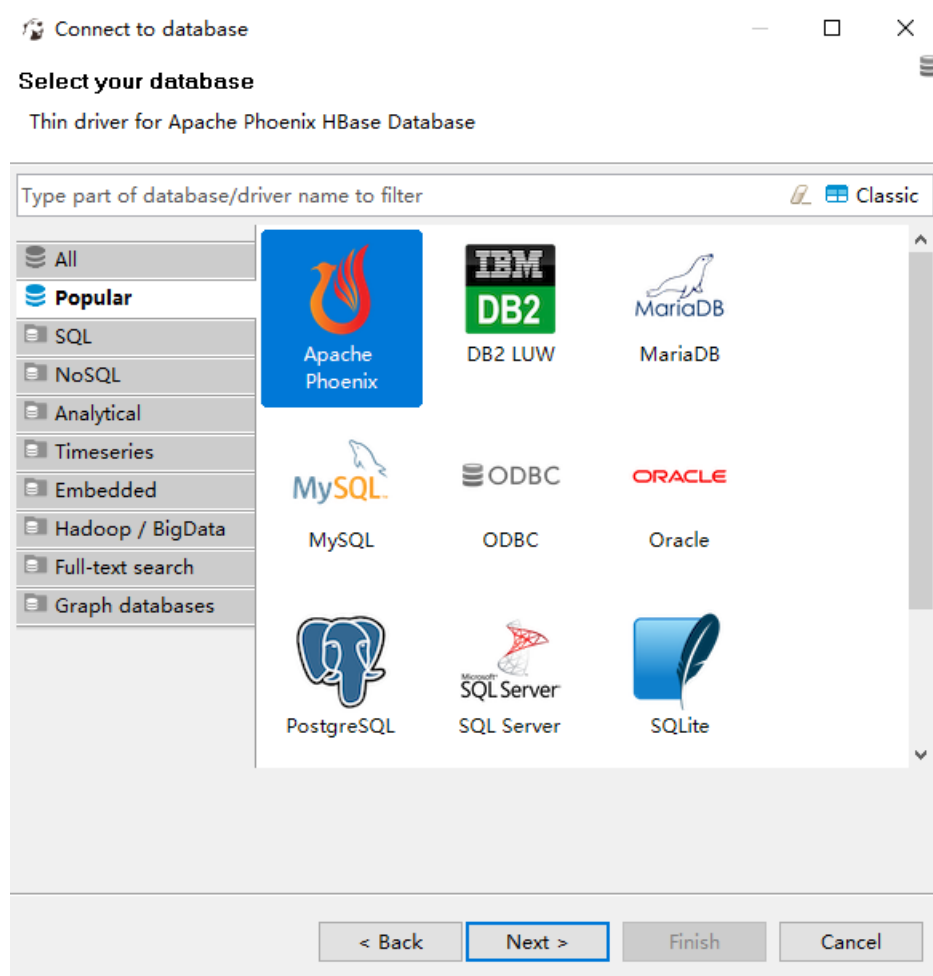
**Step 4** Open DBeaver. On the navigation pane, choose **File > New > DBeaver > Database Connection**.

**Figure 4-2** Creating a database connection



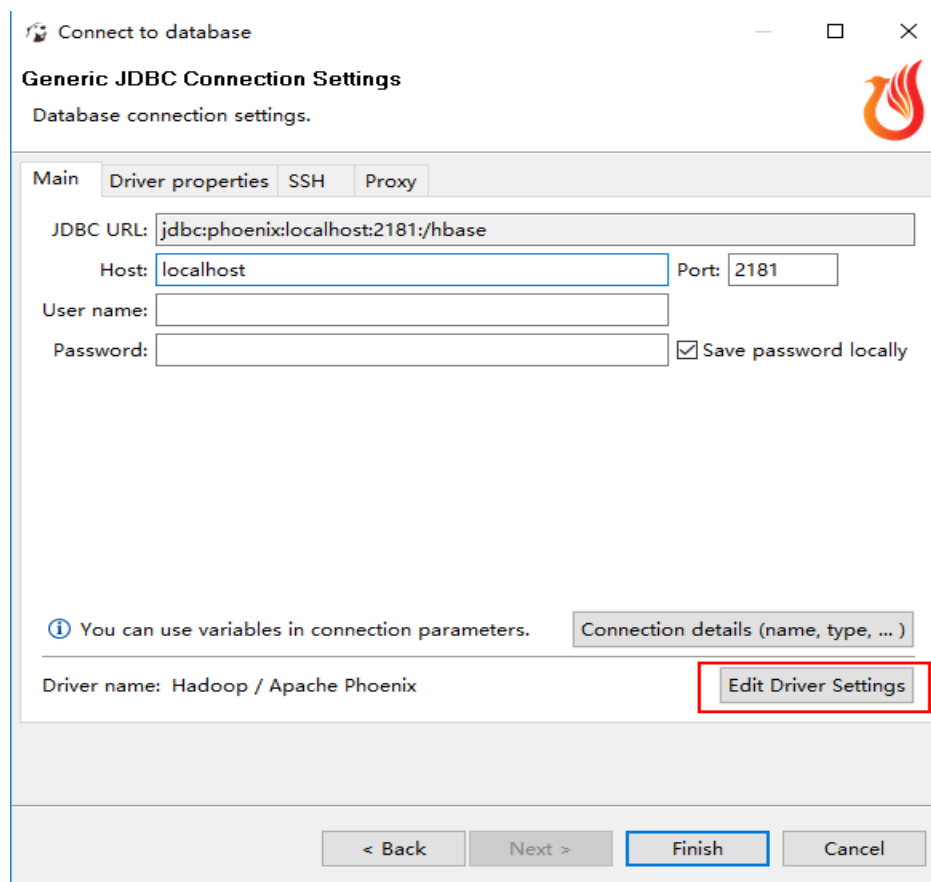
**Step 5** Click **Next**. In the **Select your database** dialog box, select **Apache Phoenix** and click **Next**.

**Figure 4-3** Selecting a database



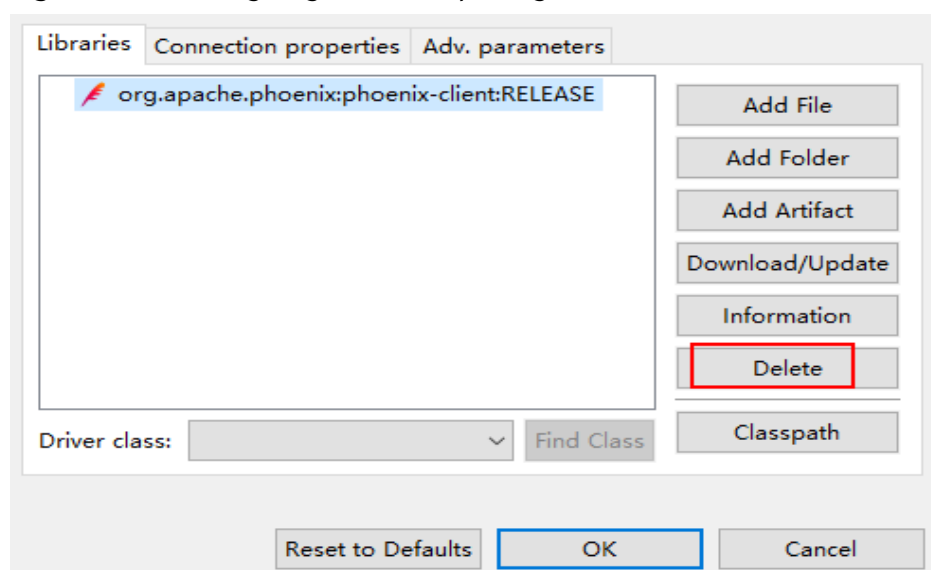
**Step 6** Click **Edit Driver Settings**.

Figure 4-4 Edit Driver Settings

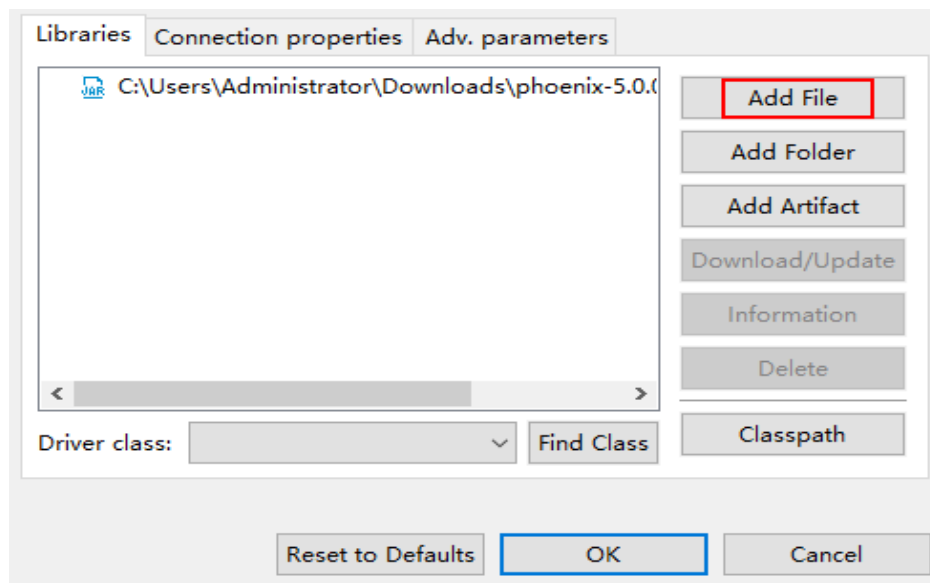


**Step 7** Click **Add File** and select the prepared **phoenix-5.0.0-HBase-2.0-client.jar** file. If there are multiple driver packages, delete them and retain only **added phoenix-5.0.0-HBase-2.0-client.jar**.

