MapReduce Service

Getting Started

Issue 01

Date 2023-11-17

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Buying and Using an MRS Cluster

1.1 Getting Started with MapReduce Service

MapReducce Service (MRS) is a Huawei Cloud service that is used to deploy and manage Hadoop clusters. MRS provides enterprise-class big data clusters on the cloud. Tenants can fully control these clusters and easily run big data components such as Hadoop, Spark, HBase, and Kafka in them.

MRS is easy to use. You can execute various tasks and process or store PB-level data using computers connected in a cluster.

The procedure of using MRS is as follows:

1. Buy a cluster on the MRS console. During this period, you can specify the cluster type, node specifications and count, data disk type (**High I/O** or **Ultra-high I/O**), and components to be installed.
2. Develop a data processing program. For details about how to quickly develop such a program and execute it properly, see the sample code and tutorials provided in **Method of Building an MRS Sample Project**.
3. Upload the prepared program and data files to Object Storage Service (OBS) or the HDFS in the cluster.
4. After a cluster is created, you can directly add jobs and run your programs or SQL statements to process and analyze data.
5. MRS provides you with MRS Manager, an enterprise-class unified management platform of big data clusters, helping you quickly know the health status of services and hosts. Through graphical metric monitoring and customization, you can obtain critical system information in a timely manner. In addition, you can modify service attribute configurations based on service performance requirements, and start or stop clusters, services, and role instances in one click.
6. Terminate the cluster if it is no longer needed after job execution. The terminated cluster is no longer billed.

1.2 Buying a Cluster

To use MRS, buy a cluster on the MRS console. The following procedure takes MRS 3.2.0-LTS.1 as an example to describe how to create a cluster on the MRS management console. Operations for other version are subjective to the UI.

Procedure

Step 1 Go to the **Buy Cluster** page.

Step 2 On the page for buying a cluster, click the **Custom Config** tab.

NOTE

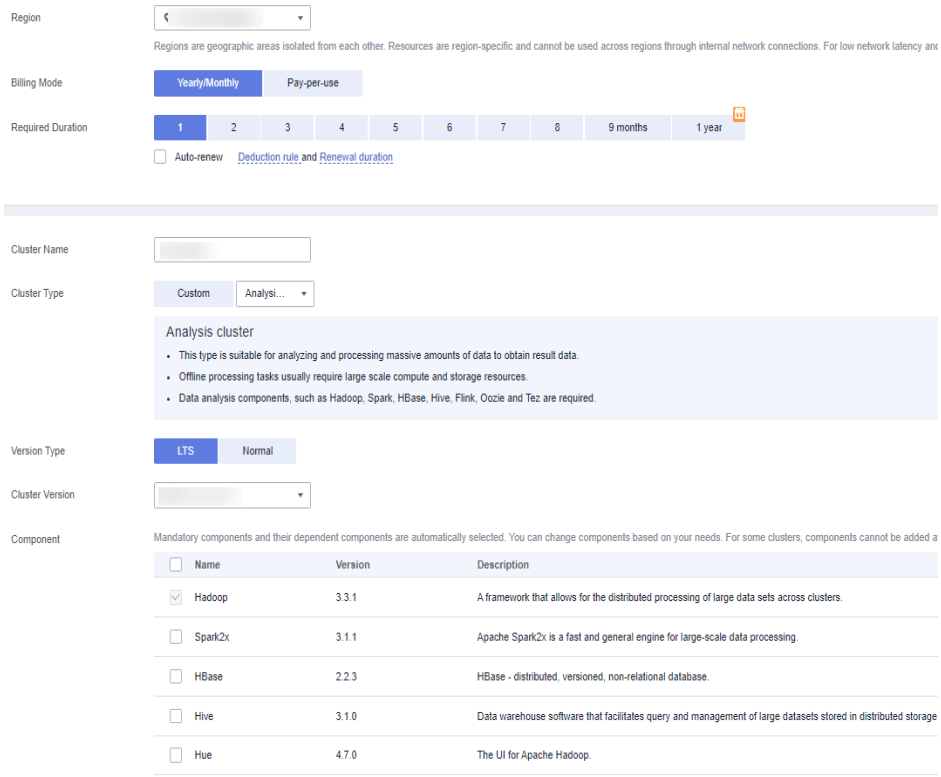
When creating a cluster, pay attention to quota notification. If a resource quota is insufficient, increase the resource quota as prompted and create a cluster.

Step 3 Configure cluster software information.

- **Region**: Retain the default value.
- **Billing Mode**: Retain the default value.
- **Required Duration**: Select a duration as needed.
- **Cluster Name**: You can use the default name. However, you are advised to include a project name abbreviation or date for consolidated memory and easy distinguishing, for example, **mrs_20180321**.
- **Cluster Type**: Select **Analysis cluster**.
- **Version Type**: **Normal** (default) or **LTS**
- **Cluster Version**: Select the latest version, which is the default value.
- **Component**: Select components such as Spark2x, HBase, and Hive for the analysis cluster. For a streaming cluster, select components such as Kafka and Storm. For a hybrid cluster, you can select the components of the analysis cluster and streaming cluster based on service requirements.

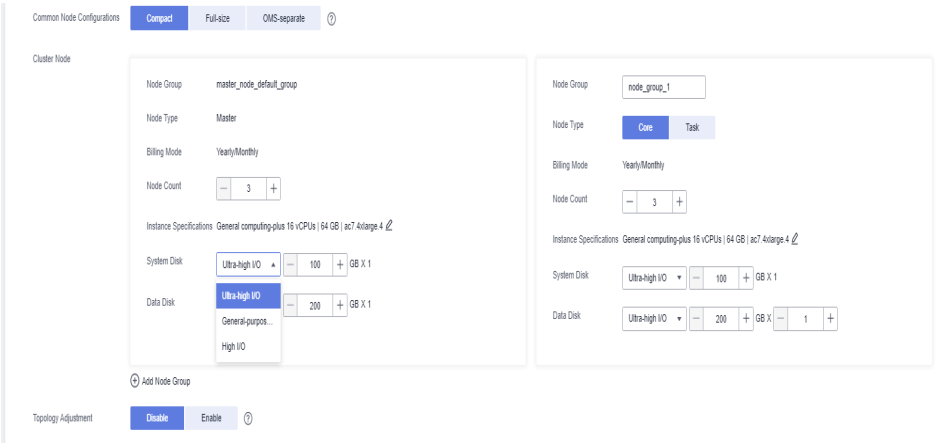
  NOTE

  For versions earlier than MRS 3.*x*, select components such as Spark, HBase, and Hive for an analysis cluster.
- **Metadata**: Retain the default value. This parameter is supported for MRS 3.*x* only.
- Component Port: Policy for setting the default communication port of each component in the cluster.



Step 4 Click **Next**.

- **AZ**: Retain the default value.
- **Enterprise Project**: Select **default**.
- **VPC**: Use the default value. If there is no available VPC, click **View VPC** to access the VPC console and create a new VPC.
- **Subnet**: Retain the default value.
- **Security Group**: Retain the default value.
- **EIP**: Retain the default value.
- **Cluster Node**
  - **Node Count**: the number of nodes you want to purchase. For MRS 3.x clusters, the default value is **3**. You can set the value as you need.
  - **Instance Specifications**: Retain the default settings for master and core nodes or select proper specifications based on service requirements.
  - **System Disk**: Retain the default **Ultra-high I/O** and storage capacity.
  - **Data Disk**: Retain the default **Ultra-high I/O**, storage capacity, and quantity.



Step 5 Click **Next**. The **Set Advanced Options** page is displayed. Configure the following parameters. Retain the default settings for the other parameters.

- Kerberos authentication:
  - **Kerberos Authentication**: Disable Kerberos authentication.
  - **Username**: name of the Manager administrator. **admin** is used by default.
  - **Password**: password of the Manager administrator.
  - **Confirm Password**: Enter the password again.
- **Login Mode**: Select a mode for logging in to an ECS.
  - **Password**: Set a password for logging in to an ECS.
  - **Key Pair**: Select a key pair from the drop-down list. Select "**I acknowledge that I have obtained private key file *SSHkey-xxx* and that without this file I will not be able to log in to my ECS.**" If you have never created a key pair, click **View Key Pair** to create or import a key pair. And then, obtain a private key file.
- **Hostname Prefix**: Prefix for the name of an ECS or BMS in the cluster.

  Enter a maximum of 20 characters that do not start or end with a hyphen (-). Only letters, numbers, and hyphens (-) are allowed.

  When a cluster is created, a DNS domain name is registered for nodes in the cluster. The complete domain name is in the following format: **[prefix]-hostname.mrs-{*XXXX*}.com**. (*XXXX* is a four-character string generated based on the UUID.)
- **Set Advanced Options**: To configure some advanced parameters, select **Configure**.

Step 6 Click **Next**.

- **Configure**: Confirm the parameters configured in the **Configure Software**, **Configure Hardware**, and **Set Advanced Options** areas.
- **Secure Communications**: Select **Enable**.

Step 7 Click **Buy Now**.

If Kerberos authentication is enabled for a cluster, check whether Kerberos authentication is required. If yes, click **Continue**. If no, click **Back** to disable Kerberos authentication and then create a cluster.

Step 8 Click **Back to Cluster List** to view the cluster status.

It takes some time to create a cluster. The initial status of the cluster is **Starting**. After the cluster has been created successfully, the cluster status becomes **Running**.

----End

1.3 Uploading Data

On the **Files** page, you can create and delete HDFS directories, as well as import, export, and delete files in an analysis cluster.

For clusters with Kerberos authentication enabled, synchronize IAM users before performing operations on the **Files** page. On the cluster details page, click **Dashboard** and click **Synchronize** on the right of **IAM User Sync** to synchronize IAM users.

Context

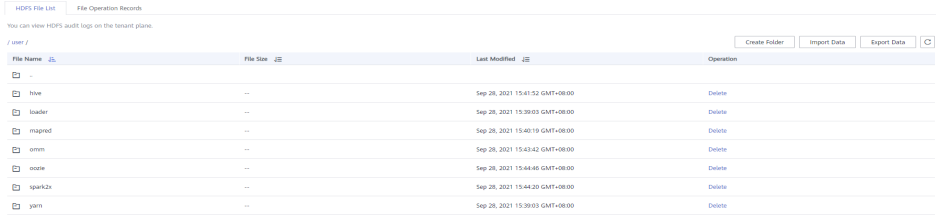
MRS clusters generally process data from OBS or HDFS. OBS provides you with the data storage capabilities that are massive, secure, reliable, and cost-effective. MRS can directly process data in OBS. You can browse, manage, and use data both on the management console and on the OBS Client. If you need to import OBS data into the HDFS system of the cluster for processing, perform the steps in this section.

Importing Data from OBS to HDFS

Currently, MRS supports only data import from OBS to HDFS. The file upload rate decreases with the increase of the file size. This mode applies to scenarios where the data volume is small.

You can perform the following steps to import files and directories:

1. Log in to the MRS console.
2. Choose **Clusters** > **Active Clusters**, and click the name of the target cluster to enter the cluster details page.
3. Click **Files** to go to the file management page.
4. Select **HDFS File List**.



5. Go to the data storage directory, for example, **bd_app1**.

   The **bd_app1** directory is only an example. You can use any directory on the page or create a new one.

   The requirements for creating a folder are as follows:

- The folder name contains a maximum of 255 characters.
- The folder name cannot be empty.
- The folder name cannot contain the following special characters: /:*?"<>|\;&,'`!{}[]$%+
- The value cannot start or end with a period (.).
- The spaces at the beginning and end are ignored.

6. Click **Import Data** and configure the HDFS and OBS paths correctly. When configuring the OBS or HDFS path, click **Browse**, select a file directory, and click **Yes**.

   Figure 1-1 Importing data from OBS to HDFS

   - **OBS Path**
     - The path must start with **obs://**.
     - Files or programs encrypted by KMS cannot be imported.
     - An empty folder cannot be imported.
     - The directory and file name can contain letters, digits, hyphens (-), and underscores (_), but cannot contain special characters ;|&>,<'$*?\
     - The directory and file name cannot start or end with a space, but can contain spaces between them.
     - The OBS full path contains a maximum of 255 characters.
   - **HDFS Path**
     - The path starts with **/user** by default.
     - The directory and file name can contain letters, digits, hyphens (-), and underscores (_), but cannot contain the following special characters: ;|&>,<'$*?\:
     - The directory and file name cannot start or end with a space, but can contain spaces between them.
     - The HDFS full path contains a maximum of 255 characters.

7. Click **OK**.

You can view the file upload progress on the **File Operation Records** page. MRS processes the data import operation as a DistCp job. You can also check whether the DistCp job is successfully executed on the **Jobs** page.

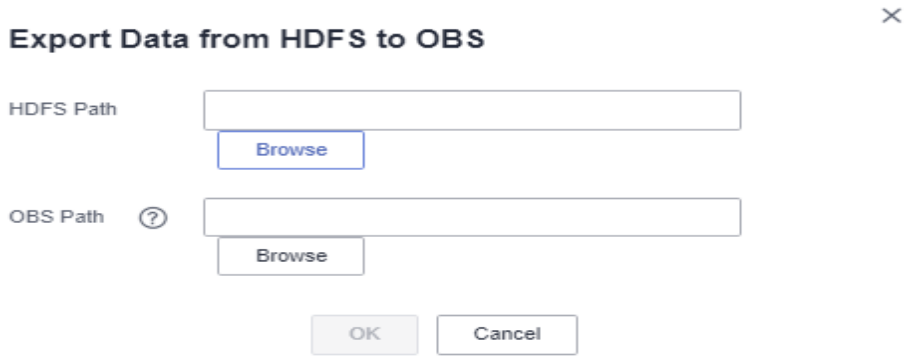
Exporting Data from HDFS to OBS

After data analysis and computing is complete, you can store the data in the HDFS or export it to OBS.

You can perform the following steps to export files and directories:

1. Log in to the MRS console.
2. Choose **Clusters** > **Active Clusters**, and click the name of the target cluster to enter the cluster details page.
3. Click **Files** to go to the file management page.
4. Select **HDFS File List**.
5. Go to the data storage directory, for example, **bd_app1**.
6. Click **Export Data** and configure the OBS and HDFS paths. When configuring the OBS or HDFS path, click **Browse**, select a file directory, and click **Yes**.

Figure 1-2 Exporting data from HDFS to OBS



- **OBS Path**
  - The path must start with **obs://**.
  - The directory and file name can contain letters, digits, hyphens (-), and underscores (_), but cannot contain special characters ;|&>,<'$*?\
  - The directory and file name cannot start or end with a space, but can contain spaces between them.
  - The OBS full path contains a maximum of 255 characters.
- **HDFS Path**
  - The path starts with **/user** by default.
  - The directory and file name can contain letters, digits, hyphens (-), and underscores (_), but cannot contain the following special characters: ;|&>,<'$*?\:

- The directory and file name cannot start or end with a space, but can contain spaces between them.
- The HDFS full path contains a maximum of 255 characters.

NOTE

When a folder is exported to OBS, a label file named **folder name_$folder$** is added to the OBS path. Ensure that the exported folder is not empty. If the exported folder is empty, OBS cannot display the folder and only generates a file named **folder name_$folder$**.

7. Click **OK**.

   You can view the file upload progress on the **File Operation Records** page. MRS processes the data export operation as a DistCp job. You can also check whether the DistCp job is successfully executed on the **Jobs** page.

1.4 Creating a Job

You can submit programs developed by yourself to MRS to execute them, and obtain the results.

This section describes how to submit a job (take a MapReduce job as an example) on the MRS console. MapReduce jobs are used to submit JAR programs to quickly process massive amounts of data in parallel and create a distributed data processing and execution environment.

If the job and file management functions are not supported on the cluster details page, submit the jobs in the background.

Before creating a job, you need to upload local data to OBS for data computing and analyzing. MRS allows exporting data from OBS to HDFS for computing and analyzing. After the data analysis and computing are completed, you can store the data in HDFS or export them to OBS. HDFS and OBS can also store the compressed data in the format of **bz2** or **gz**.

NOTE

If the IAM username contains spaces (for example, **admin 01**), a job cannot be created.

Submitting a Job on the GUI

Step 1 Log in to the MRS console.

Step 2 Choose **Clusters** > **Active Clusters**, select a running cluster, and click its name to access the cluster details page.

Step 3 If Kerberos authentication is enabled for the cluster, perform the following steps. If Kerberos authentication is not enabled for the cluster, skip this step.

In the **Basic Information** area on the **Dashboard** page, click **Synchronize** on the right side of **IAM User Sync** to synchronize IAM users.

Step 4 Click the **Jobs** tab.

Step 5 Click **Create**. The **Create Job** dialog box is displayed.

Step 6 In **Type**, select **MapReduce**. Configure other job information.