Figure 4-5 Deleting original driver packages

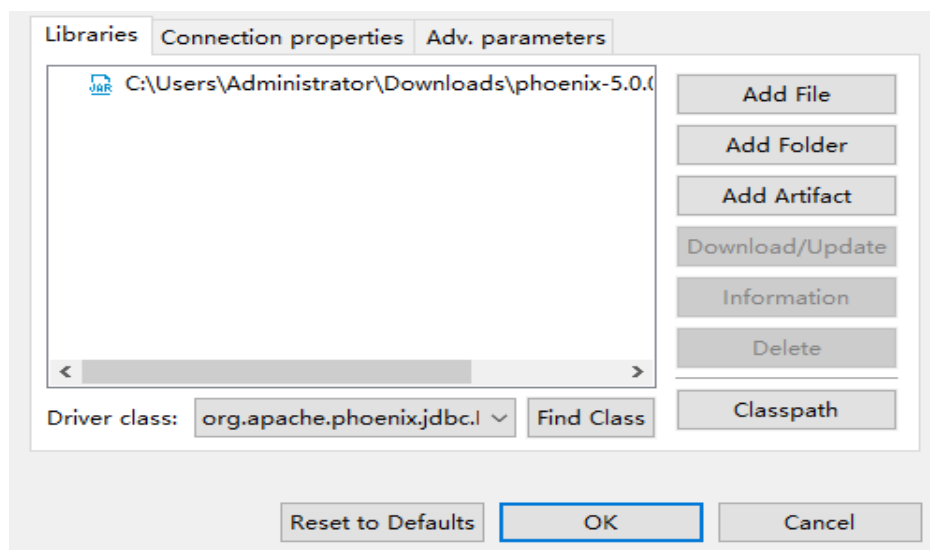


**Figure 4-6** Adding the Phoenix JAR file



**Step 8** Click **Find Class** and select **org.apache.phoenix.jdbc.PhoenixDriver** for **Driver class**.

**Figure 4-7** Loading a driver class



**Step 9** Add the **ZooKeeper Base Path**.

1. Log in to FusionInsight Manager and choose **Cluster > Services > HBase**. On the **Dashboard** tab page that is displayed, click the link next to **HMaster WebUI** to access the HBase web UI. Search for **ZooKeeper Base Path** and obtain its value. As shown in the following figure, the value of **ZooKeeper Base Path** is **/hbase**.

**Figure 4-8** Viewing the value of ZooKeeper Base Path

Software Attributes

| Attribute Name            | Value                                                                  |
|---------------------------|------------------------------------------------------------------------|
| JVM Version               |                                                                        |
| HBase Version             | ?, revision=9c59dbc63eb2daf08b29c51f4bce7c77f642ed12                   |
| HBase Compiled            | Wed Apr 28 18:49:13 CST 2021, root                                     |
| HBase Source Checksum     | 6cfcc863c31df1d8127824d2b08d604d                                       |
| Hadoop Version            | ?, revision=3f6d58324da792aaa3a5592c59561de6387cbe93                   |
| Hadoop Compiled           | 2021-04-28T10:26Z, root                                                |
| Hadoop Source Checksum    | 15ad5f9e94eaf31a9cb0fbbff55bd79                                        |
| ZooKeeper Client Version  | ?, revision=12-c9b3def3b445dca9f3ad21427ec3846b81a92453                |
| ZooKeeper Client Compiled | 04/28/2021 10:20 GMT                                                   |
| ZooKeeper Quorum          | node-master1jfm:2181<br>node-master2uiqz:2181<br>node-master3xcpw:2181 |
| ZooKeeper Base Path       | /hbase                                                                 |

2. Add a colon (:) and the **ZooKeeper Base Path** value, that is, **:/hbase** to the end of the original URL for **URL Template** and click **OK**.

**Figure 4-9** Configuring URL Template

Settings

Driver Name\*:  Driver Type:

Class Name:

URL Template:

Default Port:   Embedd  No authenticati  Allow Empty Passwo

Description

Category:  ID:

Description:

Website: <http://phoenix.apache.org/>

- Step 10** Configure EIPs. If the network between local Windows hosts and the cluster is disconnected, configure an EIP for each HBase node and ZooKeeper node, and add the mapping between the EIPs of all nodes and the host domain names to the **hosts** file on the local Windows hosts. An example is as follows:

```
100.10.10.10 node-master3xCPw node-master3xCPw.
100.10.10.11 node-group-1ZqBd0001 node-group-1ZqBd0001.
100.10.10.12 node-master2uIQz node-master2uIQz.
100.10.10.13 node-group-1ZqBd0002 node-group-1ZqBd0002.
```

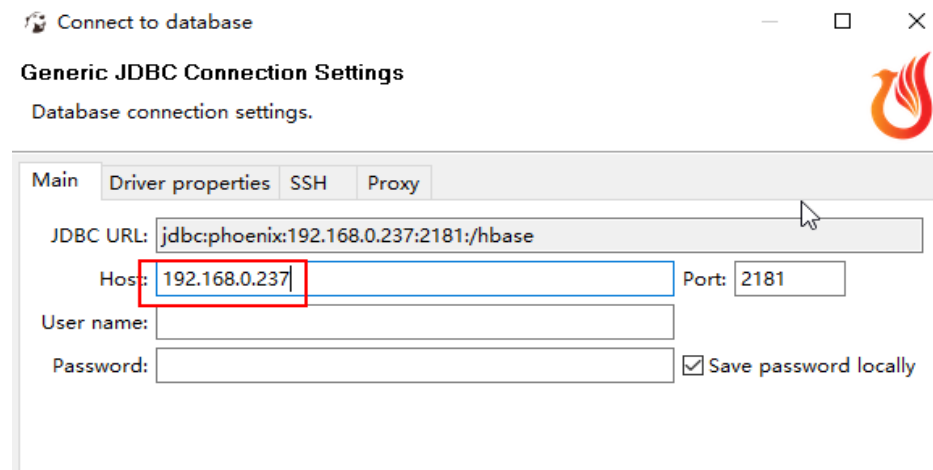
**NOTE**

If Windows ECSs are used and they can communicate with the cluster, you do not need to configure EIPs.

**Step 11** Log in to FusionInsight Manager, choose **Cluster > Services > ZooKeeper**, and click the **Instance** tab.

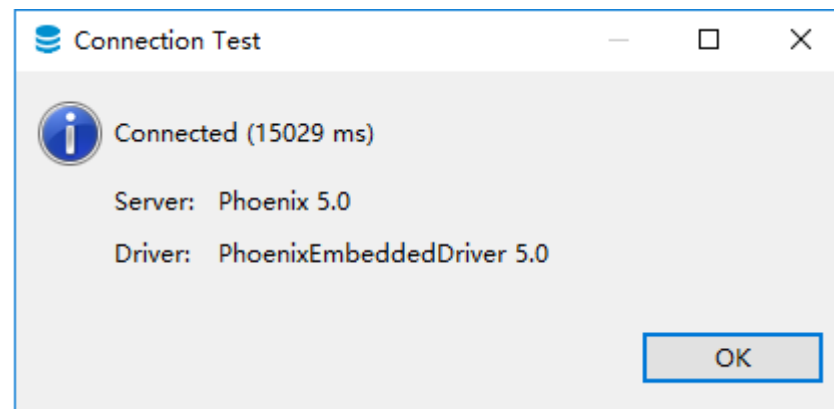
Select a node and enter the EIP of the node in **Host**. (If a Windows ECS is selected and it can communicate with the cluster properly, enter the service IP address of the ECS.)

**Figure 4-10** Configuring Host



**Step 12** Click **Test Connection**. If the information shown in **Figure 4-11** is displayed, the interconnection is successful. Click **OK**.