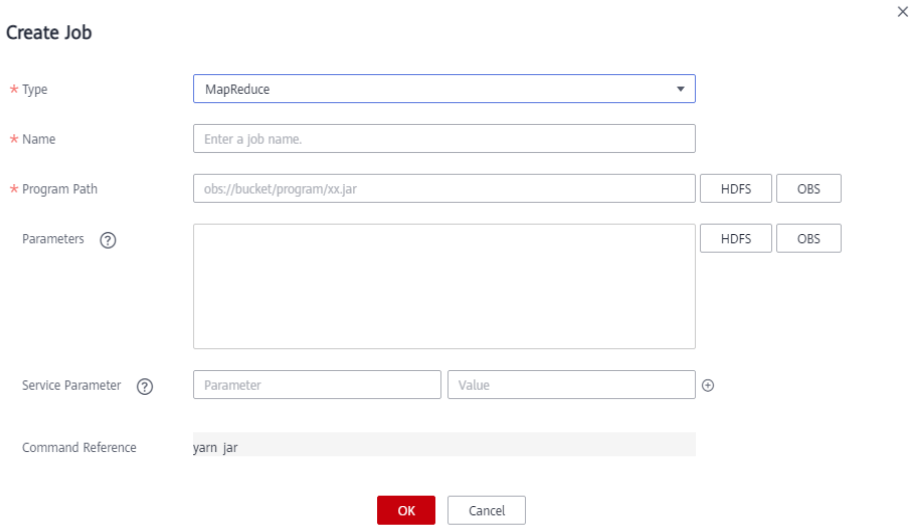


Table 1-1 Job parameters

Parameter	Description
Name	Job name. It contains 1 to 64 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed. **NOTE** You are advised to set different names for different jobs.
Program Path	Path of the program package to be executed. The following requirements must be met: • Contains a maximum of 1,023 characters, excluding special characters such as **;|&><'$**. The parameter value cannot be empty or full of spaces. • The path of the program to be executed can be stored in HDFS or OBS. The path varies depending on the file system. – OBS: The path starts with **obs://**. Example: **obs://wordcount/program/*xxx*.jar** – HDFS: The path must start with **/user**. • For SparkScript and HiveScript, the path must end with **.sql**. For MapReduce, the path must end with **.jar**. For Flink and SparkSubmit, the path must end with **.jar** or **.py**. The **.sql**, **.jar**, and **.py** are case-insensitive.

Parameter	Description
Parameters	(Optional) It is the key parameter for program execution. Separate multiple parameters with space. Configuration method: *Program class name Data input path Data output path* • Program class name: It is specified by a function in your program. MRS is responsible for transferring parameters only. • Data input path: Click **HDFS** or **OBS** to select a path or manually enter a correct path. • Data output path: Enter a directory that does not exist. The value can contain a maximum of 150,000 characters, including special characters (;|&'$), but cannot contain > or <. This parameter can also be left blank. **CAUTION** If you enter a parameter with sensitive information (such as the login password), the parameter may be exposed in the job details display and log printing. Exercise caution when performing this operation.
Service Parameters	(Optional) Used to modify service configuration parameters for the job to be executed. The parameter modification applies only to the job to be executed. To add multiple parameters, click ⊕ on the right. To delete a parameter, click **Delete** on the right. **Table 1-2** describes the common parameters of a service.
Command Reference	Command submitted to the background for execution when a job is submitted.

Table 1-2 Service configuration parameters

Parameter	Description	Example Value
fs.obs.access.key	Key ID for accessing OBS.	-
fs.obs.secret.key	Key corresponding to the key ID for accessing OBS.	-

Step 7 Confirm job configuration information and click **OK**.

After the job is created, you can manage it.

----End

1.5 Terminating a Cluster

You can terminate an MRS cluster that is no longer used after job execution is complete. The terminated or unsubscribed cluster is no longer billed.

Context

Typically after data is analyzed and stored, or when the cluster encounters an exception and cannot work, you can terminate a cluster. A cluster failed to be deployed will be automatically terminated.

Procedure

Step 1 Log in to the MRS management console.

Step 2 In the navigation pane on the left, choose **Clusters** > **Active Clusters**.

Step 3 In the cluster list, locate the row containing the cluster to be deleted, and click **Delete** in the **Operation** column.

The cluster status changes from **Running** to **Deleting**, and finally to **Deleted**. You can view the deleted cluster in **Cluster History**. The deleted cluster is no longer billed.

----End

2 Installing and Using the Cluster Client

Quickly install and use the clients of all services in an MRS 3.*x* or later cluster.

Clients can be installed on the nodes either in or outside the cluster. The following provides an example of how to install and use a client in a cluster.

NOTE

If Flume has been installed in the cluster, the Flume client must be installed independently. For details about how to install the Flume client, see **Installing the Flume Client**.

You can get started by reading the following topics:

1. **Downloading a Client**
2. **Installing a Client**
3. **Using a Client**

Video Tutorial

This video uses an MRS 3.1.0 cluster as an example to describe how to install and use the cluster client after you create a cluster. For details, see **Installing and Using the MRS Client**.

NOTE

The UI may vary depending on the version. The video tutorial is for reference only.

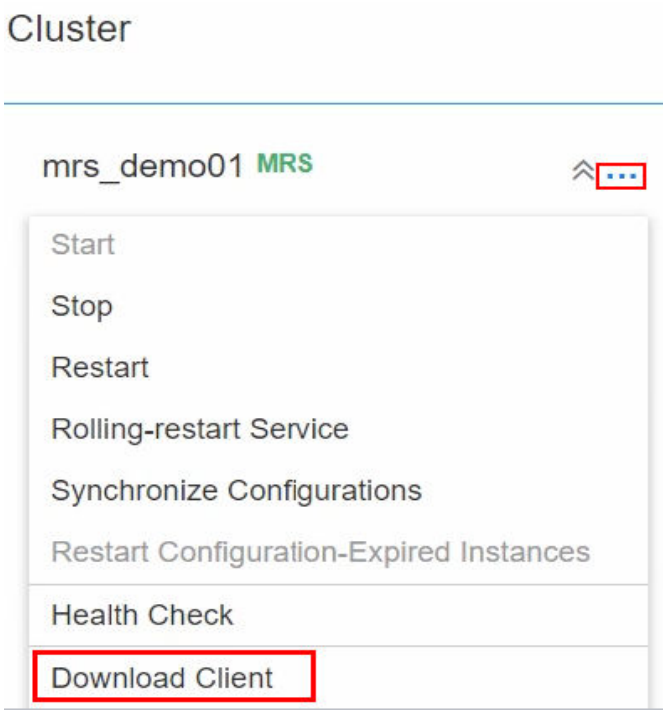
Downloading a Client

Step 1 Log in to FusionInsight Manager of the cluster by referring to **Accessing FusionInsight Manager (MRS 3.x or Later)**.

Step 2 Download the software package of the cluster client to the target node.

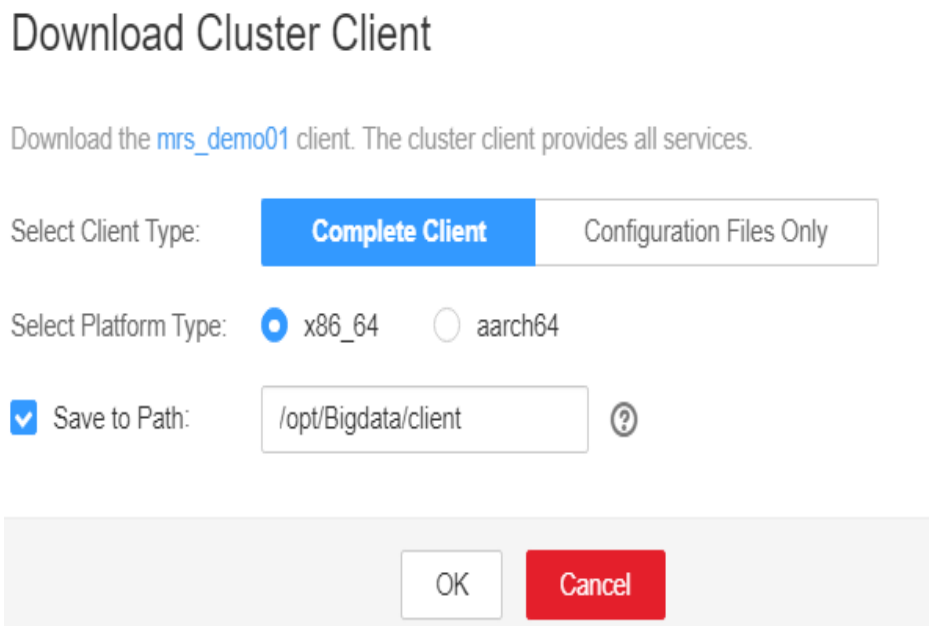
On the home page, click ··· next to the cluster name and click **Download Client** to download the cluster client.

Figure 2-1 Downloading a client



Step 3 On the **Download Cluster Client** page, enter the cluster client download information.

Figure 2-2 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, **/opt/Bigdata/client**. Ensure that user **omm** has the operation permission on the path.

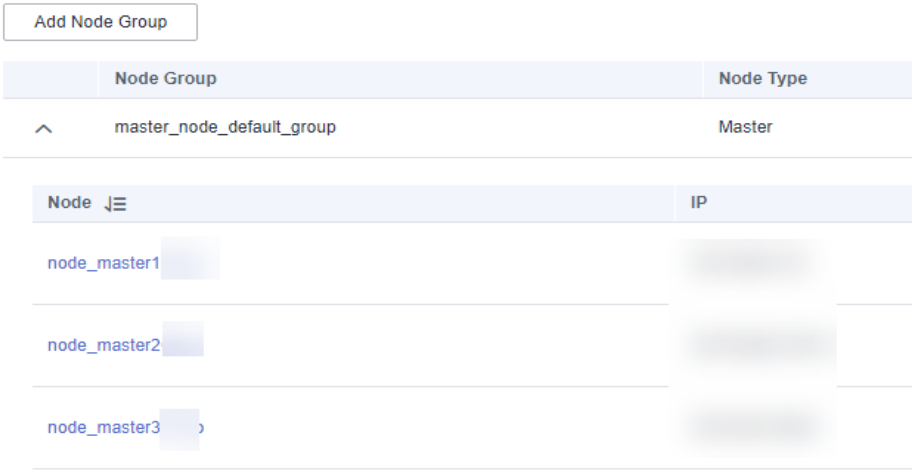
  NOTE

  The cluster supports two types of clients: **x86_64** and **aarch64**. The client type must match the architecture of the node for installing the client. Otherwise, client installation will fail.

Step 4 After the client software package is downloaded, log in to the active OMS node of the cluster as user **root**.

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.

In the cluster list on the MRS console, click the cluster name. On the **Nodes** page, click the name of the target node. On the ECS details page, you can remotely log in to this node.



scp -p /opt/Bigdata/client/FusionInsight_Cluster_1_Services_Client.tar *IP address of the node where the client is to be installed:/opt/Bigdata/client*

----End

Installing a Client

Step 1 Log in to the node where the client software package is installed as the client user (for example, user **root**) and run the following commands to decompress the software package:

cd /opt/Bigdata/client

tar -xvf FusionInsight_Cluster_1_Services_Client.tar

Step 2 Run the **sha256sum** command to verify the decompressed file.

sha256sum -c FusionInsight_Cluster_1_Services_ClientConfig.tar.sha256


```
FusionInsight_Cluster_1_Services_Client.tar: OK
```

Step 3 Decompress the obtained installation file.

tar -xvf FusionInsight_Cluster_1_Services_ClientConfig.tar

Step 4 Go to the directory where the installation package is stored and install the client.

cd /opt/Bigdata/client/FusionInsight_Cluster_1_Services_ClientConfig

Run the following command to install the client to a specified directory (an absolute path), for example, **/opt/hadoopclient**.

./install.sh /opt/hadoopclient

```
...
The component client is installed successfully
```

NOTE

- If the **/opt/hadoopclient** directory has been used by existing service clients, you need to use another directory in this step when installing other service clients.
- You must delete the client installation directory when uninstalling a client.
- If you want to prevent other users from accessing this client, add parameter **-o** during the installation. That is, run the **./install.sh /opt/hadoopclient -o** command to install the client.
- If the NTP server is to be installed in **chrony** mode, ensure that the parameter **chrony** is added during the installation, that is, run the **./install.sh /opt/hadoopclient -o chrony** command to install the client.

----End

Using a Client

Step 1 Log in to the node where the client is installed as the client installation user, and run the following command to switch to the client directory:

cd /opt/hadoopclient

Step 2 Run the following command to load environment variables:

source bigdata_env

Step 3 If Kerberos authentication is enabled for the current cluster, run the following command to authenticate the user. If Kerberos authentication is disabled for the current cluster, authentication is not required.

kinit *MRS cluster user*

For example:

kinit admin

Step 4 Run the client command of a component directly.

For example:

Run the following command to view files in the HDFS root directory:

hdfs dfs -ls /

```
Found 15 items
drwxrwx--x   - hive      hive            0 2021-10-26 16:30 /apps
```

```
drwxr-xr-x   - hdfs      hadoop          0 2021-10-18 20:54 /datasets
drwxr-xr-x   - hdfs      hadoop          0 2021-10-18 20:54 /datastore
drwxrwx---+  - flink     hadoop          0 2021-10-18 21:10 /flink
drwxr-x---   - flume     hadoop          0 2021-10-18 20:54 /flume
drwxrwx--x   - hbase     hadoop          0 2021-10-30 07:31 /hbase
...
```

----End

3 Using Clusters with Kerberos Authentication Enabled

Use security clusters and run MapReduce, Spark, and Hive programs.

In MRS 3.*x*, Presto does not support Kerberos authentication.

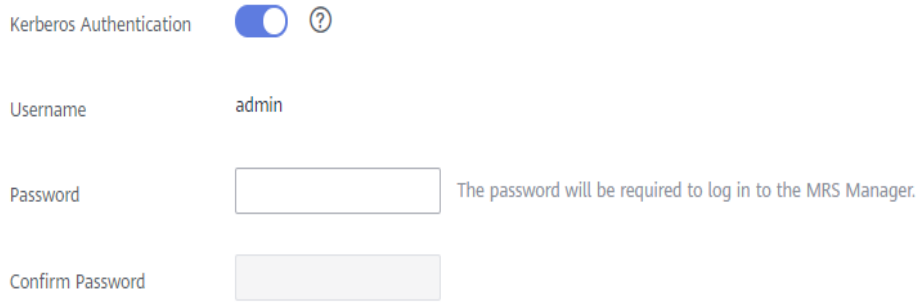
You can get started by reading the following topics:

1. **Creating a Security Cluster and Logging In to Manager**
2. **Creating a Role and a User**
3. **Running a MapReduce Program**
4. **Running a Spark Program**
5. **Running a Hive Program**

Creating a Security Cluster and Logging In to Manager

Step 1 Create a security cluster. For details, see **Buying a Custom Cluster**. Enable **Kerberos Authentication**, configure **Password**, and confirm the password. This password is used to log in to Manager. Keep it secure.

Figure 3-1 Setting security cluster parameters



Step 2 Log in to the MRS console.

Step 3 In the navigation pane on the left, choose **Active Clusters** and click the target cluster name on the right to access the cluster details page.

Step 4 Click **Access Manager** on the right of **MRS Manager** to log in to Manager.

- If you have bound an EIP when creating the cluster, perform the following operations:

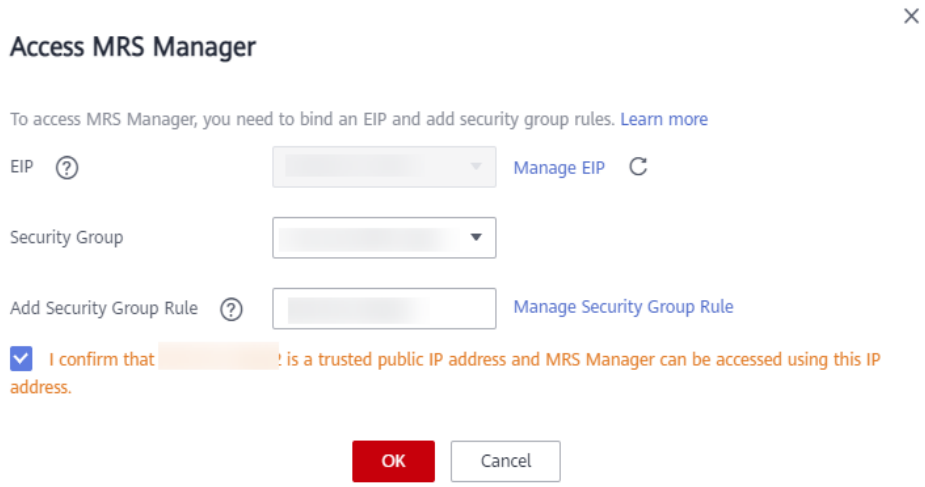
  a. Add a security group rule. By default, your public IP address used for accessing port 9022 is filled in the rule. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

  NOTE

  - It is normal that the automatically generated public IP address is different from your local IP address and no action is required.
  - If port 9022 is a Knox port, you need to enable the permission to access port 9022 of Knox for accessing Manager.

  b. Select **I confirm that xx.xx.xx.xx is a trusted public IP address and MRS Manager can be accessed using this IP address.**

  Figure 3-2 Accessing Manager



- If you have not bound an EIP when creating the cluster, perform the following operations:

  a. Select an EIP from the drop-down list or click **Manage EIP** to buy one.

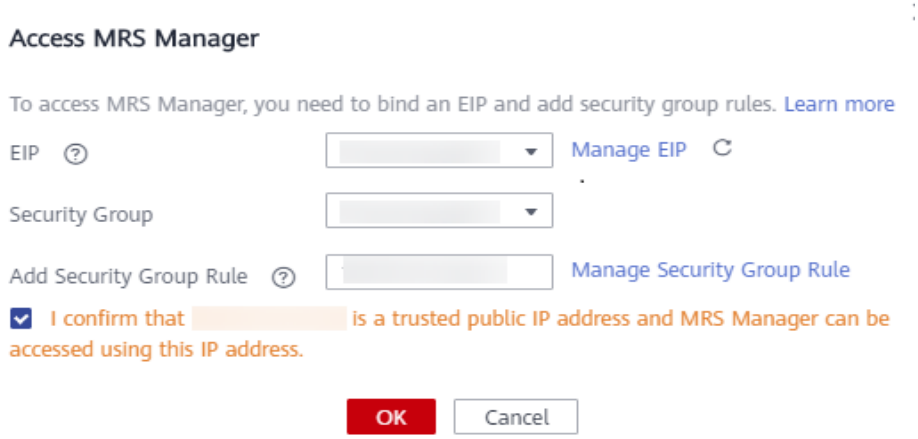
  b. Add a security group rule. By default, your public IP address used for accessing port 9022 is filled in the rule. If you want to view, modify, or delete a security group rule, click **Manage Security Group Rule**.

  NOTE

  - It is normal that the automatically generated public IP address is different from the local IP address and no action is required.
  - If port 9022 is a Knox port, you need to enable the permission of port 9022 to access Knox for accessing MRS Manager.

  c. Select **I confirm that xx.xx.xx.xx is a trusted public IP address and MRS Manager can be accessed using this IP address.**

Figure 3-3 Accessing Manager



Step 5 Click **OK**. The Manager login page is displayed. To assign other users the permission to access Manager, add the IP addresses as trusted ones by referring to **Accessing Manager**.

Step 6 Enter the default username **admin** and the password you set when creating the cluster, and click **Log In**.

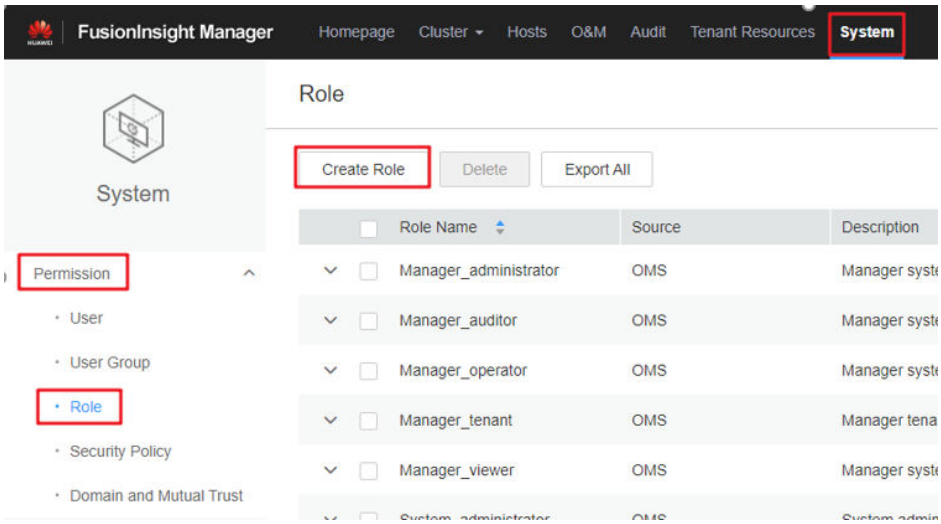
----End

Creating a Role and a User

For clusters with Kerberos authentication enabled, perform the following steps to create a user and assign permissions to the user to run programs.

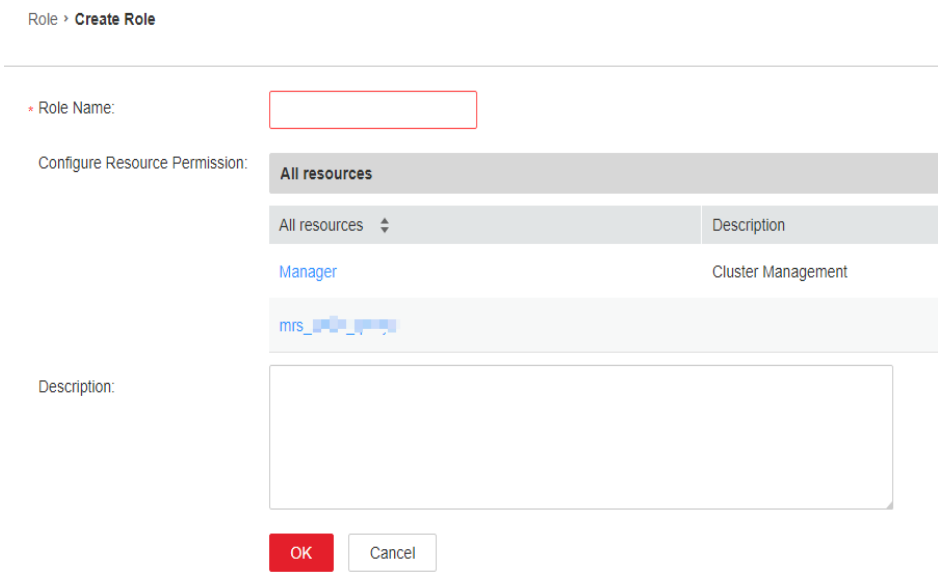
Step 1 On Manager, choose **System** > **Permission** > **Role**.

Figure 3-4 Role



Step 2 Click **Create Role**. For details, see **Creating a Role**.

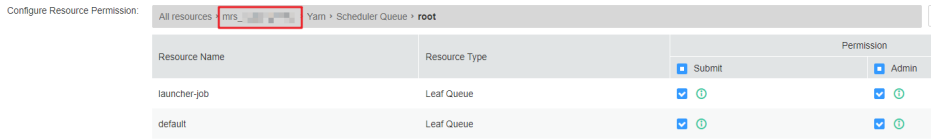
Figure 3-5 Creating a role



Specify the following information:

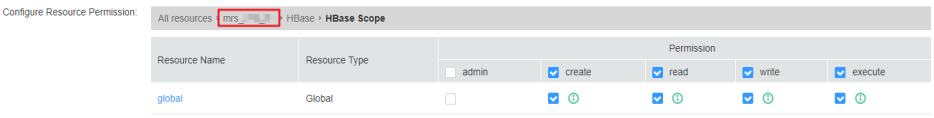
- Enter a role name, for example, **mrrole**.
- In **Configure Resource Permission**, select the cluster to be operated, choose **Yarn** > **Scheduler Queue** > **root**, and select **Submit** and **Admin** in the **Permission** column. After you finish configuration, do not click **OK** but click the name of the target cluster shown in the following figure and then configure other permissions.

  Figure 3-6 Configuring resource permissions for Yarn

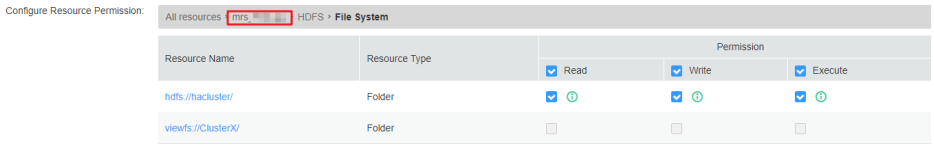
- Choose **HBase** > **HBase Scope**. Locate the row that contains **global**, and select **create**, **read**, **write**, and **execute** in the **Permission** column. After you finish configuration, do not click **OK** but click the name of the target cluster shown in the following figure and then configure other permissions.

  Figure 3-7 Configuring resource permissions for HBase

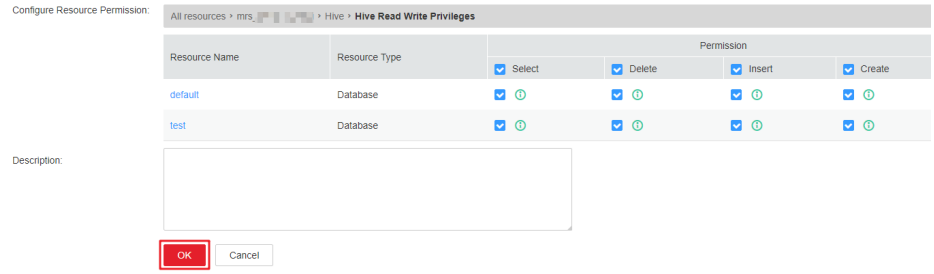
- Choose **HDFS** > **File System** > **hdfs://hacluster/** and select **Read**, **Write**, and **Execute** in the **Permission** column. After you finish configuration, do not click **OK** but click the name of the target cluster shown in the following figure and then configure other permissions.

Figure 3-8 Configuring resource permissions for HDFS



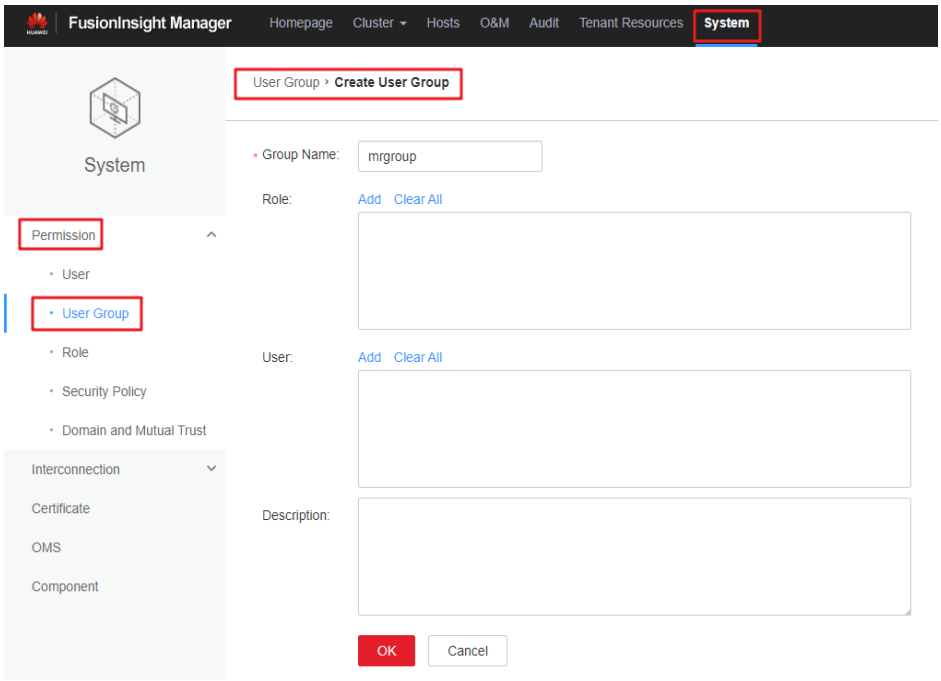
- Choose **Hive** > **Hive Read Write Privileges**, select **Select**, **Delete**, **Insert**, and **Create** in the **Permission** column, and click **OK**.

Figure 3-9 Configuring resource permissions for Hive



Step 3 Choose **System**. In the navigation pane on the left, choose **Permission** > **User Group** > **Create User Group** to create a user group for the sample project, for example, **mrgroup**. For details, see **Creating a User Group**.

Figure 3-10 Creating a user group



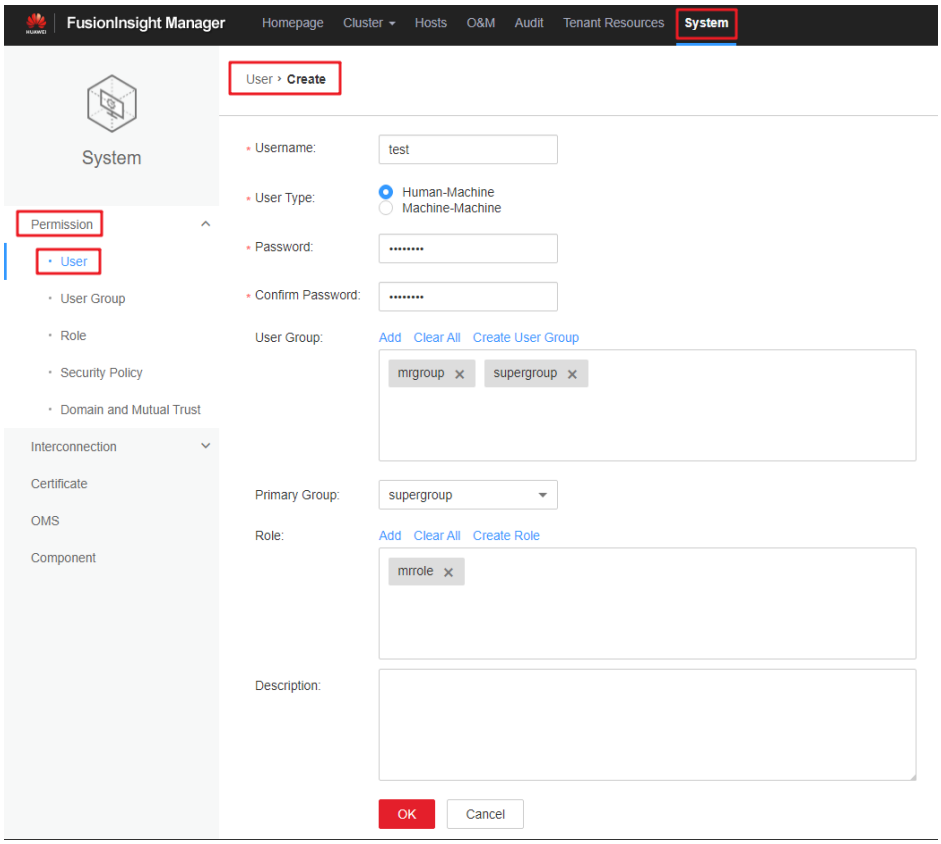
Step 4 Choose **System**. In the navigation pane on the left, choose **Permission** > **User** > **Create** to create a user for the sample project. For details, see **Creating a User**.

- Enter a username, for example, **test**. If you want to run a Hive program, enter **hiveuser** in **Username**.
- Set **User Type** to **Human-Machine**.

- Enter a password. This password will be used when you run the program.
- In **User Group**, add **mrgroup** and **supergroup**.
- Set **Primary Group** to **supergroup** and bind the **mrrole** role to obtain the permission.

  Click **OK**.

Figure 3-11 Creating a user



Step 5 Choose **System**. In the navigation pane on the left, choose **Permission** > **User**, locate the row where user **test** locates, and select **Download Authentication Credential** from the **More** drop-down list. Save the downloaded package and decompress it to obtain the **keytab** and **krb5.conf** files.

Figure 3-12 Downloading the authentication credential

----End

Running a MapReduce Program

This section describes how to run a MapReduce program in security cluster mode.

Prerequisites

You have compiled the program and prepared data files, for example, **mapreduce-examples-1.0.jar**, **input_data1.txt**, and **input_data2.txt**. For details about MapReduce program development and data preparations, see **MapReduce Introduction**.

Procedure

Step 1 Use a remote login software (for example, MobaXterm) to log in to the master node of the security cluster using SSH (using the EIP).

Step 2 After the login is successful, run the following commands to create the **test** folder in the **/opt/Bigdata/client** directory and create the **conf** folder in the **test** directory:

```
cd /opt/Bigdata/client
mkdir test
cd test
mkdir conf
```

Step 3 Use an upload tool (for example, WinSCP) to copy **mapreduce-examples-1.0.jar**, **input_data1.txt**, and **input_data2.txt** to the **test** directory, and copy the **keytab** and **krb5.conf** files obtained in **Step 5** in **Creating Roles and Users** to the **conf** directory.

Step 4 Run the following commands to configure environment variables and authenticate the created user, for example, **test**:

```
cd /opt/Bigdata/client
source bigdata_env
export YARN_USER_CLASSPATH=/opt/Bigdata/client/test/conf/
kinit test
```

Enter the password as prompted. If no error message is displayed (you need to change the password as prompted upon the first login), Kerberos authentication is complete.

Step 5 Run the following commands to import data to the HDFS:

```
cd test
hdfs dfs -mkdir /tmp/input
hdfs dfs -put input_data* /tmp/input
```

Step 6 Run the following commands to run the program:

```
yarn jar mapreduce-examples-1.0.jar com.huawei.bigdata.mapreduce.examples.FemaleInfoCollector /tmp/input /tmp/mapreduce_output
```

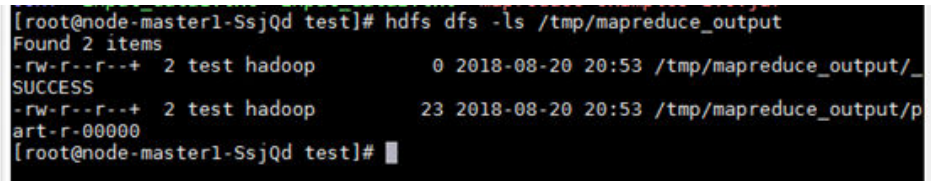
In the preceding commands:

/tmp/input indicates the input path in the HDFS.

/tmp/mapreduce_output indicates the output path in the HDFS. This directory must not exist. Otherwise, an error will be reported.

Step 7 After the program is executed successfully, run the **hdfs dfs -ls /tmp/mapreduce_output** command. The following command output is displayed.

Figure 3-13 Program running result



----End

Running a Spark Program

This section describes how to run a Spark program in security cluster mode.

Prerequisites

You have compiled the program and prepared data files, for example, **FemaleInfoCollection.jar**, **input_data1.txt**, and **input_data2.txt**. For details about Spark program development and data preparations, see **Spark Application Development Overview**.

Procedure

Step 1 Use a remote login software (for example, MobaXterm) to log in to the master node of the security cluster using SSH (using the EIP).

Step 2 After the login is successful, run the following commands to create the **test** folder in the **/opt/Bigdata/client** directory and create the **conf** folder in the **test** directory:

```
cd /opt/Bigdata/client
mkdir test
cd test
mkdir conf
```

Step 3 Use an upload tool (for example, WinSCP) to copy **FemaleInfoCollection.jar**, **input_data1.txt**, and **input_data2.txt** to the **test** directory, and copy the **keytab** and **krb5.conf** files obtained in **Step 5** in section **Creating Roles and Users** to the **conf** directory.