**Figure 4-11** Connection Test dialog box



**Step 13** Log in to the node where the HBase client is installed as the client installation user and run the following commands to create the *MY\_NS* namespace:

```
cd Client installation directory
source bigdata_env
hbase shell
create_namespace "MY_NS"
```

**Step 14** Open DBeaver and choose **SQL Editor > New SQL Editor** to run related SQL statements.

1. Enter the following commands in the editor and choose **SQL Editor > Execute SQL Statement** to create the *TEST* table in the **DEFAULT** namespace:

```
CREATE TABLE IF NOT EXISTS TEST (id VARCHAR PRIMARY KEY, name VARCHAR);
```

```
UPSERT INTO TEST(id,name) VALUES ('1','jamee');
```

2. Enter the following commands in the editor and choose **SQL Editor > Execute** to create the *TEST* table in the *MY\_NS* namespace and inset data to the namespace:


```
CREATE TABLE IF NOT EXISTS MY_NS.TEST (id integer not null primary key, name varchar);
```

```
UPSERT INTO MY_NS.TEST VALUES(1,'John');
```

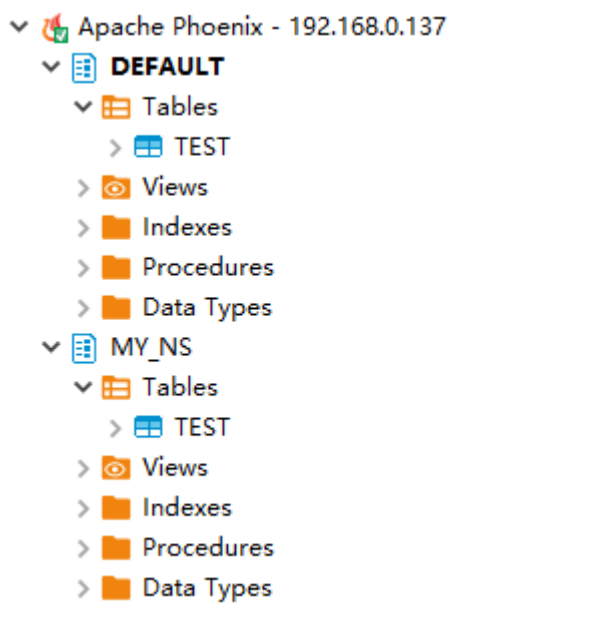
```
UPSERT INTO MY_NS.TEST VALUES(2,'Tom');
```

```
UPSERT INTO MY_NS.TEST VALUES(3,'Manson');
```

```
UPSERT INTO MY_NS.TEST VALUES(4,'Aurora');
```

- Step 15** Right-click the connection name, click **Refresh**, and click  on the left of the connection name to view the tables created in **DEFAULT** and *MY\_NS*.

**Figure 4-12** Viewing tables



----End

## 4.2 Using DBeaver to Access HetuEngine

Use DBeaver 7.2.0 as an example to describe how to access HetuEngine.

### Prerequisites

- The DBeaver has been installed properly. Download the DBeaver software from <https://dbeaver.io/files/7.2.0/>.
- You have created a human-machine user, for example, **hetu\_user**, in the cluster. For details, see [Creating a HetuEngine User](#). For clusters with Ranger authentication enabled, you need to grant the Ranger permission to

**hetu\_user** based on service requirements. For details, see [Adding a Ranger Access Permission Policy for HetuEngine](#).

- A compute instance has been created and is running properly. For details, see [Creating a HetuEngine Compute Instance](#).

## Procedure

**Step 1** Download the HetuEngine client to obtain the JDBC JAR package.

1. Log in to FusionInsight Manager.
2. Choose **Cluster > Services > HetuEngine > Dashboard**.
3. In the upper right corner of the page, choose **More > Download Client** and download the **Complete Client** to the local PC as prompted.
4. Decompress the HetuEngine client package **FusionInsight\_Cluster\_Cluster ID\_HetuEngine\_Client.tar** to obtain the JDBC file and save it to a local directory, for example, **D:\test**.

### NOTE

How to obtain the JDBC file:

Decompress the package in the **FusionInsight\_Cluster\_Cluster ID\_HetuEngine\_ClientConfig\HetuEngine\xxx\** directory to obtain the **hetu-jdbc-\*.jar** file.

Note: *xxx* can be **arm** or **x86**.

**Step 2** Add the host mapping to the local **hosts** file.

Add the mapping of the host where the instance is located in the HSFabric or HSBroker mode. The format is *Host IP address Host name*.

Example: **192.168.42.90 server-2110081635-0001**

### NOTE

The local **hosts** file in a Windows environment is stored in, for example, **C:\Windows\System32\drivers\etc**.

**Step 3** Open DBeaver, choose **Database > New Database Connection**, search for **PrestoSQL** in **ALL**, and open PrestoSQL.

**Step 4** Click **Edit Driver Settings** and set parameters by referring to the following table.

**Table 4-1** Driver settings

| Parameter  | Value Description              |
|------------|--------------------------------|
| Class Name | io.prestosql.jdbc.PrestoDriver |



| Parameter    | Value Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| URL Template | <ul style="list-style-type: none"> <li>Access HetuEngine with HSFabric<br/> <pre>jdbc:presto:// &lt;HSFabricIP1:port1&gt;,&lt;HSFabricIP2:port2&gt;,&lt;HSFabricIP3:port3&gt;/hive/ default?serviceDiscoveryMode=hsfabric</pre>                     The following is an example.<br/> <pre>jdbc:presto:// 192.168.42.90:29902,192.168.42.91:29902,192.168.42.92:29902/hive/ default?serviceDiscoveryMode=hsfabric</pre> </li> <li>Access HetuEngine with HSBroker<br/> <pre>jdbc:presto:// &lt;HSBrokerIP1:port1&gt;,&lt;HSBrokerIP2:port2&gt;,&lt;HSBrokerIP3:port3&gt;/ hive/default?serviceDiscoveryMode=hsbroker</pre>                     The following is an example.<br/> <pre>jdbc:presto:// 192.168.42.90:29860,192.168.42.91:29860,192.168.42.92:29860/hive/ default?serviceDiscoveryMode=hsbroker</pre> </li> </ul> |

 NOTE

- To obtain the IP addresses and port numbers of the HSFabric and HSBroker nodes, perform the following operations:
  - Log in to FusionInsight Manager.
  - Choose **Cluster > Services > HetuEngine**. Click the **Instance** tab to obtain the service IP addresses of all HSFabric or HSBroker instances. You can select one or more normal instances for connection.
  - To obtain the port numbers, choose **Cluster > Services > HetuEngine**. Click **Configurations** then **All Configurations**.  
 Search for **gateway.port** to obtain the HSFabric port number. The default port number is **29902** in security mode and **29903** in normal mode.  
 Search for **server.port** to obtain the HSBroker port number. The default port number is **29860** in security mode and **29861** in normal mode.
- If the connection fails, disable the proxy and try again.

**Step 5** Click **Add File** and upload the JDBC driver package obtained in [Step 1](#).

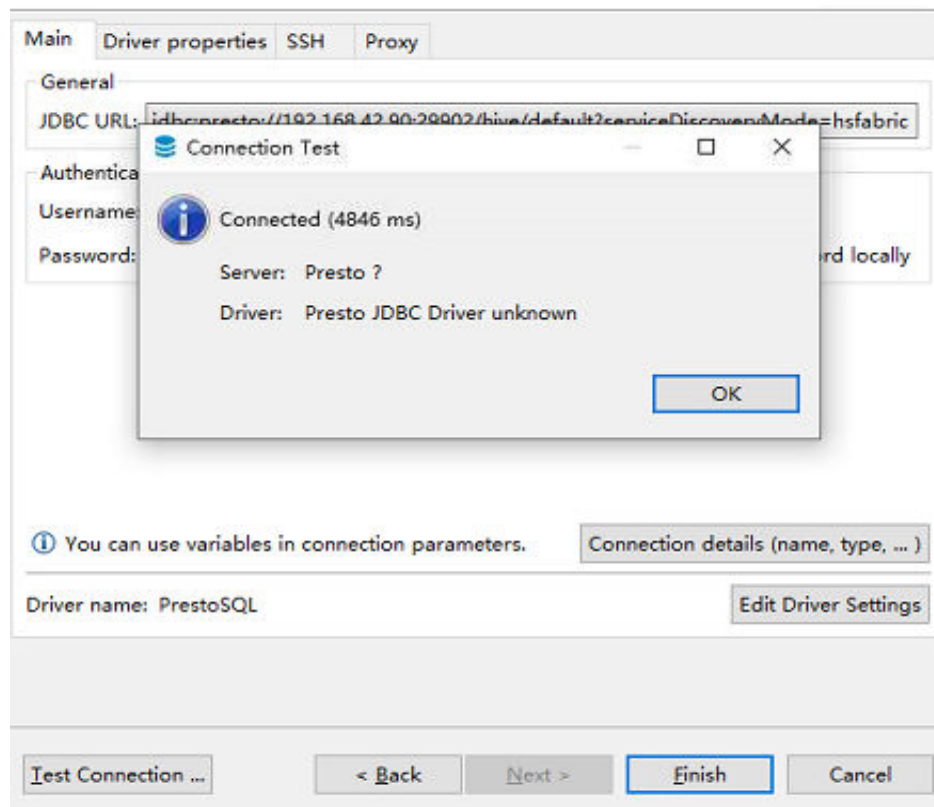
**Step 6** Click **Find Class**. The driver class is automatically obtained. Click **OK** to complete the driver setting. If **io.prestosql:presto-jdbc:RELEASE** exists in **Libraries**, delete it before clicking **Find Class**.