Step 4 Run the following commands to configure environment variables and authenticate the created user, for example, **test**:

```
cd /opt/Bigdata/client
source bigdata_env
export YARN_USER_CLASSPATH=/opt/Bigdata/client/test/conf/
kinit test
```

Enter the password as prompted. If no error message is displayed, Kerberos authentication is complete.

Step 5 Run the following commands to import data to the HDFS:

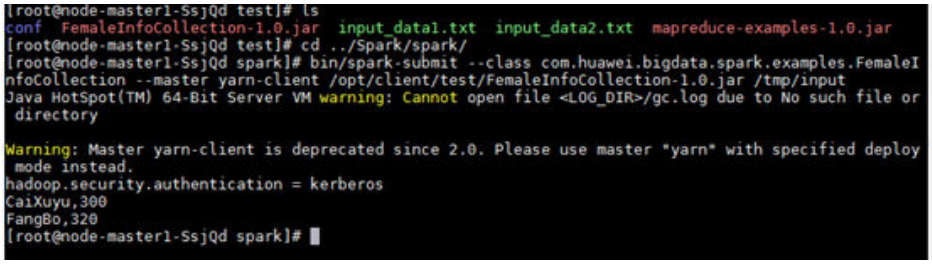
```
cd test
hdfs dfs -mkdir /tmp/input
hdfs dfs -put input_data* /tmp/input
```

Step 6 Run the following commands to run the program:

```
cd /opt/Bigdata/client/Spark/spark
bin/spark-submit --class com.huawei.bigdata.spark.examples.FemaleInfoCollection --master yarn-client /opt/
Bigdata/client/test/FemaleInfoCollection-1.0.jar /tmp/input
```

Step 7 After the program is run successfully, the following information is displayed.

Figure 3-14 Program running result



----End

Running a Hive Program

This section describes how to run a Hive program in security cluster mode.

Prerequisites

You have compiled the program and prepared data files, for example, **hive-examples-1.0.jar**, **input_data1.txt**, and **input_data2.txt**. For details about Hive program development and data preparations, see **Hive Application Development Overview**.

Procedure

Step 1 Use a remote login software (for example, MobaXterm) to log in to the master node of the security cluster using SSH (using the EIP).

Step 2 After the login is successful, run the following commands to create the **test** folder in the **/opt/Bigdata/client** directory and create the **conf** folder in the **test** directory:

```
cd /opt/Bigdata/client
mkdir test
cd test
mkdir conf
```

Step 3 Use an upload tool (for example, WinSCP) to copy **FemaleInfoCollection.jar**, **input_data1.txt**, and **input_data2.txt** to the **test** directory, and copy the **keytab** and **krb5.conf** files obtained in **Step 5** in section **Creating Roles and Users** to the **conf** directory.

Step 4 Run the following commands to configure environment variables and authenticate the created user, for example, **test**:

```
cd /opt/Bigdata/client
source bigdata_env
export YARN_USER_CLASSPATH=/opt/Bigdata/client/test/conf/
kinit test
```

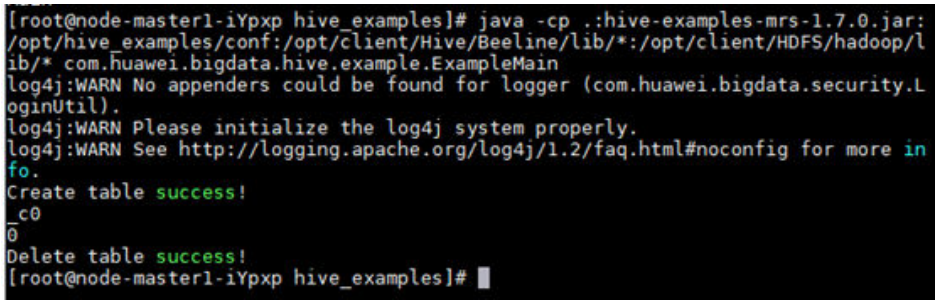
Enter the password as prompted. If no error message is displayed, Kerberos authentication is complete.

Step 5 Run the following command to run the program:

```
chmod +x /opt/hive_examples -R  cd /opt/hive_examples  java -cp .:hive-examples-1.0.jar:/opt/hive_examples/conf:/opt/Bigdata/client/Hive/Beeline/lib/*:/opt/Bigdata/client/HDFS/hadoop/lib/* com.huawei.bigdata.hive.example.ExampleMain
```

Step 6 After the program is run successfully, the following information is displayed.

Figure 3-15 Program running result



----End

4 Using Hadoop from Scratch

- MRS provides Hadoop-based high-performance big data components, such as Spark, HBase, Kafka, and Storm.
- This section describes how to use Hadoop to submit wordcount jobs through the GUI and cluster nodes. A wordcount job is the most classic Hadoop job that counts words in massive amounts of text.
- Purchase a cluster; prepare the Hadoop sample program and data files; upload data to OBS; create a job; and view job execution results.

  You can get started by reading the following steps:

  a. **Buy an MRS cluster.**
  b. **Configure software.**
  c. **Configure hardware.**
  d. **Set advanced options.**
  e. **Confirm the configuration.**
  f. **Prepare the Hadoop sample program and data files.**
  g. **Upload data to OBS.**
  h. **Submit a job on the GUI.**
  i. **Submit a job through a cluster node.**
  j. **Query job execution results.**

Procedure

Step 1 Buy an MRS cluster.

1. Log in to the Huawei Cloud console.
2. Choose **Service List** > **Analytics** > **MapReduce Service**.
3. On the **Active Clusters** page that is displayed, click **Buy Cluster**.
4. Click the **Custom Config** tab.

Step 2 Configure software.

1. **Region**: Select a region as required.
2. **Billing Mode**: Select **Pay-per-use**.
3. **Cluster Name**: Enter **mrs_demo** or specify a name according to naming rules.

4. **Cluster Version**: Select **MRS 3.1.0**.
5. **Cluster Type**: Select **Analysis Cluster**.
6. Select all analysis cluster components.
7. Click **Next**.

Step 3 Configure hardware.

1. **AZ**: Select **AZ2**.
2. **Enterprise Project**: Select **default**.
3. **VPC** and **Subnet**: Retain their default values or click **View VPC** and **View Subnet** to create ones.
4. **Security Group**: Use the default value **Auto create**.
5. **EIP**: **Bind later** is selected by default.
6. **Cluster Node**: Retain the default values. Do not add task nodes.
7. Click **Next**.

Step 4 Set advanced options.

1. **Kerberos Authentication**: **Disabled**
2. **Username**: **admin** is used by default.
3. **Password** and **Confirm Password**: Set them to the password of the FusionInsight Manager administrator.
4. **Login Mode**: Select **Password**. Enter a password and confirm the password for user **root**.
5. **Host Name Prefix**: Retain the default value.
6. Select **Advanced Settings** and set **Agency** to **MRS_ECS_DEFAULT_AGENCY**.
7. Click **Next**.

Step 5 Confirm the configuration.

1. **Configure**: Confirm the parameters configured in the **Configure Software**, **Configure Hardware**, and **Set Advanced Options** areas.
2. **Secure Communications**: Select **Enable**.
3. Click **Buy Now**. The page is displayed showing that the task has been submitted.
4. Click **Back to Cluster List**. You can view the status of the cluster on the **Active Clusters** page. Wait for the cluster creation to complete. The initial status of the cluster is **Starting**. After the cluster has been created, the cluster status becomes **Running**.

Step 6 Prepare the Hadoop sample program and data files.

1. Prepare the wordcount program.

   Download the **Hadoop sample program** (including wordcount). **hadoop-3.3.1.tar.gz** is used as an example. Use the actual program version provided in the link. For example, choose **hadoop-3.3.1**. On the page that is displayed, click **hadoop-3.3.1.tar.gz** to download it. Then, decompress it to obtain **hadoop-mapreduce-examples-3.3.1.jar** (the Hadoop sample program) from **hadoop-3.3.1\share\hadoop\mapreduce**.

2. Prepare data files.

There is no requirement on the format of data files. Prepare two **.txt** files. In this example, files **wordcount1.txt** and **wordcount2.txt** are used.

Step 7 Upload data to OBS.

1. Log in to the OBS console and choose **Parallel File Systems**. On the **Parallel File Systems** page, click **Create Parallel File System**. On the **Create Parallel File System** page that is displayed, configure parameters to create a file system named **mrs-word01**.
2. Click the name of the **mrs-word01** file system. In the navigation pane on the left, choose **Files**. On the page that is displayed, click **Create Folder** to create the **program** and **input** folders.
3. Go to the **program** folder and upload the Hadoop sample program downloaded in **Step 6**.
4. Go to the **input** folder and upload the **wordcount1.txt** and **wordcount2.txt** data files prepared in **Step 6**.
5. To submit a job on the GUI, go to **Step 8**.

   To submit a job through a cluster node, go to **Step 9**.

Step 8 Submit a job on the GUI.

1. In the navigation pane of the MRS console, choose **Clusters** > **Active Clusters**. On the **Active Clusters** page, click the **mrs_demo** cluster.
2. On the cluster information page, click the **Jobs** tab then **Create** to create a job. To submit a job through a cluster node, go to **Step 9**.
3. **Type**: **MapReduce**
4. **Job Name**: Enter **wordcount**.
5. **Program Path**: Click **OBS** and select the Hadoop sample program uploaded in **Step 7**.
6. **Parameters**: Enter **wordcount obs://mrs-word01/input/ obs://mrs-word01/output/**. **output** indicates the output path. Enter a directory that does not exist.
7. **Service Parameters**: Leave it blank.
8. Click **OK** to submit the job. After a job is submitted, it is in the **Accepted** state by default. You do not need to manually execute the job.
9. Go to the **Jobs** tab page, view the job status and logs, and go to **Step 10** to view the job execution result.

Step 9 Submit a job through a cluster node.

1. Log in to the MRS console and click the cluster named **mrs_demo** to go to its details page.
2. Click the **Nodes** tab. On this tab page, click the name of a master node to go to the ECS management console.
3. Click **Remote Login** in the upper right corner of the page.
4. Enter the username and password of the master node as prompted. The username is **root** and the password is the one configured during cluster creation.
5. Run the **source /opt/Bigdata/client/bigdata_env** command to configure environment variables.

6. If Kerberos authentication has been enabled, run the **kinit** *MRS cluster user* command, for example, **kinit admin**, to authenticate the current cluster user. Skip this step if Kerberos authentication is not enabled.
7. Run the following command to copy the sample program in the OBS bucket to the master node in the cluster:

   hadoop fs -Dfs.obs.access.key=AK -Dfs.obs.secret.key=SK -copyToLocal source_path.jar target_path.jar Example: **hadoop fs -Dfs.obs.access.key=XXXX -Dfs.obs.secret.key=XXXX -copyToLocal "obs://mrs-word01/program/hadoop-mapreduce-examples-XXX.jar" "/home/omm/hadoop-mapreduce-examples-XXX.jar"** To obtain the AK/SK pair for logging in to the OBS console, hover your cursor over the username in the upper right corner of the management console, and choose **My Credentials** > **Access Keys**, or click **Create Access Key** to create one.
8. Run the following command to submit a wordcount job. To read data from or write data to OBS, add AK/SK parameters. **source /opt/Bigdata/client/bigdata_env;hadoop jar execute_jar wordcount input_path output_path** Example: **source /opt/Bigdata/client/bigdata_env;hadoop jar /home/omm/hadoop-mapreduce-examples-XXX.jar wordcount -Dfs.obs.access.key=XXXX -Dfs.obs.secret.key=XXXX "obs://mrs-word01/input/*" "obs://mrs-word01/output/"** In this command, **input_path** indicates a path for storing job input files on OBS. **output_path** indicates a path for storing job output files on OBS and needs to be set to a directory that does not exist

Step 10 Query job execution results.

1. Log in to OBS console and click the name of the **mrs-word01** parallel file system.
2. On the page that is displayed, choose **Files** in the navigation pane on the left. Go to the output path in the **mrs-word01** bucket specified during job submission, and view the job output file. You need to download the file to the local host and open it in a **.txt** format.

----End

5 Using Kafka from Scratch

MRS provides Hadoop-based high-performance big data components, such as Spark, HBase, Kafka, and Storm.

This section uses a cluster with Kerberos authentication disabled as an example to describe how to generate and consume messages in a Kafka topic.

You can get started by reading the following steps:

1. **Purchasing a Cluster**
2. **Installing the Kafka Client**
3. **Logging In to a Master Node Using VNC**
4. **Creating a Topic Using the Kafka Client**
5. **Managing Messages in Kafka Topics**

Video Tutorial

This video uses an MRS 3.1.0 cluster (with Kerberos authentication disabled) as an example to describe how to use a Kafka client to create, query, and delete a topic. For details about how to create a topic, see **Creating a Topic Using the Kafka Client**.

NOTE

The UI may vary depending on the version. The video tutorial is for reference only.

Purchasing a Cluster

Step 1 Buy an MRS cluster.

1. Log in to the Huawei Cloud console.
2. Choose **Service List** > **Analytics** > **MapReduce Service**.
3. On the **Active Clusters** page that is displayed, click **Buy Cluster**.
4. Click the **Custom Config** tab.

Step 2 Configure software.

1. **Region**: Select a region as required.
2. **Billing Mode**: Select **Pay-per-use**.
3. **Cluster Name**: Enter **mrs_demo** or specify a name according to naming rules.
4. **Cluster Version**: Select **MRS 3.1.0**.
5. **Cluster Type**: Select **Streaming cluster**.
6. Select all streaming cluster components.
7. Click **Next**.

Step 3 Configure hardware.

1. **AZ**: Select **AZ2**.
2. **Enterprise Project**: Select **default**.
3. **VPC** and **Subnet**: Retain their default values or click **View VPC** and **View Subnet** to create ones.
4. **Security Group**: Use the default value **Auto create**.
5. **EIP**: **Bind later** is selected by default.
6. **Cluster Node**: Retain the default values. Do not add task nodes.
7. Click **Next**.

Step 4 Set advanced options.

1. **Kerberos Authentication**: **Disabled**
2. **Username**: **admin** is used by default.
3. **Password** and **Confirm Password**: Set them to the password of the FusionInsight Manager administrator.
4. **Login Mode**: Select **Password**. Enter a password and confirm the password for user **root**.
5. **Host Name Prefix**: Retain the default value.
6. Select **Advanced Settings** and set **Agency** to **MRS_ECS_DEFAULT_AGENCY**.
7. Click **Next**.

Step 5 Confirm the configuration.

1. **Configure**: Confirm the parameters configured in the **Configure Software**, **Configure Hardware**, and **Set Advanced Options** areas.
2. **Secure Communications**: Select **Enable**.
3. Click **Buy Now**. The page is displayed showing that the task has been submitted.
4. Click **Back to Cluster List**. You can view the status of the cluster on the **Active Clusters** page. Wait for the cluster creation to complete. The initial status of the cluster is **Starting**. After the cluster has been created, the cluster status becomes **Running**.

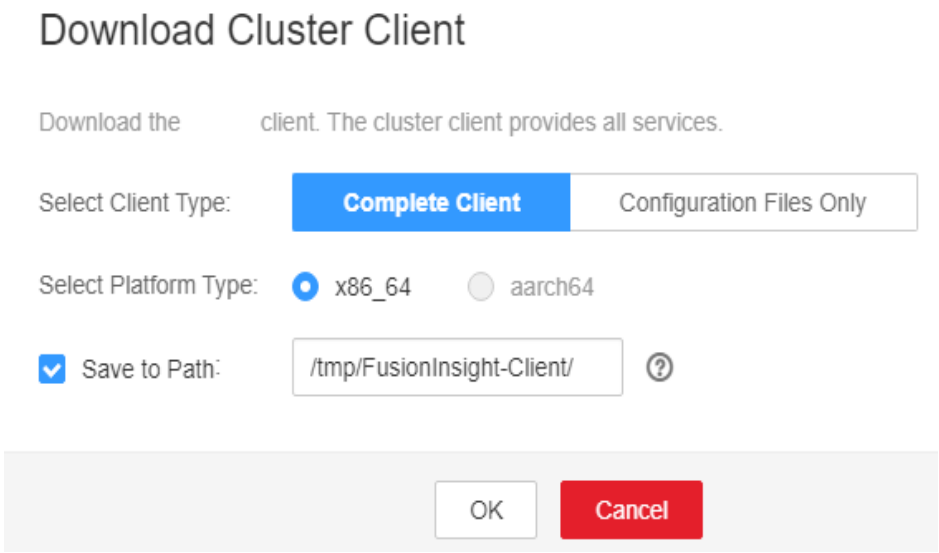
----End

Installing the Kafka Client

Step 1 Choose **Clusters** > **Active Clusters**. On the **Active Clusters** page, click the cluster named **mrs_demo** to go to its details page.

Step 2 Click **Access Manager** next to **MRS Manager**. On the page that is displayed, configure the EIP information and click **OK**. Enter the username and password to access FusionInsight Manager.

Step 3 Choose **Cluster** > **Services** > **HBase**. On the page displayed, choose **More** > **Download Client**. In the **Download Cluster Client** dialog box, select **Complete Client** for **Select Client Type**, select a platform type, select **Save to Path**, and click **OK**. The Kafka client software package, for example, **FusionInsight_Cluster_1_Kafka_Client.tar**, is downloaded.



Step 4 Log in to the active node as user **root**.

Step 5 Go to the directory where the software package is stored and run the following commands to decompress and verify the software package, and decompress the obtained installation file:

cd /tmp/FusionInsight-Client

tar -xvf FusionInsight_Cluster_1_Kafka_Client.tar

sha256sum -c FusionInsight_Cluster_1_Kafka_ClientConfig.tar.sha256

tar -xvf FusionInsight_Cluster_1_Kafka_ClientConfig.tar

Step 6 Go to the directory where the installation package is stored, and run the following command to install the client to a specified directory (absolute path), for example, **/opt/hadoopclient**:

cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_Kafka_ClientConfig

Run the **./install.sh /opt/hadoopclient** command and wait until the client installation is complete.

Step 7 Check whether the client is installed.

cd /opt/hadoopclient

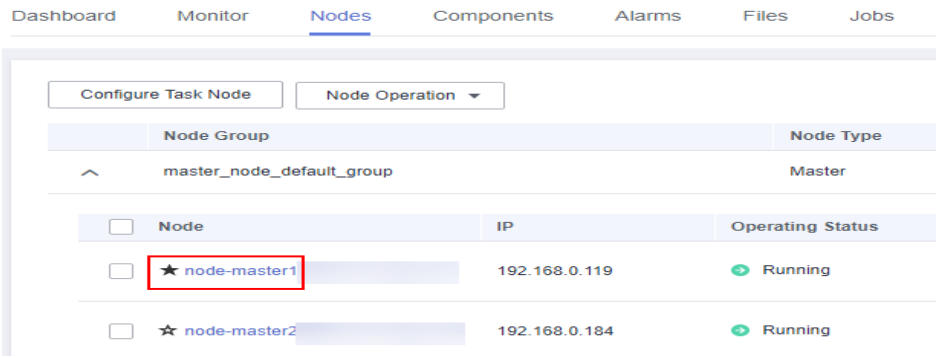
source bigdata_env

Run the **klist** command to query and confirm authentication details. If the command is executed, the Kafka client is installed.

----End

Logging In to a Master Node Using VNC

Step 1 Choose **Clusters** > **Active Clusters**. On the **Active Clusters** page that is displayed, click the cluster named **mrs_demo**. On the cluster details page that is displayed, click the **Nodes** tab. On this tab page, locate the node whose type is **Master1** and click the node name to go to the ECS details page.



Step 2 Click **Remote Login** in the upper right corner of the page to remotely log in to the master node. Log in using the username **root** and the password configured during cluster purchase.

----End

Creating a Topic Using the Kafka Client

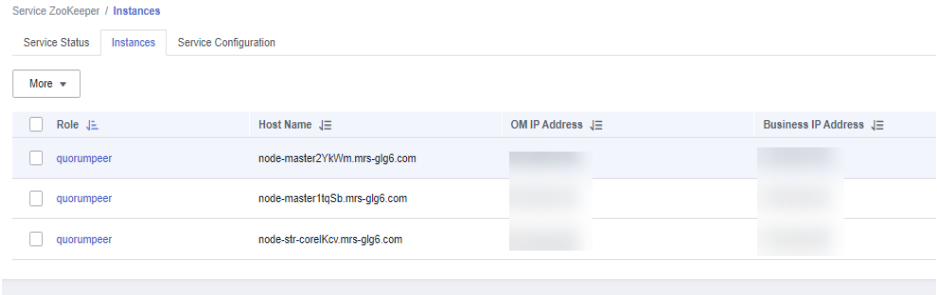
Step 1 Configure environment variables. For example, if the Kafka client installation directory is **/opt/hadoopclient**, run the following command:

source /opt/hadoopclient/bigdata_env

Step 2 Choose **Clusters** > **Active Clusters**. On the **Active Clusters** page, click the cluster named **mrs_demo** to go to the **Dashboard** tab page. On this page, click **Synchronize** next to **IAM User Sync**.

Step 3 After the synchronization is complete, click the **Components** tab. On this tab page, select **ZooKeeper**. On the page that is displayed, click the **Instances** tab. Record the IP address of any ZooKeeper instance, for example, **192.168.7.35**.

Figure 5-1 IP addresses of ZooKeeper role instances



Step 4 Run the following command to create a Kafka topic:

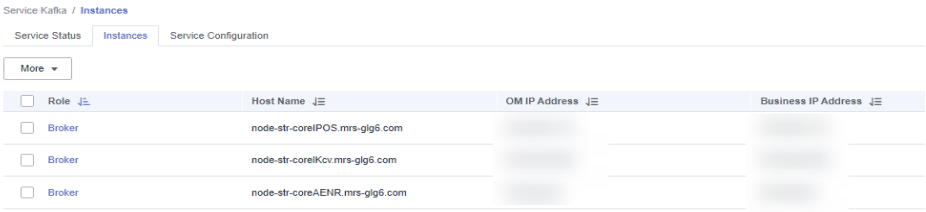
kafka-topics.sh --create --zookeeper <*IP address of the node where the ZooKeeper instance resides***:2181/kafka> --partitions 2 --replication-factor 2 --topic <***Topic name***>**

----End

Managing Messages in Kafka Topics

Step 1 Click the **Components** tab. On this tab page, select **Kafka**. On the page that is displayed, click the **Instances** tab. On the **Instances** tab page, view the IP addresses of Kafka instances. Record the IP address of any Kafka instance, for example, **192.168.7.15**.

Figure 5-2 IP addresses of Kafka role instances



Step 2 Log in to the master node and run the following command to generate messages in a topic test:

kafka-console-producer.sh --broker-list <*IP address of the node where the Kafka instance resides***:9092> --topic <***Topic name***> --producer.config /opt/hadoopclient/Kafka/kafka/config/producer.properties**

Enter the specified content as the messages generated by the producer and press **Enter** to send the messages. To stop generating messages, press **Ctrl+C** to exit.

Step 3 Consume messages in the topic test.

kafka-console-consumer.sh --topic <*Topic name***> --bootstrap-server <***IP address of the node where the Kafka instance resides***:9092> --consumer.config /opt/hadoopclient/Kafka/kafka/config/consumer.properties**

----End

6 Using HBase from Scratch

MRS provides Hadoop-based high-performance big data components, such as Spark, HBase, and Kafka.

This section uses a cluster with Kerberos authentication disabled as an example to describe how to log in to the HBase client, create a table, insert data into the table, and modify the table.

You can get started by reading the following topics:

1. **Preparing an MRS Cluster**
2. **Installing the HBase Client**
3. **Creating a Table Using the HBase Client**

Video Tutorial

This video uses an MRS 3.1.0 cluster (with Kerberos authentication disabled) as an example to describe how to use an HBase client to create a table, insert data into the table, and modify table data. For details about how to use an HBase client to create a table, see **Using an HBase Client**.

NOTE

The UI may vary depending on the version. The video tutorial is for reference only.

Preparing an MRS Cluster

Step 1 Purchase an MRS cluster.

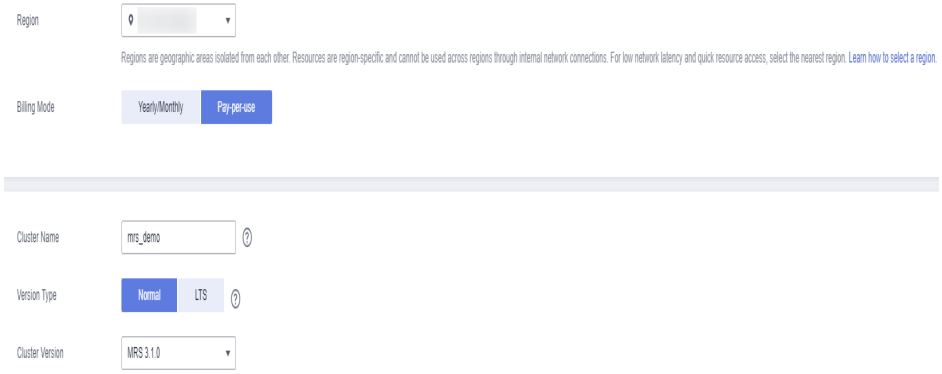
1. Go to the **Buy Cluster** page.
2. Click the **Custom Config** tab.

Step 2 Set the following parameters and click **Next**.

- **Region**: Select a region as required.
- **Billing Mode**: Select **Pay-per-use**.
- **Cluster Name**: Enter **mrs_demo** or specify a name according to naming rules.
- **Version Type**: Select **Normal**.

- **Cluster Version**: Select **MRS 3.1.0**.

Figure 6-1 Configure Software page



- **Cluster Type**: Select **Analysis Cluster** and select HBase.

Figure 6-2 Selecting the cluster type and components

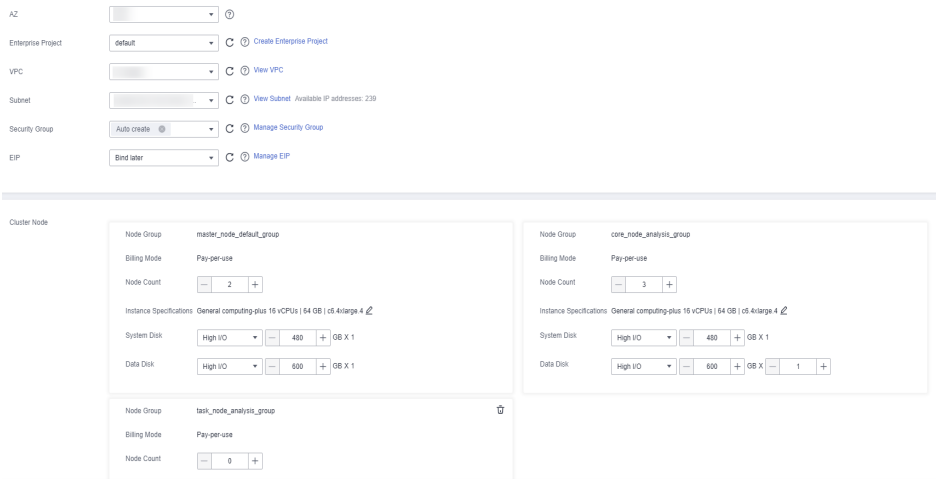
Step 3 On the **Configure Hardware** page, set the parameters by referring to **Table 6-1**, and click **Next**.

Table 6-1 MRS cluster hardware configuration

Parameter	Example Value
AZ	**AZ2**
Enterprise Project	**default**
VPC	Retain the default value. You can also click **View VPC** to create a VPC.

Parameter	Example Value
EIP	You can select an existing EIP from the drop-down list. If no EIP is available in the drop-down list, click **Manage EIP** to access the **EIPs** page to create one.

Figure 6-3 Hardware configurations



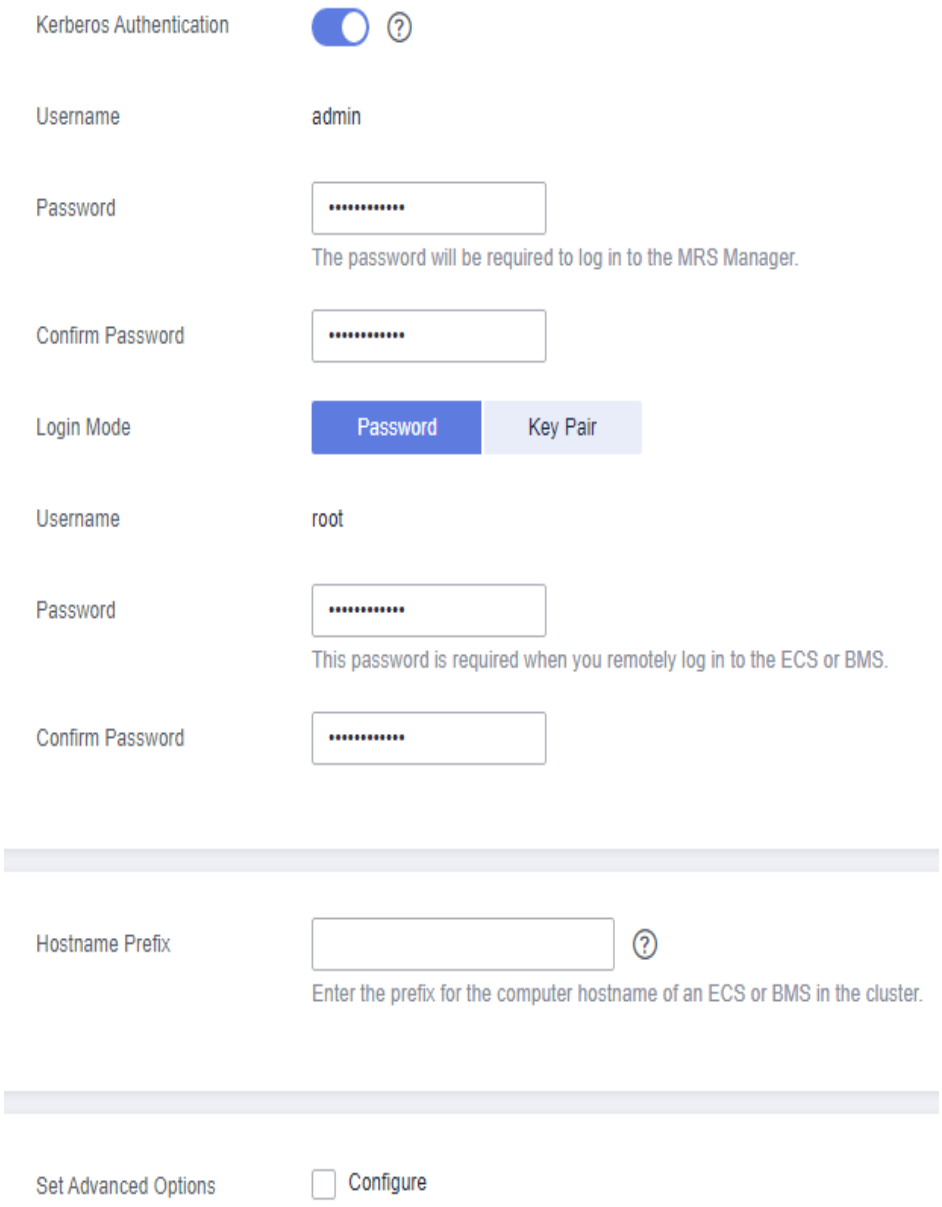
Step 4 Configure advanced options.

1. On the **Set Advanced Options** page, configure the parameters according to **Table 6-2**, and click **Next**.

   Table 6-2 MRS cluster advanced options

Parameter	Example Value
Kerberos Authentication	Disable this function.
Password	**Test@!123456**
Confirm Password	**Test@!123456**
Login Mode	**Password**
Password	**Test@#123456**
Confirm Password	**Test@#123456**

Figure 6-4 Set Advanced Options



2. 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.
3. Select **Enable** for **Secure Communications**. Click **Buy Now**. A page is displayed, indicating that the task is submitted successfully.
4. Click **Back to Cluster List**. You can view the status of the cluster on the **Active Clusters** page.
5. Wait for the cluster creation to complete. The initial status of the cluster is **Starting**. After the cluster has been created successfully, the cluster status becomes **Running**.

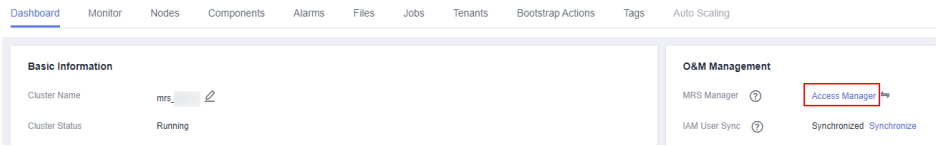
----End

Installing the HBase Client

Step 1 Choose **Clusters** > **Active Clusters** and click **mrs_demo**. The cluster information page is displayed.

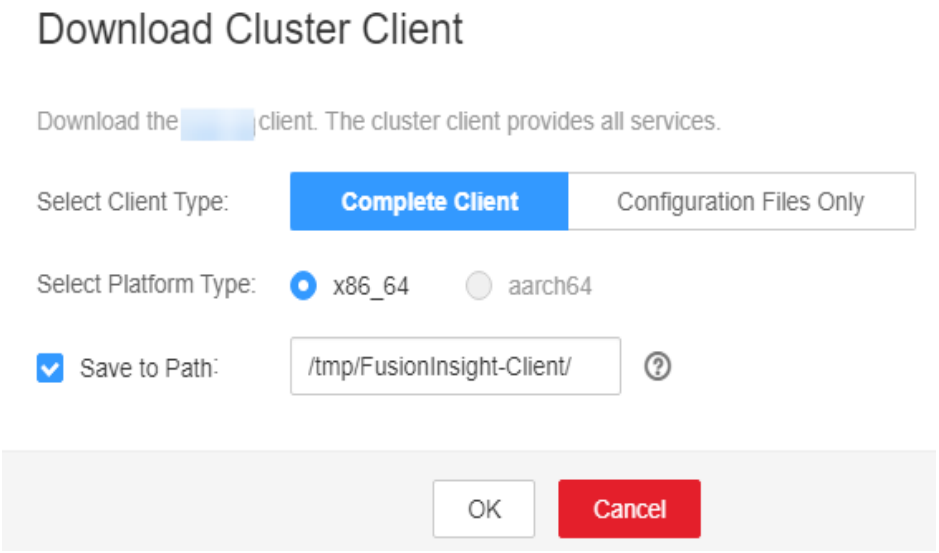
Step 2 Click **Access Manager** next to **MRS Manager**. On the displayed page, configure the EIP information and click **OK**. Enter the username and password to access FusionInsight Manager.

Figure 6-5 Logging in to FusionInsight Manager from the management console



Step 3 Choose **Cluster** > **Services** > **HBase**, and select **Download Client** from the **More** drop-down list. Select **Complete Client**, the corresponding platform type, and **Save to path**, and click OK.