**Figure 4-13** Configuring the driver in security mode

**Step 7** Configure the connection.

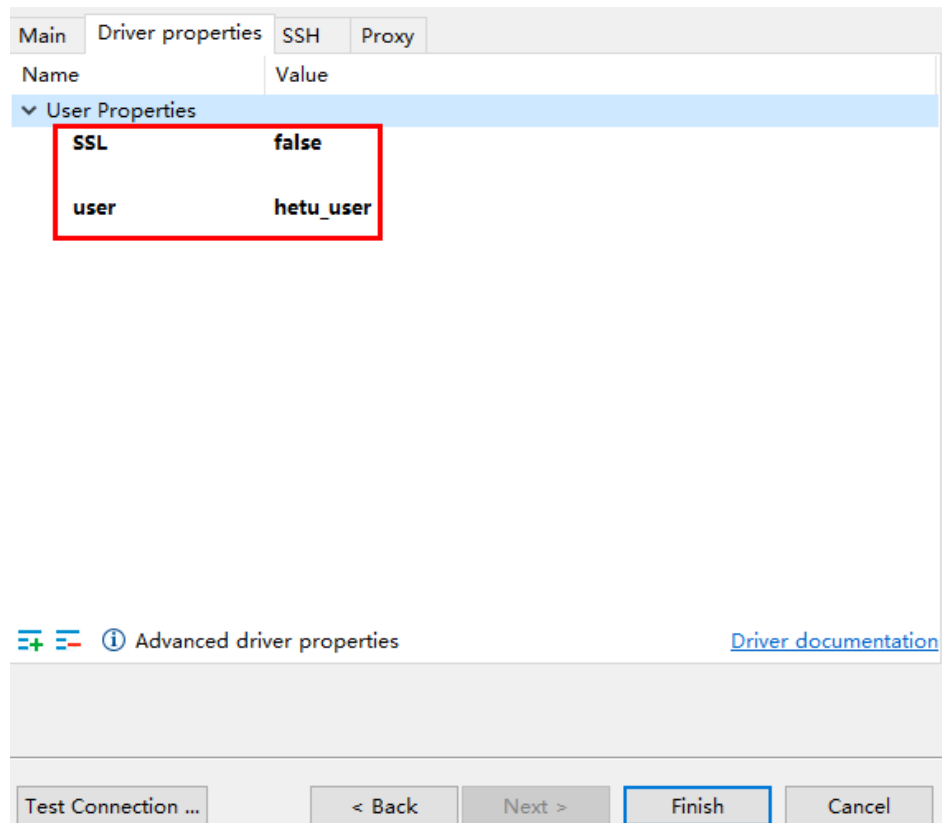
- Security mode (clusters with Kerberos authentication enabled):  
On the **Main** tab page for creating a connection, enter the username and password created in **Prerequisites**, and click **Test Connection**. After the connection is successful, click **OK** then **Finish**. You can click **Connection details (name, type, ... )** to change the connection name.

**Figure 4-14** Configuring parameters on the Main tab in security mode



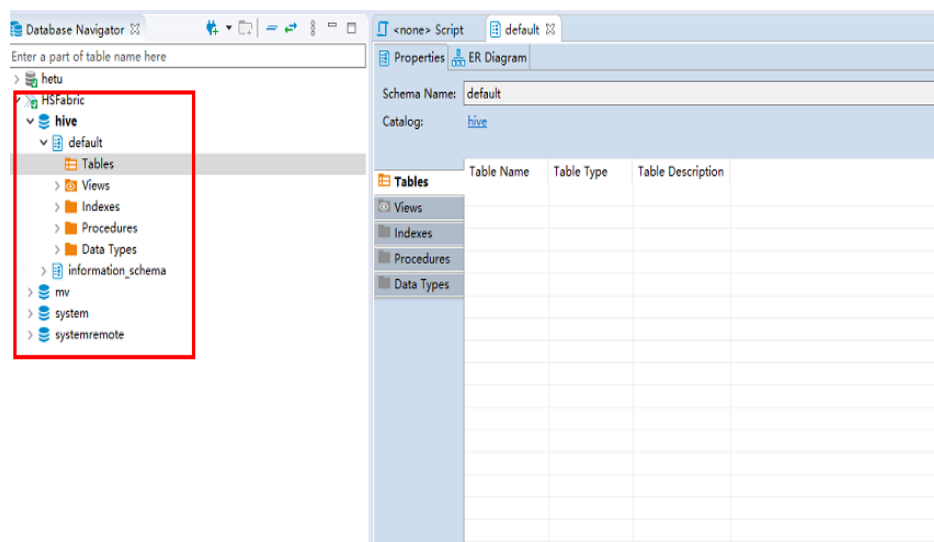
- Normal mode (clusters with Kerberos authentication disabled):  
On the page for creating a connection, configure the parameters on the **Driver properties** tab. Set **user** to the user created in **Prerequisites**. Click **Test Connection**. After the connection is successful, click **OK** then **Finish**. You can click **Connection details (name, type, ... )** to change the connection name.

**Figure 4-15** Configuring parameters on the Driver properties tab in normal mode



**Step 8** After the connection is successful, the page shown in **Figure 4-16** is displayed.

**Figure 4-16** Successful connection



----End

## 4.3 Using Tableau to Access HetuEngine

Use Tableau Desktop 2022.2 as an example to describe how to access HetuEngine in a security cluster.

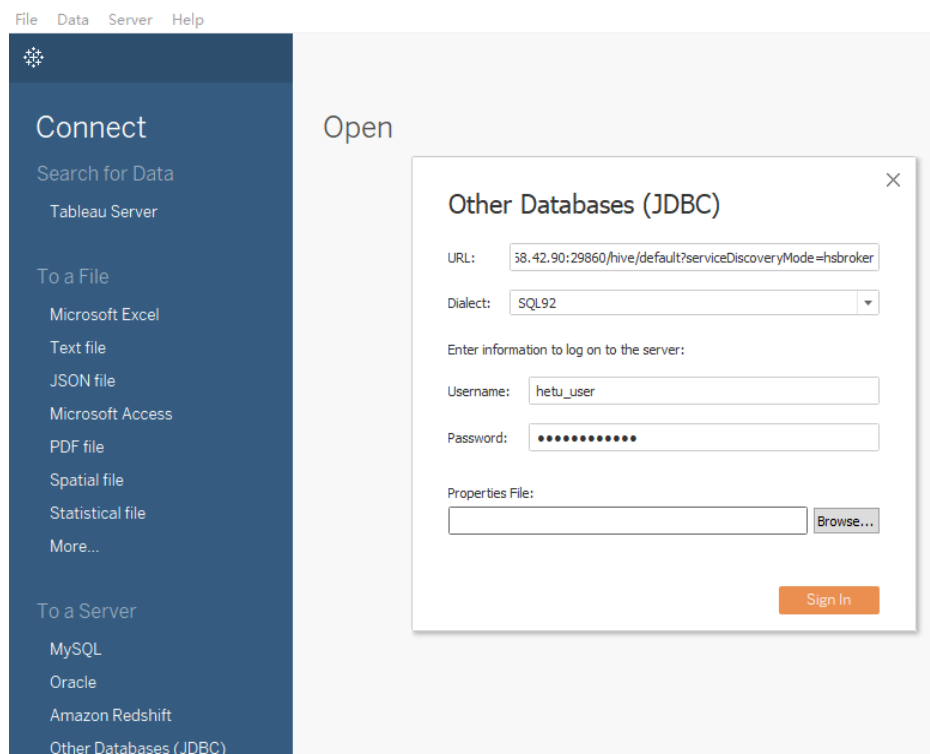
### Prerequisites

- Tableau Desktop has been installed.
- The JDBC JAR file has been obtained. For details, see [Step 1](#).
- You have created a human-machine user, for example, **hetu\_user**, in the cluster. For details, see [Creating a HetuEngine User](#). For clusters with Ranger authentication enabled, you need to grant the Ranger permission to **hetu\_user** based on service requirements. For details, see [Adding a Ranger Access Permission Policy for HetuEngine](#).
- A compute instance has been created and is running properly. For details, see [Creating a HetuEngine Compute Instance](#).

### Procedure

**Step 1** Place the obtained JAR file to the Tableau installation directory, for example, **C:\Program Files\Tableau\Drivers**.

**Step 2** Open Tableau, choose **To a Server > Other Databases (JDBC)**, enter the URL and the username and password of the created human-machine user, and click **Sign In**. HetuEngine is accessible either with HSFabric or HSBroker. For details about the URL format, see [Table 4-1](#).



**Step 3** After the login is successful, drag the desired data table to the operation window on the right and refresh data.

----End

## 4.4 Using Yonghong BI to Access HetuEngine

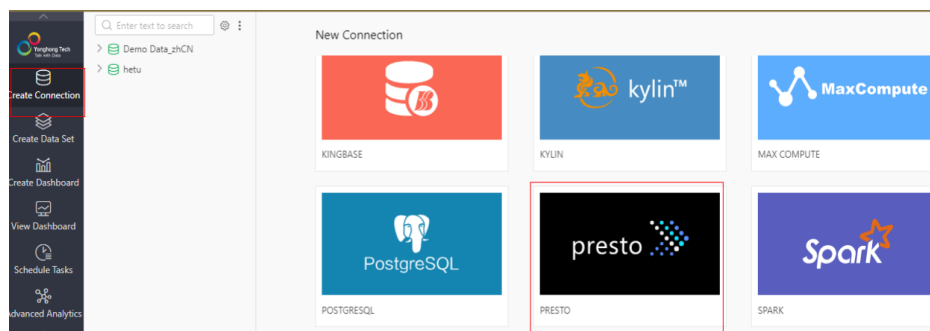
Use Yonghong Desktop 9.1 as an example to describe how to access HetuEngine in a security cluster.

### Prerequisites

- Yonghong Desktop has been installed.
- The JDBC JAR file has been obtained. For details, see [Step 1](#).
- You have created a human-machine user, for example, **hetu\_user**, in the cluster. For details, see [Creating a HetuEngine User](#). For clusters with Ranger authentication enabled, you need to grant the Ranger permission to **hetu\_user** based on service requirements. For details, see [Adding a Ranger Access Permission Policy for HetuEngine](#).
- A compute instance has been created and is running properly. For details, see [Creating a HetuEngine Compute Instance](#).

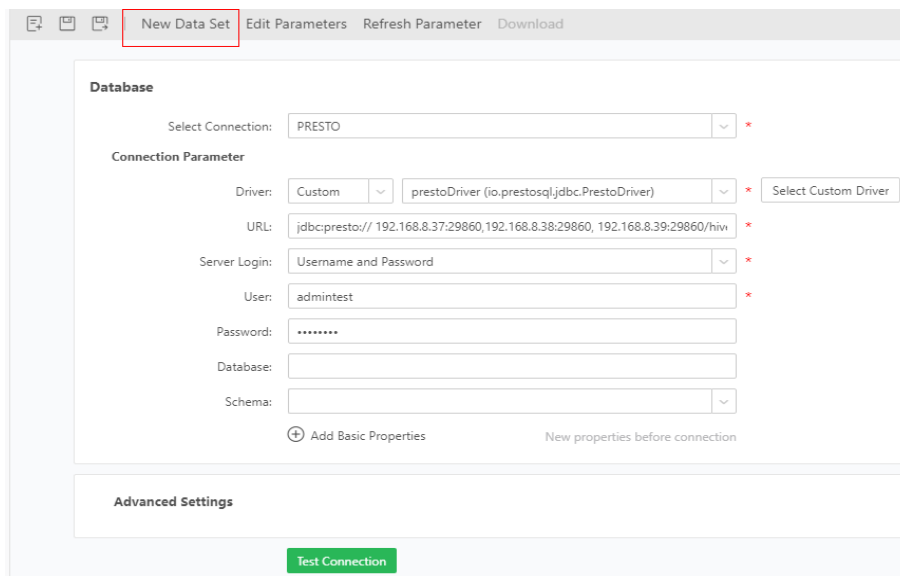
### Procedure


**Step 1** Open Yonghong Desktop and choose **Create Connection > presto**.

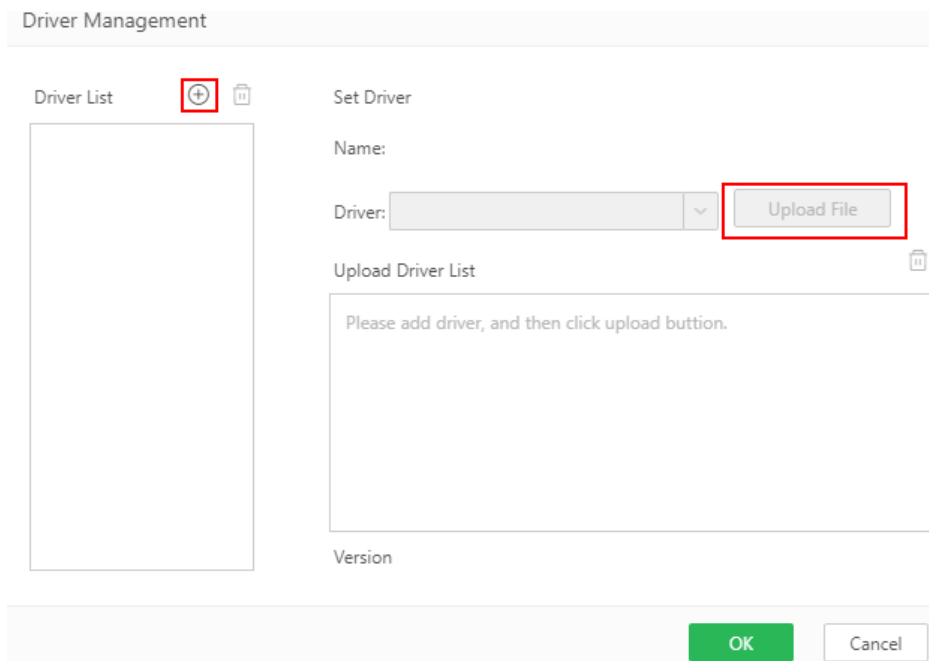


**Step 2** On the data source configuration page, set parameters by referring to [Figure 4-17](#). **User** and **Password** are the username and password of the created human-machine user. After the configuration is complete, click **Test Connection**.

**Figure 4-17** Configuring the data source



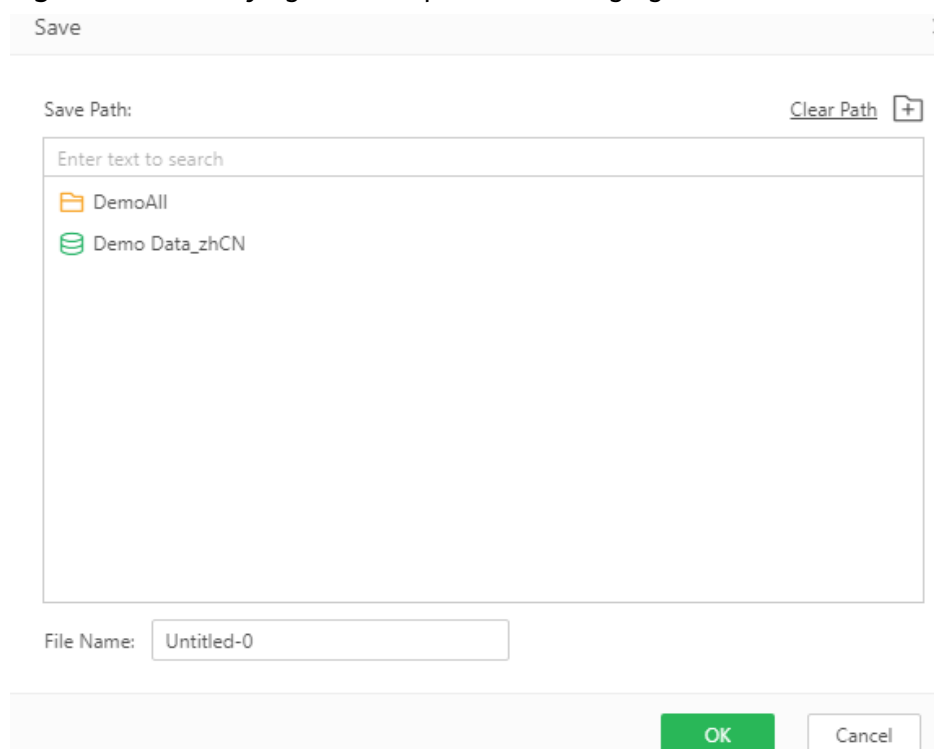
- **Driver:** Choose **Custom** > **Select Custom Driver**. Click , edit the driver name, click **Upload File** to upload the obtained JDBC JAR file, and click **OK**.