Figure 6-6 Downloading the cluster client



Step 4 Log in to the active management node as user **root**.

NOTE

To identify the active and standby management nodes, see **Determining Active and Standby Management Nodes of Manager**

Step 5 Go to the directory where the installation package is stored and run the following commands to decompress and verify the installation package, and decompress the obtained installation file:

cd /tmp/FusionInsight-Client

tar -xvf FusionInsight_Cluster_1_HBase_Client.tar

sha256sum -c FusionInsight_Cluster_1_HBase_ClientConfig.tar.sha256

tar -xvf FusionInsight_Cluster_1_HBase_ClientConfig.tar

Step 6 Go to the directory where the installation package is stored, and run the following command to install the client to a specified directory (an absolute path), for example, **/opt/hbaseclient**:

cd /tmp/FusionInsight-Client/FusionInsight_Cluster_1_HBase_ClientConfig

Run the **./install.sh /opt/hbaseclient** command and wait until the client installation is complete.

Step 7 Check whether the client is successfully installed.

cd /opt/hbaseclient

source bigdata_env

hbase shell

If the command is successfully executed, the HBase client is successfully installed.

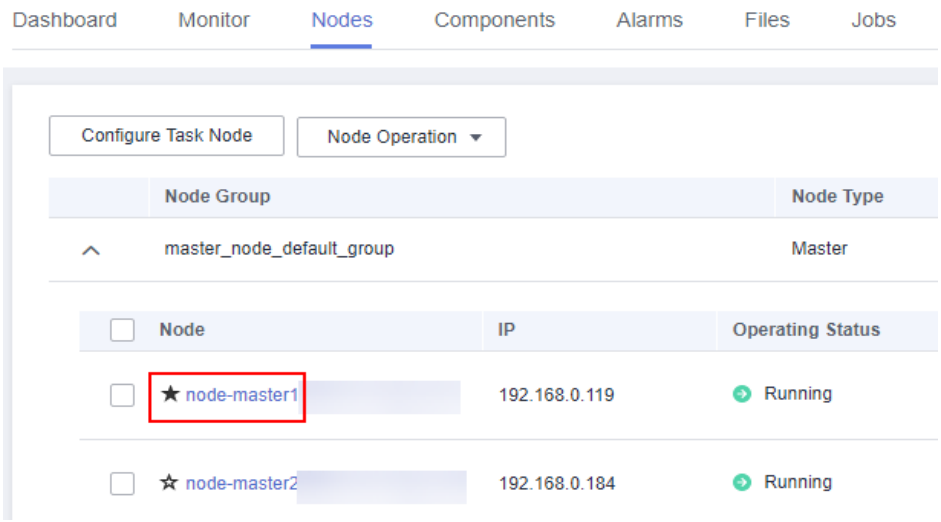
----End

Creating a Table Using the HBase Client

Step 1 Log in to the master node using VNC.

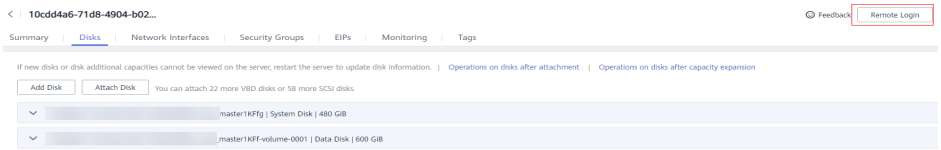
1. On the MRS console, choose **Clusters** > **Active Clusters**, and select **mrs_demo** from the cluster list. Click **Nodes**, and click the node whose name contains **master1** to access its ECS details page.

   Figure 6-7 Nodes tab page where the Master1 node is

2. Click **Remote Login** in the upper right corner of the page to log in to the master node as user **root**. The password is the one set when the cluster is purchased.

   Figure 6-8 Remotely logging in to the Master1 node

Step 2 Run the following command to go to the client directory:

cd /opt/hbaseclient

Step 3 Run the following command to configure environment variables:

source bigdata_env

NOTE

If Kerberos authentication is enabled for the cluster, run the following command to authenticate the current user. The current user must have the permission to create HBase tables.

For example:

kinit *hbaseuser*

Step 4 Run the following command to access the HBase shell CLI:

hbase shell

Step 5 Run the HBase client command to create the **user_info** table.

1. Create the **user_info** table and add related data.

   create '*user_info*',{**NAME** => 'i'}

   put '*user_info*','*12005000201*','***i:name***','*A*'

   put '*user_info*','*12005000201*','***i:gender***','*Male*'

   put '*user_info*','*12005000201*','***i:age***','*19*'

   put '*user_info*','*12005000201*','***i:address***','*City A*'

2. Add users' educational backgrounds and professional titles to the **user_info** table.

   put '*user_info*','*12005000201*','**i:degree**','*master*'

   put '*user_info*','*12005000201*','**i:pose**','*manager*'

3. Query user names and addresses by user ID.

   scan'*user_info*',
   {**STARTROW**=>'*12005000201*',**STOPROW**=>'*12005000201*',**COLUMNS**=>['**i:name**','**i:address**']}

   ```
   ROW                                   COLUMN
   +CELL
    12005000201                          column=i:address, timestamp=2021-10-30T10:21:42.196, value=City
   A
    12005000201                          column=i:name, timestamp=2021-10-30T10:21:18.594,
   value=A
   1 row(s)
   Took 0.0996 seconds
   ```

4. Query information by user name.

   scan'*user_info*',{**FILTER**=>"SingleColumnValueFilter('i','name',=,'binary:*A*')"}

   ```
   ROW                                   COLUMN
   +CELL
    12005000201                          column=i:address, timestamp=2021-10-30T10:21:42.196, value=City
   A
    12005000201                          column=i:age, timestamp=2021-10-30T10:21:30.777,
   value=19
    12005000201                          column=i:degree, timestamp=2021-10-30T10:21:53.284,
   value=master
    12005000201                          column=i:gender, timestamp=2021-10-30T10:21:18.711,
   value=Male
   ```



```
 12005000201                 column=i:name, timestamp=2021-10-30T10:21:18.594,
value=A
 12005000201                 column=i:pose, timestamp=2021-10-30T10:22:07.152,
value=manager
1 row(s)
Took 0.2158 seconds
```

5. Delete user data from the user information table.

   delete'*user_info*','*12005000201*','i'

6. Delete the user information table.

   disable *'user_info'*

   drop *'user_info'*

----End

7 Modifying MRS Configurations

After an MRS cluster is created, you can modify configuration parameters of services in the cluster on the MRS console or Manager.

This section uses the **hbase.log.maxbackupindex** parameter of the HBase service as an example to describe how to modify the MRS configuration parameters.

You can get started by reading the following topics:

1. **Modifying Service Parameters on the MRS Console**
2. **Modifying Service Parameters on FusionInsight Manager**

Video Tutorial

This video uses an MRS 3.1.0 cluster as an example to describe how to modify service parameters on the management console and FusionInsight Manager. For details, see **Modifying Cluster Service Configuration Parameters**.

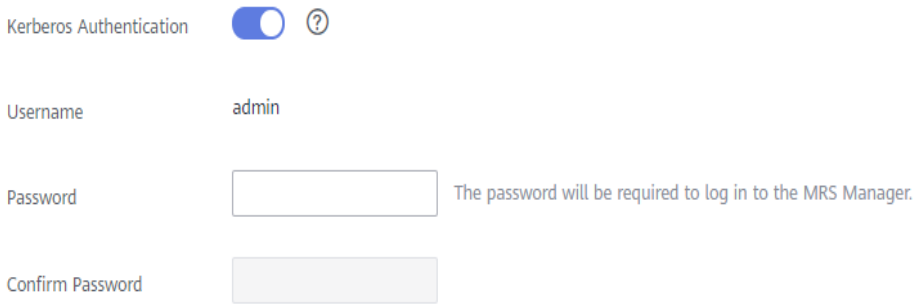
NOTE

The UI may vary depending on the version. The video tutorial is for reference only.

Modifying Service Parameters on the MRS Console

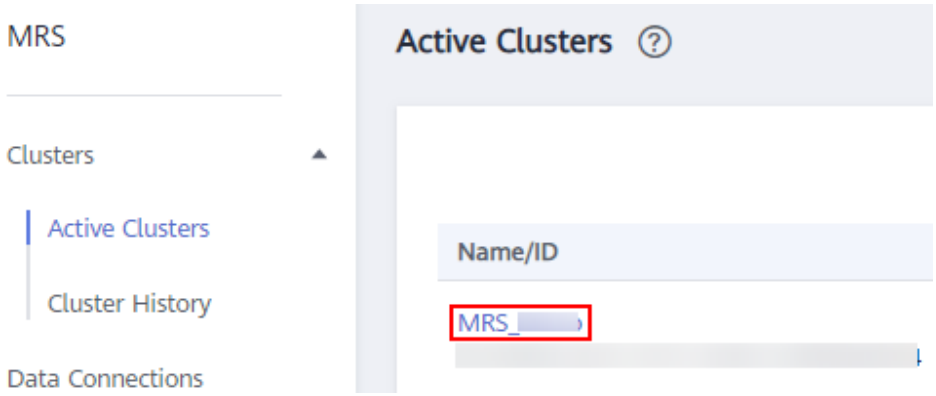
Step 1 Create a security cluster. For details, see **Purchasing a Custom Cluster**. Enable **Kerberos Authentication**, configure **Password**, and confirm the password. This password is used to log in to Manager. Keep it secure.

Figure 7-1 Setting security cluster parameters



Step 2 Log in to the MRS console. In the navigation pane on the left, choose **Clusters** > **Active Clusters** and click a cluster name.

Figure 7-2 Clicking a cluster name

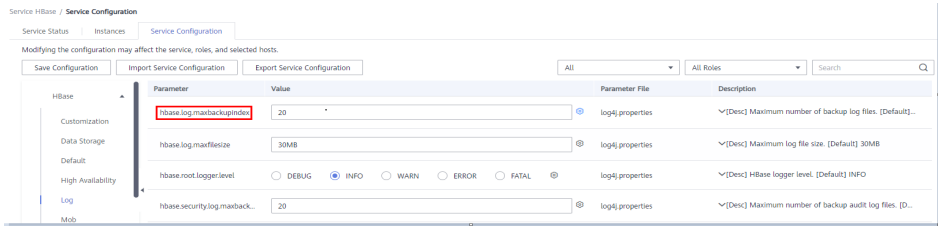


Step 3 Choose **Components** > **HBase**, click **Service Configuration**, and choose **All** in the upper right corner of the page.

Step 4 In the navigation tree on the left, choose **HBase** > **Log**.

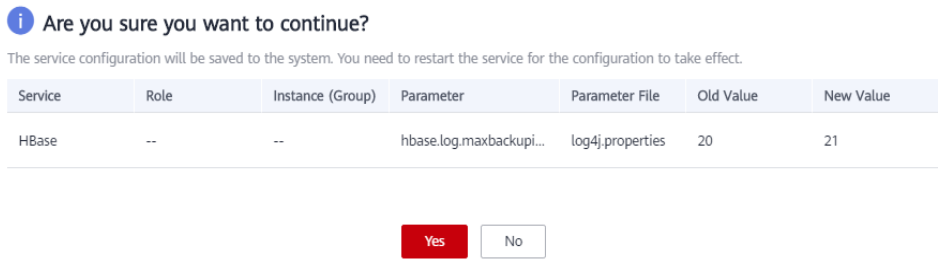
Step 5 Locate the **hbase.log.maxbackupindex** parameter and change its value based on service requirements.

Figure 7-3 Changing the parameter value



Step 6 Click **Save Configuration**. In the displayed dialog box, confirm the changed parameter value, and click **Yes**. Wait for the system to save and update the configuration, and click **Finish**.

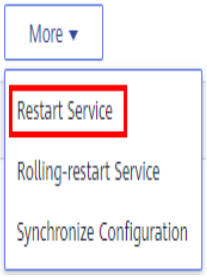
Figure 7-4 Confirming the modification



Step 7 Check the current service configuration status.

Click **Service Status** to view the current service configuration status. If the configuration of a service has expired, click **More** and select **Restart Service** to restart the service. In the displayed dialog box, click **Yes**. Then wait until the service is restarted.

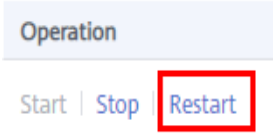
Figure 7-5 Restarting a service



Step 8 Check the service configuration status of related services.

Return to the **Components** page to check the configuration status of related services. If the configuration of a service has expired, click **Restart** in the **Operation** column of the service. In the displayed dialog box, click **Yes** to restart it.

Figure 7-6 Restarting a service



----End

Modifying Service Parameters on FusionInsight Manager

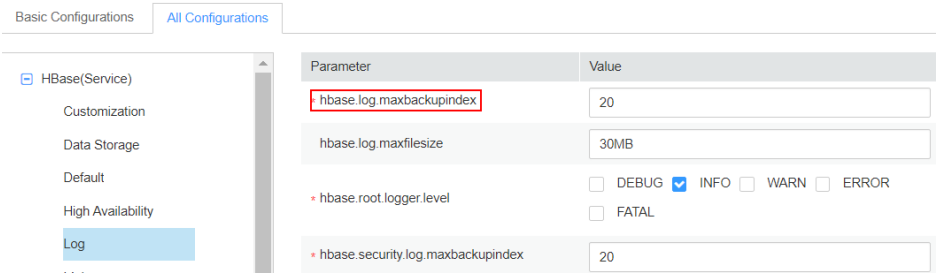
Step 1 Create a cluster and log in to FusionInsight Manager. For details, see **Creating a Security Cluster and Logging In to Manager**.

Step 2 Choose **Cluster** > **Services** > **HBase**, choose **Configurations**, and click **All Configurations**.

Step 3 Choose **HBase(Service)** > **Log**.

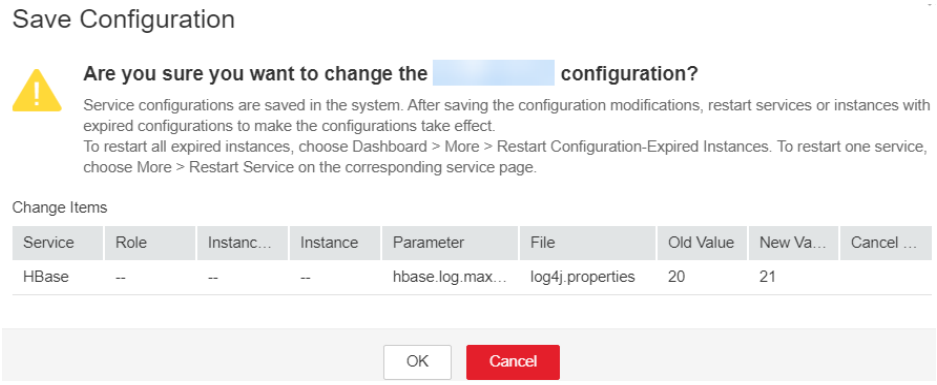
Step 4 Locate the **hbase.log.maxbackupindex** parameter and change its value based on service requirements.

Figure 7-7 Changing the parameter value



Step 5 Click **Save**. In the displayed dialog box, confirm the changed parameter value and click **OK**. Wait for the system to save and update the configuration, and click **Finish**.

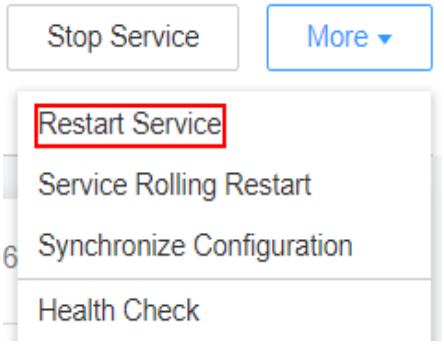
Figure 7-8 Confirming the modification



Step 6 Check the current service configuration status.

Click **Dashboard** to view the current service configuration status. If the configuration of a service has expired, click **More** and select **Restart Service**. Then enter the password and click **OK** to restart the service. Wait until the service is restarted.

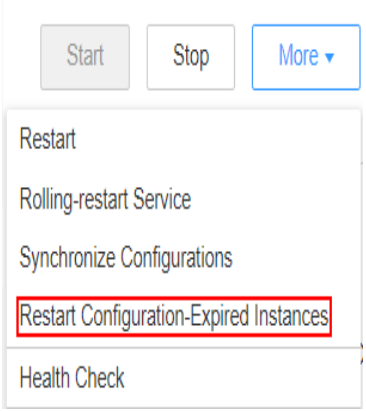
Figure 7-9 Restarting a service



Step 7 Check the service configuration status of related services.

Choose **Cluster** > **Service** to view the configuration status of other related services. If the configuration of a service has expired, choose **Cluster** > **Dashboard**, select **Restart Configuration-Expired Instances** from the **More** drop-down list, enter the password, and click **OK** to restart it.

Figure 7-10 Restarting configuration-expired instances



----End

8 Configuring Auto Scaling for an MRS Cluster

In big data application scenarios, especially real-time data analysis and processing, the number of cluster nodes needs to be dynamically adjusted according to data volume changes to provide proper resources. The auto scaling function of MRS enables clusters to be automatically scaled out or in based on cluster load.

- Auto scaling rules: You can increase or decrease Task nodes based on real-time cluster loads. Auto scaling will be triggered when the data volume changes but there may be some delays.
- Resource plan (setting the task node quantity based on the time range): If the data volume changes periodically, you can create resource plans to resize the cluster before the data volume changes, thereby avoiding delays in increasing or decreasing resources.