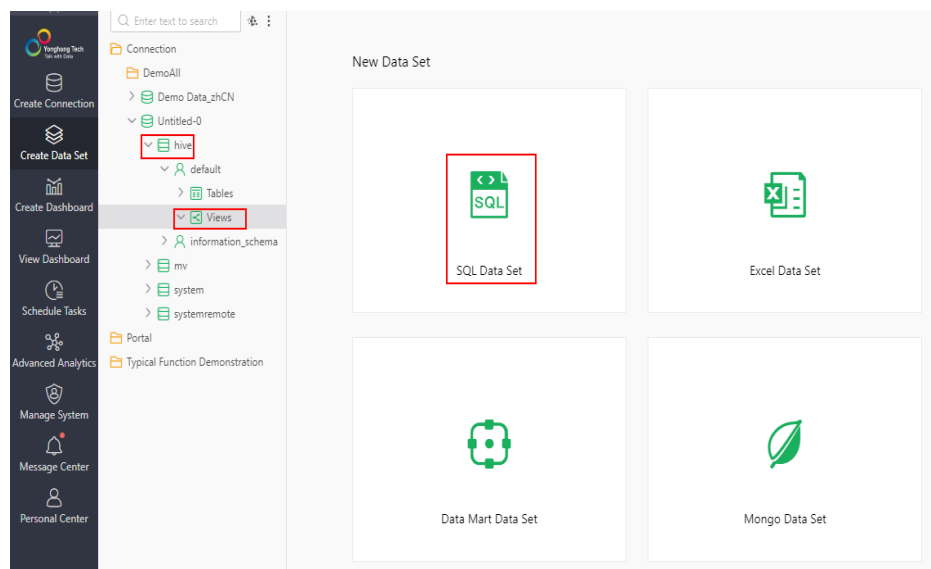
- **URL:** Enter the URL required for using either HSFabric or HSBroker. For details, see [Table 4-1](#).
- **Server Login:** Select **Username and Password** and enter the username and password.

**Step 3** Click **New Data Set**. On the page that is displayed, modify the save path and change the file name by referring to [Figure 4-18](#), and click **OK**.

**Figure 4-18** Modifying the save path and changing the file name

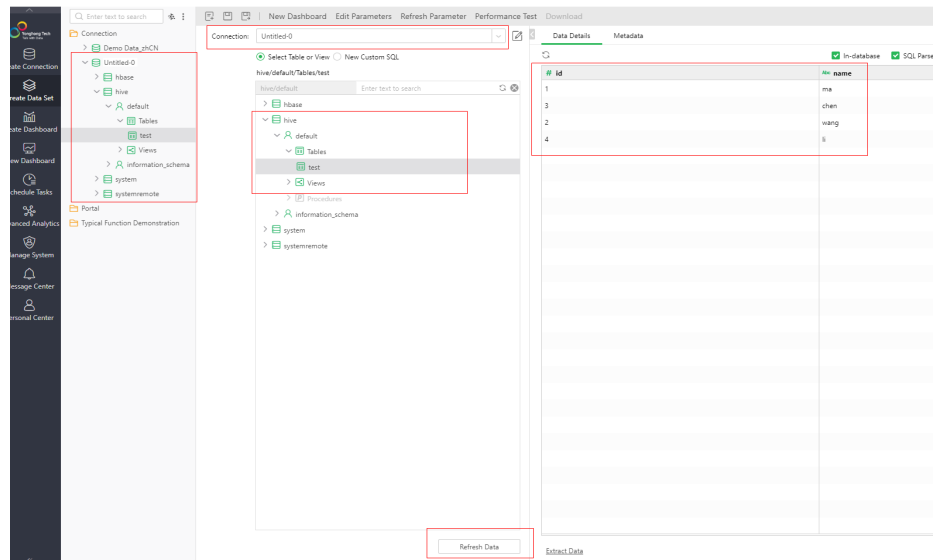


**Step 4** Select the file name of the data set created in **Step 3** under **DemoAll**. The default file name **Untitled-0** is used as an example. Choose **Untitled-0 > hive > default > Views** and select **SQL Data Set** under **New Data Set** in the right pane.



**Step 5** In the **Connection** area, select the new data set created in **Step 3**. All table information is displayed. Select a table, for example, **test**, and click **Refresh Data**. All table information is displayed in the **Data Details** area on the right.





----End

## 4.5 Interconnecting Hive with External Self-Built Relational Databases

### NOTE

- This section applies to MRS 3.x or later.
- This section describes how to connect Hive with built-in relational databases open-source MySQL and Postgres.
- After an external metadata database is deployed in a cluster with Hive data, the original metadata tables will not be automatically synchronized. Before installing Hive, determine whether to store metadata in an external database or DBService. For the former, deploy an external database when installing Hive or when there is no Hive data. After Hive installation, the metadata storage location cannot be changed. Otherwise, the original metadata will be lost.
- After external metadata is imported to the MySQL database, Hive supports only table names, field names, and table description in Chinese.

**Hive supports access to open source MySQL and Postgres metabases.**

**Step 1** Install the open source MySQL or Postgres database.

### NOTE

The node where the database is installed must be in the same network segment as the cluster, so that they can access each other.

**Step 2** Upload the driver package.

- PostgreSQL:
  - Use the open source driver package to replace the cluster's existing one. Download the open source PostgreSQL driver package **postgresql-42.2.5.jar** at <https://repo1.maven.org/maven2/org/postgresql/postgresql/42.2.5/> and upload it to the `$(BIGDATA_HOME)/third_lib/Hive` directory on all MetaStore nodes.

Run the following commands on all MetaStore nodes to modify the permission on the driver package:

```
cd ${BIGDATA_HOME}/third_lib/Hive
chown omm:wheel postgresql-42.2.5.jar
chmod 600 postgresql-42.2.5.jar
```

- MySQL:

Visit the MySQL official website at <https://www.mysql.com/>, choose **DOWNLOADS > MySQL Community(GPL) DownLoads > Connector/J**, and download the driver package of the required version.

- For versions earlier than MRS 8.2.0, upload the MySQL driver package of the required version to the `/opt/Bigdata/FusionInsight_HD_*/install/FusionInsight-Hive-*/hive-*/lib/` directory on all Metastore nodes.
- For MRS 8.2.0 and later versions, upload the MySQL driver package of the required version to the `${BIGDATA_HOME}/third_lib/Hive` directory on all Metastore nodes.

Run the following commands on all MetaStore nodes to modify the permission on the driver package:

```
cd /opt/Bigdata/FusionInsight_HD_*/install/FusionInsight-Hive-*/hive-*/lib/
chown omm:wheel mysql-connector-java-*.jar
chmod 600 mysql-connector-java-*.jar
```

**Step 3** Create a user and metadata database in the self-built database and assign all permissions on the database to the user. For example:

- Run the following commands as the database administrator in PostgreSQL to create database **test** and user **testuser**, and assign all permissions on **test** to **testuser**:

```
create user testuser with password 'password';
create database test owner testuser;
grant all privileges on database test to testuser;
```

- Run the following commands as the database administrator in MySQL to create database **test** and user **testuser**, and assign all permissions on **test** to **testuser**:

```
create database test;
create user 'testuser'@'%' identified by 'password';
grant all privileges on test.* to 'testuser';
flush privileges;
```

**Step 4** Import the SQL statements for creating metadata tables.

- SQL script path in the PostgreSQL database: `${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-Hive-*/hive-*/scripts/metastore/upgrade/postgres/hive-schema-3.1.0.postgres.sql`

Run the following command to import the SQL file to Postgres:

```
./bin/psql -U username -d databasename -f hive-schema-3.1.0.postgres.sql
```

`./bin/psql` is in the Postgres installation directory.

`username` indicates the username for logging in to Postgres.

*dbname* indicates the database name.

- SQL script path in the MySQL database: `${BIGDATA_HOME}/FusionInsight_HD_*/install/FusionInsight-Hive-*/hive-*/scripts/metastore/upgrade/mysql/hive-schema-3.1.0.mysql.sql`

Run the following command to import the SQL file to the MySQL database:

```
./bin/mysql -u username -p -D dbname<hive-schema-3.1.0.mysql.sql
```

`./bin/mysql` is in the MySQL installation directory.

*username* indicates the user name for logging in to MySQL.

*dbname* indicates the database name.

- Step 5** Log in to FusionInsight Manager, choose **Cluster > Services**, and click **Hive**. On the displayed page, click **Configuration > All Configurations**, and choose **Hive (Service) > MetaDB**. Modify the parameters described in [Table 1 Parameters](#), and save the modification so that the Hive configuration can be connected to the open-source database.