You can configure either auto scaling rules or resource plans or both of them to trigger the auto scaling.

Scenario

The following example describes how to use both auto scaling rules and resource plans:

A real-time processing service sees an unstable increase in data volume from 7:00 to 13:00 on Monday, Tuesday, and Saturday. For example, 5 to 8 task nodes are required from 7:00 to 13:00 on Monday, Tuesday, and Saturday, and 2 to 4 are required beyond this period.

You can set an auto scaling rule based on a resource plan. When the data volume exceeds the expected value, the number of Task nodes changes with resource loads, without exceeding the node range specified in the resource plan. When a resource plan is triggered, the number of nodes changes within the specified range with minimum affect. That is, increase nodes to the upper limit and decrease nodes to the lower limit.

Video Tutorial

This video uses an MRS 3.1.0 cluster as an example to describe how to configure an auto scaling policy when you purchase a cluster and how to add an auto

scaling policy to an existing cluster. For details, see **Configuring Auto Scaling for an MRS Cluster**.

NOTE

The UI may vary according to the version. The video tutorial is for reference only.

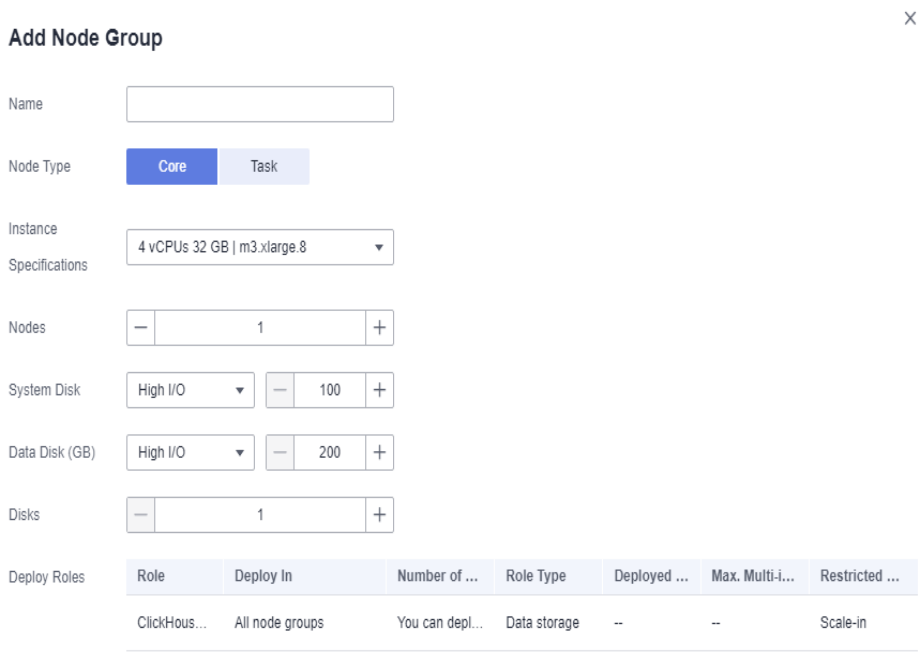
Adding a Task Node

You can scale out an MRS cluster by manually adding task nodes.

To add a task node to a custom cluster, perform the following steps:

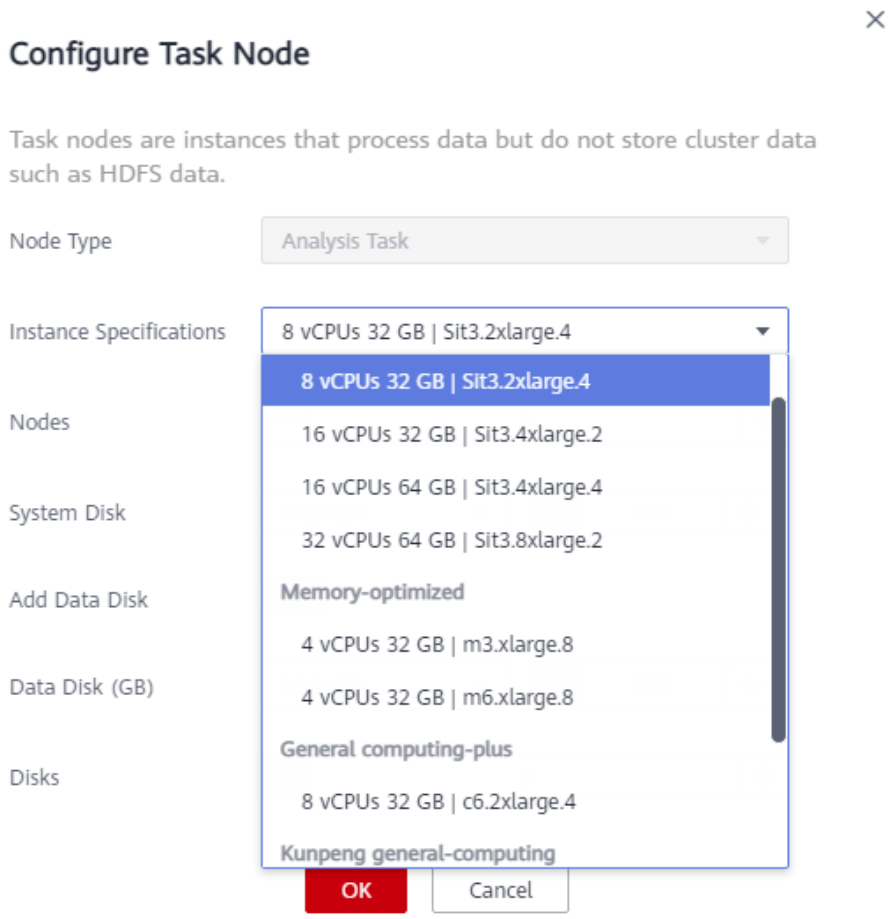
1. On the cluster details page, click the **Nodes** tab and click **Add Node Group**. The **Add Node Group** page is displayed.
2. Retain the default value **NM** for **Deploy Roles**. To deploy the NodeManager role, the node type must be **Task**. Set other parameters as required.

Figure 8-1 Adding a task node group



To add a task node to a non-custom cluster, perform the following steps:

1. On the cluster details page, click the **Nodes** tab and click **Configure Task Node**. The **Configure Task Node** page is displayed.
2. On the **Configure Task Node** page, set **Node Type**, **Instance Specifications**, **Nodes**, **System Disk**. In addition, if **Add Data Disk** is enabled, configure the storage type, size, and number of data disks.



3. Click **OK**.

Using Auto Scaling Rules and Resource Plans Together

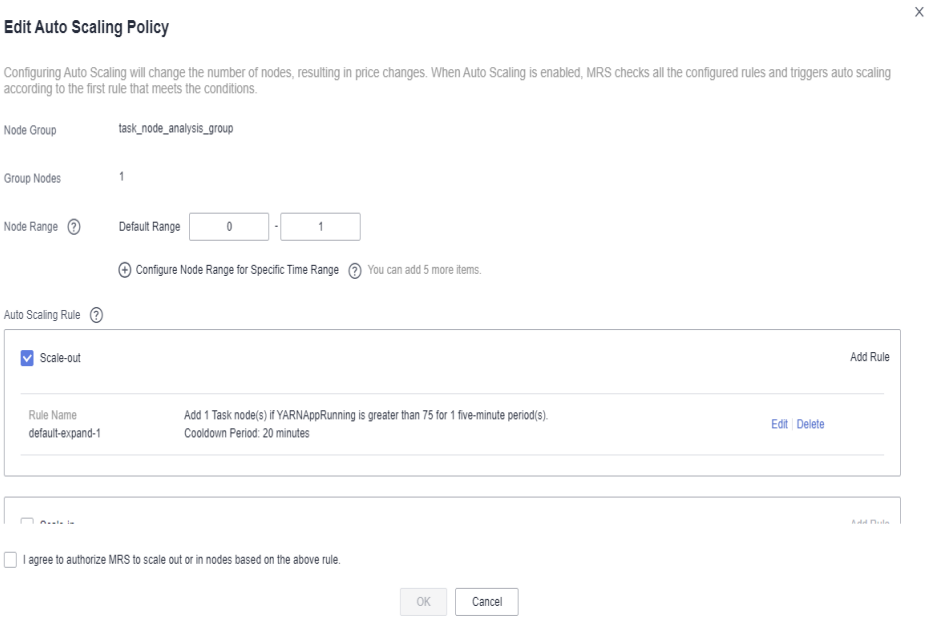
Step 1 Log in to the MRS management console.

Step 2 Choose **Clusters** > **Active Clusters**, and click the name of the target cluster. The cluster details page is displayed.

Step 3 On the page that is displayed, click the **Auto Scaling** tab.

Step 4 Click **Add Auto Scaling Policy** and set **Node Range** to **2-4**.

Figure 8-2 Configuring auto scaling



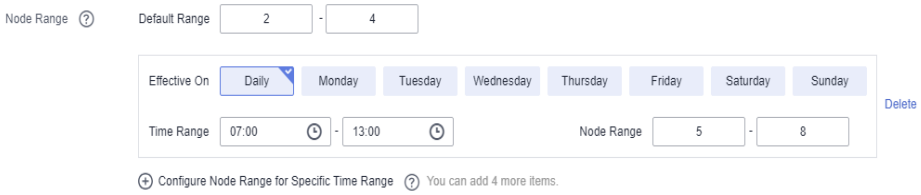
Step 5 Configure a resource plan.

1. Click **Configure Node Range for Specific Time Range** under **Default Range**.
2. Configure the **Time Range** and **Node Range** parameters.

   Time Range: Set it to **07:00-13:00**.

   Node Range: Set it to **5-8**.

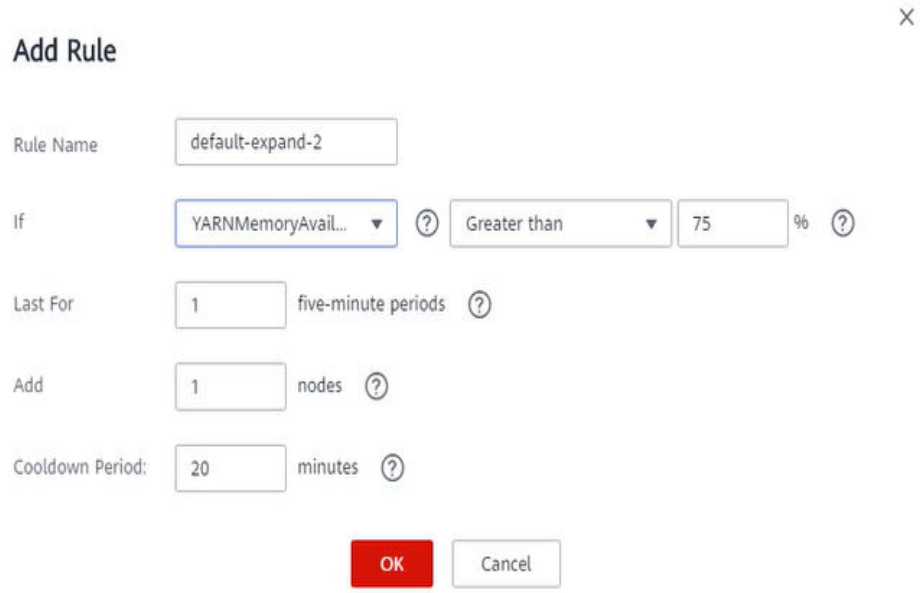
   Figure 8-3 Auto scaling



Step 6 Configure an auto scaling rule.

1. Select **Scale-out**.
2. Click **Add Rule** on the right.

Figure 8-4 Adding a rule



Rule Name: **default-expand-2**.

If: Select the rule objects and constraints from the drop-down list boxes, for example, YARNAppRunning is greater than 75.

Last For: Set it to **1 five-minute periods**.

Add: Set it to **1 node**.

Cooldown Period: Set it to **20 minutes**.

3. Click **OK**.

Step 7 Select **I agree to authorize MRS to scale out or in nodes based on the above rule**.

Step 8 Click **OK**.

----End

Reference Information

When adding a rule, you can refer to **Table 8-1** to configure the corresponding metrics.

NOTE

- Hybrid clusters support all metrics of analysis and streaming clusters.
- The accuracy of different value types in **Table 8-1** is as follows:
  - **Integer**: integer
  - **Percentage**: 0.01
  - **Ratio**: 0.01

Table 8-1 Auto scaling metrics

Cluster Type	Metric	Value Type	Description
Streaming cluster	StormSlotAvailable	Integer	Number of available Storm slots. Value range: 0 to 2147483646.
	StormSlotAvailablePercentage	Percentage	Percentage of available Storm slots, that is, the proportion of the available slots to total slots. Value range: 0 to 100.
	StormSlotUsed	Integer	Number of used Storm slots. Value range: 0 to 2147483646.
	StormSlotUsedPercentage	Percentage	Percentage of the used Storm slots, that is, the proportion of the used slots to total slots. Value range: 0 to 100.
	StormSupervisorMemAverageUsage	Integer	Average memory usage of the Supervisor process of Storm. Value range: 0 to 2147483646.
	StormSupervisorMemAverageUsagePercentage	Percentage	Average percentage of the used memory of the Supervisor process of Storm to the total memory of the system. Value range: 0 to 100.
	StormSupervisorCPUAverageUsagePercentage	Percentage	Average percentage of the used CPUs of the Supervisor process of Storm to the total CPUs. Value range: [0, 6000].
Analysis cluster	YARNAppPending	Integer	Number of pending tasks on Yarn. Value range: 0 to 2147483646.
	YARNAppPendingRatio	Ratio	Ratio of pending tasks on Yarn, that is, the ratio of pending tasks to running tasks on Yarn. Value range: 0 to 2147483646.
	YARNAppRunning	Integer	Number of running tasks on Yarn. Value range: 0 to 2147483646.
	YARNContainerAllocated	Integer	Number of containers allocated to YARN. Value range: 0 to 2147483646.

Cluster Type	Metric	Value Type	Description
	YARNContainerPending	Integer	Number of pending containers on Yarn. Value range: 0 to 2147483646.
	YARNContainerPendingRatio	Ratio	Ratio of pending containers on Yarn, that is, the ratio of pending containers to running containers on Yarn. Value range: 0 to 2147483646.
	YARNCPUAllocated	Integer	Number of virtual CPUs (vCPUs) allocated to Yarn. Value range: 0 to 2147483646.
	YARNCPUAvailable	Integer	Number of available vCPUs on Yarn. Value range: 0 to 2147483646.
	YARNCPUAvailablePercentage	Percentage	Percentage of available vCPUs on Yarn, that is, the proportion of available vCPUs to total vCPUs. Value range: 0 to 100.
	YARNCPUPending	Integer	Number of pending vCPUs on Yarn. Value range: 0 to 2147483646.
	YARNMemoryAllocated	Integer	Memory allocated to Yarn. The unit is MB. Value range: 0 to 2147483646.
	YARNMemoryAvailable	Integer	Available memory on Yarn. The unit is MB. Value range: 0 to 2147483646.
	YARNMemoryAvailablePercentage	Percentage	Percentage of available memory on Yarn, that is, the proportion of available memory to total memory on Yarn. Value range: 0 to 100.
	YARNMemoryPending	Integer	Pending memory on Yarn. Value range: 0 to 2147483646.

When adding a resource plan, you can set parameters by referring to **Table 8-2**.

Table 8-2 Configuration items of a resource plan

Parameter	Description
Effective On	The effective date of a resource plan. **Daily** is selected by default. You can also select one or multiple days from Monday to Sunday.
Time Range	Start time and end time of a resource plan are accurate to minutes, with the value ranging from **00:00** to **23:59**. For example, if a resource plan starts at 8:00 and ends at 10:00, set this parameter to **8:00-10:00**. The end time must be at least 30 minutes later than the start time. Time ranges configured for different resource plans cannot overlap.
Node Range	The number of nodes in a resource plan ranges from **0** to **500**. In the time range specified in the resource plan, if the number of task nodes is less than the specified minimum number of nodes, it will be increased to the specified minimum value of the node range at a time. If the number of task nodes is greater than the maximum number of nodes specified in the resource plan, the auto scaling function reduces the number of task nodes to the maximum value of the node range at a time. The minimum number of nodes must be less than or equal to the maximum number of nodes.

9 Configuring Hive with Storage and Compute Decoupled

MRS allows you to store data in OBS and use an MRS cluster for data computing only. In this way, storage and compute are decoupled. You can use the IAM service to perform simple configurations to access OBS.

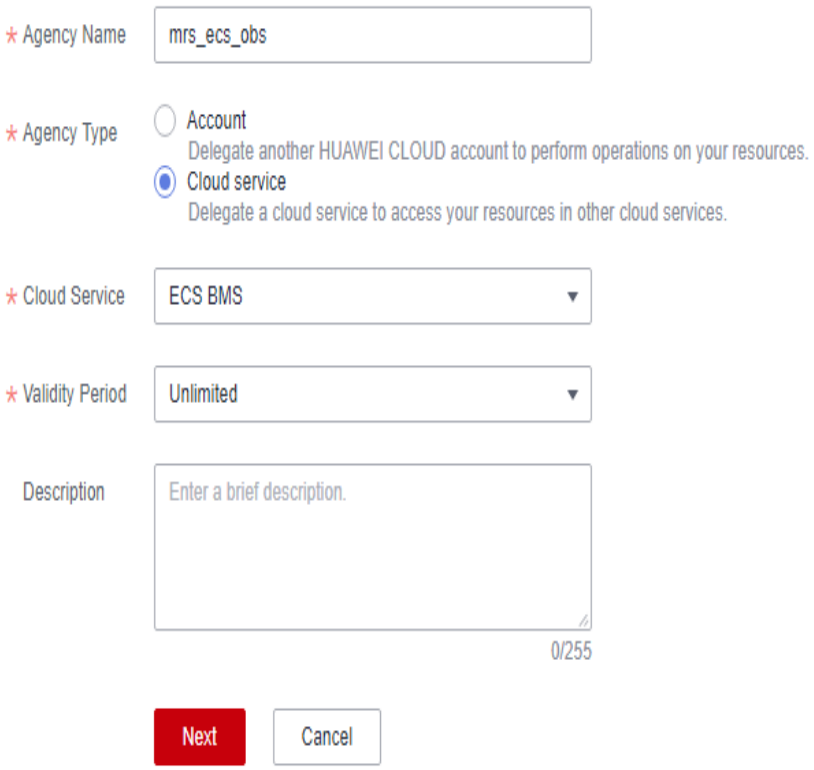
This section describes how to create a Hive table to store data to OBS.