**Table 4-2** Parameters

| Parameter                             | Default value         | Description                                                                                                                                                                                                                                                                                         |
|---------------------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| javax.jdo.option.ConnectionDriverName | org.postgresql.Driver | Driver class for connecting metadata on MetaStore <ul style="list-style-type: none"> <li>• If an external MySQL database is used, the value is: <code>com.mysql.jdbc.Driver</code></li> <li>• If an external Postgres database is used, the value is: <code>org.postgresql.Driver</code></li> </ul> |

| Parameter                           | Default value                                                                                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| javax.jdo.option.ConnectionURL      | jdbc:postgresql://%<br>{DBSERVICE_FLOAT_IP}%<br>{DBServer}:%<br>{DBSERVICE_CPORT}/<br>hivemeta?<br>socketTimeout=60 | <p>URL of the JDBC link of the MetaStore metadata</p> <ul style="list-style-type: none"> <li>If an external MySQL database is used, the value is:<br/><b>jdbc:mysql://IP<br/>address of the MySQL<br/>database:Port number<br/>of the MySQL<br/>database/test?<br/>characterEncoding=utf-8</b></li> <li>If an external Postgres database is used, the value is:<br/><b>jdbc:postgresql://IP<br/>address of the<br/>PostgreSQL<br/>database:Port number<br/>of the PostgreSQL<br/>database/test</b></li> </ul> <p><b>NOTE</b><br/><b>test</b> is the name of the database created in MySQL or PostgreSQL in <a href="#">Step 3</a>.</p> |
| javax.jdo.option.ConnectionUserName | hive\${SERVICE_INDEX}\$<br>{SERVICE_INDEX}                                                                          | Username for connecting to the metadata database on Metastore                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

**Step 6** Change the Postgres database password in MetaStore. Choose **Cluster > Name of the desired cluster > Services > Hive**. On the displayed page, click **Configurations > All Configurations** and choose **MetaStore(Role) > MetaDB**, modify the following parameters, and click **Save**.

**Table 4-3** Parameters

| Parameter                                  | Default value | Description                                                                                                               |
|--------------------------------------------|---------------|---------------------------------------------------------------------------------------------------------------------------|
| javax.jdo.option.extend.ConnectionPassword | *****         | User password for connecting to the external metadata database on Metastore. The password is encrypted in the background. |

- Step 7** Log in to each MetaStore background node and check whether the local directory `/opt/Bigdata/tmp` exists.
- If it exists, go to [Step 8](#).
  - If it is not, run the following commands to create one:  
**mkdir -p /opt/Bigdata/tmp**  
**chmod 755 /opt/Bigdata/tmp**
- Step 8** Save the configuration. Choose **Dashboard > More > Restart Service**, and enter the password to restart the Hive service.
- Step 9** Log in to the MySQL or PostgreSQL database and view metadata tables generated in the metadata database created in [Step 3](#).

```
Tables_in_hivemeta
aux_table
bucketing_cols
cds
columns_v2
compaction_queue
completed_compactions
completed_txn_components
ctlgs
database_params
db_privs
dbs
delegation_tokens
```

- Step 10** Check whether the metadata database is successfully deployed.
1. Log in to the node where the Hive client is installed as the client installation user.  
**cd Client installation directory**  
**source bigdata\_env**  
**kinit Component service user** (Skip this step for clusters with Kerberos authentication disabled.)
  2. Run the following command to log in to the Hive client CLI:  
**beeline**
  3. Run the following command to create the **test** table:  
**create table test(id int,str1 string,str2 string);**
  4. Run the following command in the **test** database of the MySQL or PostgreSQL database to check whether there is any information about the **test** table:  
**select \* from TBLS;**  
If information about the **test** table is displayed, the external database is successfully deployed. For example:
    - The result in the MySQL database is as follows:

```
mysql> mysql> select * from TBLS;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| TBL_ID | CREATE_TIME | DB_ID | LAST_ACCESS_TIME | OWNER | OWNER_TYPE | RETENTION | SD_ID | TBL_NAME | TBL_TYPE | VIEW_EXPANDED_TEXT | VIEW_ORIGINAL_TEXT | IS_REWRITE_ENABLED |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 6 | 1673413291 | 1 | 0 | root | USER | 0 | 6 | test1 | MANAGED_TABLE | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

- The result in the PostgreSQL database is as follows:

```
hive> select * from "TBL5";
TBL_ID	CREATE_TIME	DB_ID	LAST_ACCESS_TIME	OWNER	OWNER_TYPE	RETENTION	SD_ID	TBL_NAME	TBL_TYPE	VIEW_EXPANDED_TEXT	VIEW_ORIGINAL_TEXT	IS_REWRITE_ENABLED
2 | 1673425195 | 1 | 0 | root | USER | 0 | 2 | test1 | MANAGED_TABLE | | | f
(1 row)
```

----End

## 4.6 Interconnecting Hive with CSS

### Scenario

Use the Elasticsearch-Hadoop plug-in to exchange data between Hive and Elasticsearch of Cloud Search Service (CSS) so that Elasticsearch index data can be mapped to Hive tables.

#### NOTE

This section applies to MRS 3.x or later.

### Prerequisites

The Hive service of MRS and the Elasticsearch service of CSS have been installed, and the two clusters can communicate with each other.

### Procedure

- Step 1** On the **Clusters** page of the CSS console, locate the row containing the target cluster and click **Access Kibana** in the **Operation** column. In the navigation pane of Kibana, click **Dev Tools**. On the console page that is displayed, run the following command to create the index **ddj\_study\_card\_ratio\_v12**:

```
PUT /ddj_study_card_ratio_v12
{
 "mappings": {
 "properties": {
 "uniq_id": {
 "type": "text",
 "fields": {
 "keyword": {
 "type": "keyword",
 "ignore_above": 256
 }
 }
 }
 }
 }
}
```

If the following information is displayed, the index is created:

```
{
 "acknowledged": true,
 "shards_acknowledged": true,
 "index": "ddj_study_card_ratio_v12"
}
```

- Step 2** Run the following command to insert data into the **ddj\_study\_card\_ratio\_v12** index:

```
POST /ddj_study_card_ratio_v12/_doc/_bulk
{"index":{}}
{"id":"1", "uniq_id":"23323"}
```

If **errors** is **false** in the command output, the data is imported.

**Step 3** Download the corresponding JAR file from [Past Releases](#) based on the Elasticsearch version in CSS.

For example, the JAR file corresponding to Elasticsearch 7.6.2 is **elasticsearch-hadoop-7.6.2.jar**.

 **NOTE**

- The JAR file and Elasticsearch of CSS must have the same version. This section uses an Elasticsearch 7.6.2 cluster with security mode enabled as an example.
- If there are any additional custom modules, pack them into a separate JAR file.

**Step 4** Upload the JAR file in [Step 3](#) to the **/opt/Bigdata/third\_lib/Hive** directory on all HiveServer nodes and run the following command to modify the permission:

```
chown omm:wheel -R /opt/Bigdata/third_lib/Hive
```

**Step 5** Log in to FusionInsight Manager and choose **Cluster > Services > Hive** . On the page that is displayed, click the **Instance** tab. On this tab page, select all HiveServer instances, and choose **More > Restart Instance**.

**Step 6** Download **commons-httpclient-3.1.jar** from [Maven central warehouse](#) and upload this JAR file and the JAR file in [Step 3](#) to any node where the HDFS and Hive clients are installed in the cluster.

**Step 7** Log in to the node to which the JAR files in [Step 6](#) are uploaded as the client installation user.

**Step 8** Run the following command to authenticate the user:

```
cd Client installation directory
```

```
source bigdata_env
```

```
kinit Component service user (Skip this step for clusters with Kerberos authentication disabled.)
```

**Step 9** Run the following command to create a directory for storing JAR files in HDFS:

```
hdfs dfs -mkdir HDFS path for storing JAR files
```

**Step 10** Run the following command to upload the JAR files in [Step 6](#) to HDFS:

```
hdfs dfs -put JAR file storage path HDFS path for storing JAR files
```

**Step 11** Run the following command to enable Hive to load a specified JAR file when executing a command line task:

```
beeline
```

```
add jar HDFS path for storing JAR files; (Execute this command once for each JAR file.)
```

**Step 12** Run the following command to create an Elasticsearch external table:

```
CREATE EXTERNAL TABLE `ddj_study_card_ratio_v12_test` (
 `uniq_id` string)
ROW FORMAT SERDE
 'org.elasticsearch.hadoop.hive.EsSerDe'
STORED BY
 'org.elasticsearch.hadoop.hive.EsStorageHandler'
```

```
WITH SERDEPROPERTIES (
 'field.delim'='',
 'serialization.format'='')
TBLPROPERTIES (
 'bucketing_version'='2',
 'es.index.auto.create'='false',
 'es.mapping.date.rich'='false',
 'es.net.http.auth.pass'='Pzh6537project',
 'es.net.http.auth.user'='elastic',
 'es.nodes'='vpcep-e0b33065-75b7-4193-8395-dbd00d10bc39.cn-east-3.huaweicloud.com',
 'es.nodes.wan.only'='true',
 'es.port'='9200',
 'es.read.metadata'='true',
 'es.resource'='ddj_study_card_ratio_v12',
 'es.set.netty.runtime.available.processors'='false',
 'es.write.operation'='index',
 'last_modified_by'='root',
 'last_modified_time'='1655264909',
 'transient_lastDdlTime'='1655264909');
```

**NOTE**

Key parameters are described as follows:

- **es.net.http.auth.pass** and **es.net.http.auth.user**: indicate the password and username of the user created in Kibana who has the permission to perform operations on indexes created in **Step 1**.
- **es.nodes**: IP address to be connected. You can log in to the CSS management console and view the IP address of the cluster in the **Internal Access Addresses** column of the cluster list.
- **es.port**: port for external access to the Elasticsearch cluster. The default value is **9200**.
- **es.resource**: name of the index created in **Step 1**.

For details about parameter configurations, visit <https://www.elastic.co/guide/en/elasticsearch/hadoop/6.1/hive.html>.

**Step 13** Run the following command to view the Elasticsearch external table created in **Step 12**:

```
select * from ddj_study_card_ratio_v12_test;
```

If no error information is displayed and the query is successful, Hive is interconnected with CSS. The command output is as follows:

```
hdfs://192.168.0.129:10000/> select * from ddj_study_card_ratio_v12_test002;
INFO [State: Compiling,
INFO [Compiling command(queryId=omn_20220727154319_1c7f3fdf-5c8c-4c3d-80a2-f95888bcc5a4): select * from ddj_study_card_ratio_v12_test002; Current sessionId=a9f6dd55-f39e-4000-b010-000000000000]
INFO [hive.compile.auto.avoid.cbo=true
INFO [Concurrency mode is disabled, not creating a lock manager
INFO [Current sql is not contains insert syntax, not need record dest table flag
INFO [Semantic Analysis Completed (retry) = false)
INFO [Returning hive schema: SchemaFieldSchema(FieldSchema(name:ddj_study_card_ratio_v12_test002.uniq_id, type:string, comment:null), properties:null)
INFO [Completed compiling command(queryId=omn_20220727154319_1c7f3fdf-5c8c-4c3d-80a2-f95888bcc5a4); Time taken: 0.086 seconds
INFO [Concurrency mode is disabled, not creating a lock manager
INFO [State: Executing,
INFO [Executing command(queryId=omn_20220727154319_1c7f3fdf-5c8c-4c3d-80a2-f95888bcc5a4): select * from ddj_study_card_ratio_v12_test002; Current sessionId=a9f6dd55-f39e-4000-b010-000000000000]
INFO [Completed executing command(queryId=omn_20220727154319_1c7f3fdf-5c8c-4c3d-80a2-f95888bcc5a4); Time taken: 0.0 seconds
INFO [ok
INFO [Concurrency mode is disabled, not creating a lock manager
ddj_study_card_ratio_v12_test002.uniq_id |
-----+-----+
NULL |
-----+-----+
1 row selected (0.31 seconds)
```

----End

## 4.7 Interconnecting Hive with External LDAP

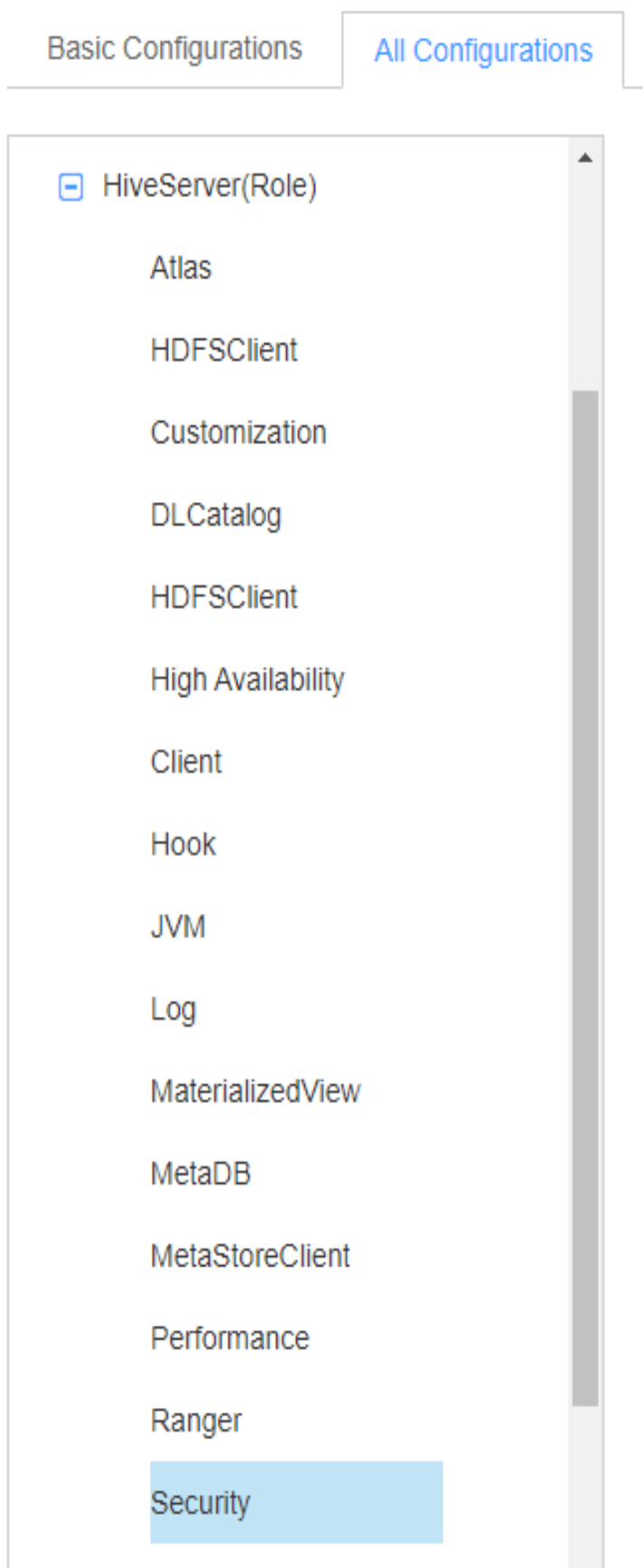
This section applies to MRS 3.1.0 or later.

**Step 1** Log in to FusionInsight Manager.

**Step 2** On FusionInsight Manager, choose **Cluster > Name of the desired cluster > Services > Hive**. On the page that is displayed, click the **Configurations** tab then



the **All Configurations** sub-tab. On this sub-tab page, click **HiveServer(Role)** and select **Security**.



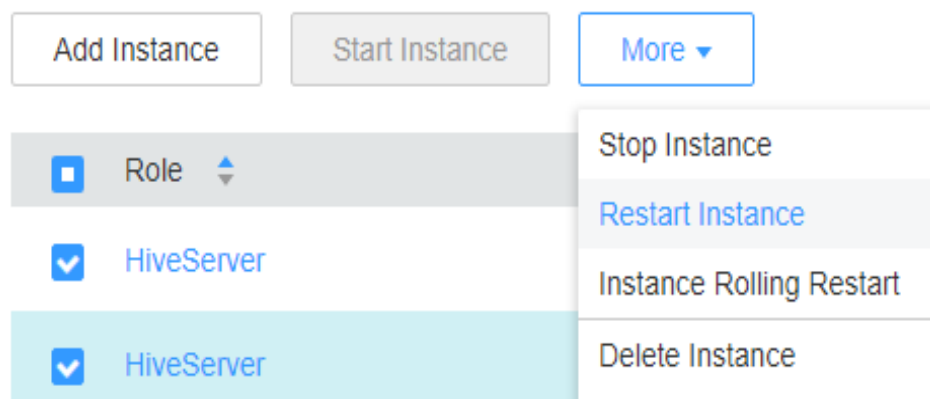
**Step 3** Set the following parameters.

**Table 4-4** Parameter configuration

| Parameter                                      | Description                    | Remarks                                                                                                                             |
|------------------------------------------------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| hive.server2.authentication                    | HiveServer authentication mode | Value: <b>KERBEROS</b> or <b>LDAP</b><br>Default value: <b>KERBEROS</b>                                                             |
| hive.server2.authentication.ldap.baseDN        | LDAP base DN                   | -                                                                                                                                   |
| hive.server2.authentication.ldap.password      | LDAP password                  | LDAP password used for health check                                                                                                 |
| hive.server2.authentication.ldap.url.ip        | LDAP IP address                | -                                                                                                                                   |
| hive.server2.authentication.ldap.url.port      | LDAP port number               | Default value: <b>389</b>                                                                                                           |
| hive.server2.authentication.ldap.userDNPattern | LDAP user DN pattern           | Separate multiple values with colons (:), for example, <b>cn=%s,ou=People1,dc=huawei,dc=com: cn=%s,ou=People2,dc=huawei,dc=com.</b> |
| hive.server2.authentication.ldap.username      | LDAP username                  | LDAP username used for health check                                                                                                 |

**Step 4** After the modification, click **Save** in the upper left corner. In the displayed dialog box, click **OK**.

**Step 5** Choose **Cluster > Name of the desired cluster > Services > Hive > Instance**. On the displayed page, select the instances whose **Configuration Status** is **Expired**, choose **More > Restart Instance**, and restart the instance.



----End