1. **Creating an ECS Agency**
2. **Configuring an Agency for an MRS Cluster**
3. **Creating an OBS File System**
4. **Accessing the OBS File System Through Hive**

Creating an ECS Agency

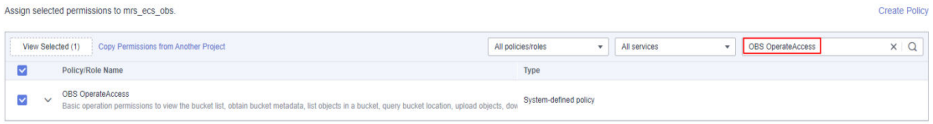
1. Log in to the Huawei Cloud management console.
2. Choose **Service List** > **Management & Governance** > **Identity and Access Management**.
3. Click **Agencies**. On the displayed page, click **Create Agency**.
4. Enter an agency name, for example, **mrs_ecs_obs**.
5. Set **Agency Type** to **Cloud service** and select **ECS BMS** to authorize ECS or BMS to invoke OBS.
6. Set **Validity Period** to **Unlimited** and click **Next**.

Figure 9-1 Creating an agency



7. On the page that is displayed, search for **OBS OperateAccess** in the search box and select it in the result list.

Figure 9-2 Assigning permissions



8. Click **Next**. On the page that is displayed, select the desired scope for the permissions you selected. By default, **All resources** is selected. Click **Show More**, select **Global resources**, and click **OK**.
9. In the dialog box that is displayed, click **OK** to start authorization. After the message "Authorization successful." is displayed, click **Finish**. The agency is created successfully.

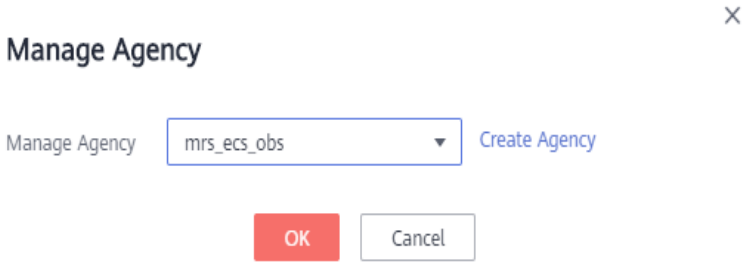
Configuring an Agency for an MRS Cluster

You can configure an agency when creating a cluster or bind an agency to an existing cluster to decouple storage and compute. This section uses an existing cluster as an example to describe how to configure an agency.

1. Log in to the MRS console. In the navigation pane on the left, choose **Clusters** > **Active Clusters**.
2. Click the name of a cluster to go to the cluster details page.
3. On the **Dashboard** page, click **Synchronize** on the right side of **IAM User Sync** to synchronize IAM users.

4. On the **Dashboard** page, click **Manage Agency** on the right side of **Agency** to select the agency created in **Creating an ECS Agency**, and click **OK** to bind it to the cluster. Alternatively, click **Create Agency** to go to the IAM console to create an agency and bind it to the cluster.

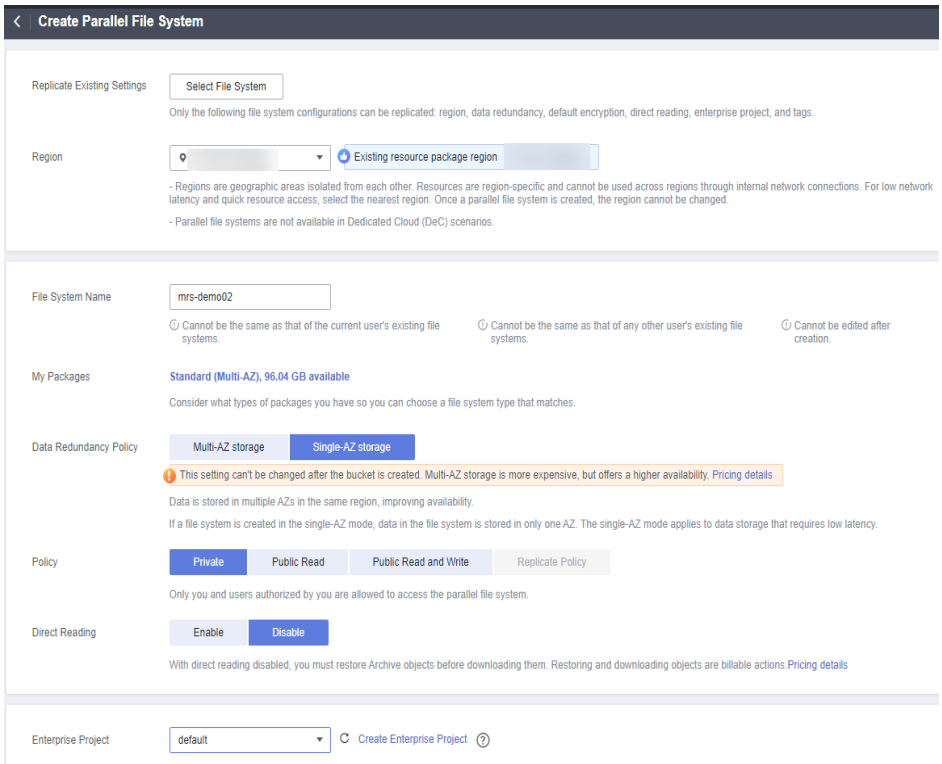
Figure 9-3 Binding an agency



Creating an OBS File System

1. Log in to the OBS console.
2. Choose **Parallel File System** > **Create Parallel File System**.
3. Enter the file system name, for example, **mrs-demo01**.
   Set other parameters as required.

Figure 9-4 Creating a parallel file system



4. Click **Create Now**.
5. In the parallel file system list on the OBS console, click a file system name to go to the details page.

6. In the navigation pane, choose **Files** and create **program** and **input** folders.
   - **program**: Upload the program package to this folder.
   - **input**: Upload the input data to this folder.

Accessing the OBS File System Through Hive

1. Log in to a master node as user **root**. For details, see **Logging In to an ECS**.
2. Verify that Hive can access OBS.
   a. Log in to the master node of the cluster as user **root** and run the following commands:

      cd /opt/Bigdata/client

      source bigdata_env

      source Hive/component_env

   b. View the list of files in file system **mrs-demo01**.

      hadoop fs -ls obs://mrs-demo01/

   c. Check whether the file list is returned. If it is returned, access to OBS is successful.

      Figure 9-5 Viewing the file list in mrs-demo01

      ```
      Found 2 items
      drwxrwxrwx   - hive hive          0 2021-10-22 10:08 obs://mrs-demo01/input
      drwxrwxrwx   - hive hive          0 2021-10-22 10:08 obs://mrs-demo01/program
      ```

   d. Run the following command to authenticate the user (skip this step for a normal cluster, that is, with Kerberos authentication disabled):

      kinit hive

      Enter the password of user **hive**. The default password is **Hive@123**. Change the password upon the first login.

   e. Run the Hive client command.

      beeline

   f. Access the OBS directory in the Beeline. For example, run the following command to create a Hive table and specify that data is stored in the **test_demo01** table of file system **mrs-demo01**:

      create table test_demo01(name string) location "obs://mrs-demo01/test_demo01";

   g. Run the following command to query all tables. If the **test_demo01** table is displayed in the command output, the access to OBS is successful.

      show tables;

      Figure 9-6 Checking whether the test_demo01 table exists

      ```
      +--------------+
      |   tab_name   |
      +--------------+
      | test_demo01  |
      +--------------+
      1 row selected (0.301 seconds)
      ```


h. Run the following command to check the table location.

show create table test_demo01;

Check whether the location of the table starts with **obs://***OBS bucket name***/**.

Figure 9-7 Checking the location of the test_demo01 table

```
  'serialization.format' = ',' )
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTe
LOCATION
  'obs://mrs-demo01/test_demo01'
TBLPROPERTIES (
  'bucketing_version'='2',
  'transient_lastDdlTime'='1634872329')
```

i. Run the following command to write data into the table.

insert into test_demo01 values('mm'),('ww'),('ww');

Run the **select * from test_demo01;** command to check whether the data is written successfully.

Figure 9-8 Viewing data in the test_demo01 table

```
+--------------------+
| test_demo01.name   |
+--------------------+
| mm                 |
| ww                 |
| ww                 |
+--------------------+
```

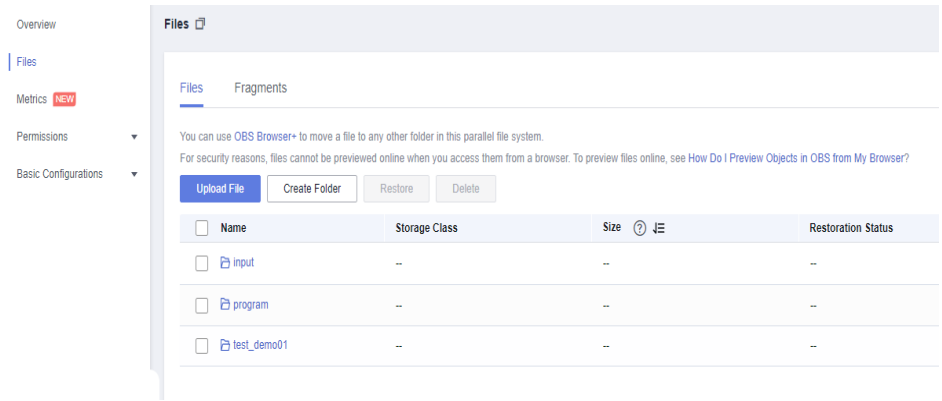
j. Run the **!q** command to exit the Beeline client.

k. Log in to the OBS console again.

l. Click **Parallel File System** and select the created file system.

m. Click **Files** to check whether the data exists in the created table.

Figure 9-9 Viewing data



10 Submitting Spark Tasks to New Task Nodes

Add task nodes to a custom MRS cluster to increase compute capability. Task nodes are mainly used to process data instead of permanently storing data.

NOTE

Currently, task nodes can only be added to custom MRS clusters.

This section describes how to bind a new task node using tenant resources and submit Spark tasks to the new task node. You can get started by reading the following topics:

1. **Adding Task Nodes**
2. **Creating a Resource Pool**
3. **Creating a Tenant**
4. **Configuring Queues**
5. **Configuring Resource Distribution Policies**
6. **Creating a User**
7. **Using spark-submit to Submit a Task**
8. **Deleting Task Nodes**

Adding Task Nodes

1. On the details page of a custom MRS cluster, click the **Nodes** tab. On this tab page, click **Add Node Group**.
2. On the **Add Node Group** page that is displayed, set parameters as needed.

Table 10-1 Parameters for adding a node group

Parameter	Description
Instance Specifications	Select the flavor type of the hosts in the node group.

Parameter	Description
Nodes	Configure the number of nodes in the node group.
System Disk	Configure the specifications and capacity of the system disks on the new nodes.
Data Disk (GB)/Disks	Set the specifications, capacity, and number of data disks of the new nodes.
Deploy Roles	Select **NM** to add a NodeManager role.

3. Click **OK**.

Creating a Resource Pool

Step 1 On the cluster details page, click **Tenants**.

Step 2 Click **Resource Pools**.

Step 3 Click **Create Resource Pool**.

Step 4 On the **Create Resource Pool** page, set the properties of the resource pool.

- **Name**: Enter the name of the resource pool, for example, **test1**.
- **Resource Label**: Enter the resource pool label, for example, **1**.
- **Available Hosts**: Enter the node added in **Adding Task Nodes**.

Step 5 Click **OK**.

----End

Creating a Tenant

Step 1 On the cluster details page, click **Tenants**.

Step 2 Click **Create Tenant**. On the displayed page, configure tenant properties.

Table 10-2 Tenant parameters

Parameter	Description
Name	Set the tenant name, for example, **tenant_spark**.
Tenant Type	Select **Leaf**. If **Leaf** is selected, the current tenant is a leaf tenant and no sub-tenant can be added. If **Non-leaf** is selected, sub-tenants can be added to the current tenant.
Dynamic Resource	If **Yarn** is selected, the system automatically creates a task queue using the tenant name in Yarn. If **Yarn** is not selected, the system does not automatically create a task queue.

Parameter	Description
Default Resource Pool Capacity (%)	Set the percentage of computing resources used by the current tenant in the **default** resource pool, for example, **20%**.
Default Resource Pool Max. Capacity (%)	Set the maximum percentage of computing resources used by the current tenant in the **default** resource pool, for example, **80%**.
Storage Resource	If HDFS is selected, the system automatically creates the **/tenant** directory under the root directory of the HDFS when a tenant is created for the first time. If **HDFS** is not selected, the system does not create a storage directory under the root directory of the HDFS.
Maximum Number of Files/Directories	Set the maximum number of files or directories, for example, **100000000000**.
Storage Space Quota (MB)	Set the quota for using the storage space, for example, 50000 MB. This parameter indicates the maximum HDFS storage space that can be used by a tenant, but not the actual space used. If its value is greater than the size of the HDFS physical disk, the maximum space available is the full space of the HDFS physical disk. **NOTE** To ensure data reliability, the system automatically generates one backup file when a file is stored in the HDFS. That is, two replicas of the same file are stored by default. The HDFS storage space indicates the total disk space occupied by all these replicas. For example, if the value of **Storage Space Quota** is set to **500**, the actual space for storing files is about 250 MB (500/2 = 250).
Storage Path	Set the storage path, for example, **tenant/spark_test**. The system automatically creates a folder named after the tenant under the **/tenant** directory by default, for example, **spark_test**. The default HDFS storage directory for tenant **spark_test** is **tenant/spark_test**. When a tenant is created for the first time, the system creates the **/tenant** directory in the HDFS root directory. The storage path is customizable.
Services	Set other service resources associated with the current tenant. HBase is supported. To configure this parameter, click **Associate Services**. In the displayed dialog box, set **Service** to **HBase**. If **Association Mode** is set to **Exclusive**, service resources are occupied exclusively. If **share** is selected, service resources are shared.
Description	Enter the description of the current tenant.

Step 3 Click **OK** to save the settings.

It takes a few minutes to save the settings. If the **Tenant created successfully** is displayed in the upper-right corner, the tenant is added successfully.

NOTE

- Roles, computing resources, and storage resources are automatically created when tenants are created.
- The new role has permissions on the computing and storage resources. The role and its permissions are controlled by the system automatically and cannot be controlled manually under **Manage Role**.
- If you want to use the tenant, create a system user and assign the Manager_tenant role and the role corresponding to the tenant to the user.

----End

Configuring Queues

Step 1 On the cluster details page, click **Tenants**.

Step 2 Click the **Queue Configuration** tab.

Step 3 In the tenant queue table, click **Modify** in the **Operation** column of the specified tenant queue.

NOTE

- In the tenant list on the left of the **Tenant Management** page, click the target tenant.

  In the displayed window, choose **Resource**. On the displayed page, click  to open the queue modification page.
- A queue can be bound to only one non-default resource pool.

By default, the resource tag is the one specified in **Creating a Resource Pool**. Set other parameters based on the site requirements.

Step 4 Click **OK**.

----End

Configuring Resource Distribution Policies

Step 1 On the cluster details page, click **Tenants**.

Step 2 Click **Resource Distribution Policies** and select the resource pool created in **Creating a Resource Pool**.

Step 3 Locate the row that contains **tenant_spark**, and click **Modify** in the **Operation** column.

- **Weight**: **20**
- **Minimum Resource**: **20**
- **Maximum Resource**: **80**
- **Reserved Resource**: **10**

Step 4 Click **OK**.

----End

Creating a User

Step 1 Log in to FusionInsight Manager. For details, see **Accessing FusionInsight Manager**.

Step 2 Choose **System** > **Permission** > **User**. On the displayed page, click **Create User**.

- **Username**: **spark_test**
- **User Type**: **Human-Machine**
- **User Group**: hadoop and hive
- **Primary Group**: **hadoop**
- **Role**: **tenant_spark**

Step 3 Click **OK** to add the user.

----End

Using spark-submit to Submit a Task

1. Log in to the client node as user **root** and run the following commands:

   cd *Client installation directory*

   source bigdata_env

   source Spark2x/component_env

   For a cluster with Kerberos authentication enabled, run the **kinit spark_test** command. For a cluster with Kerberos authentication disabled, skip this step.

   Enter the password for authentication. Change the password upon the first login.

   cd Spark2x/spark/bin

   sh spark-submit --queue tenant_spark --class org.apache.spark.examples.SparkPi --master yarn-client ../examples/jars/spark-examples_*.jar

Deleting Task Nodes

1. On the cluster details page, click **Nodes**.
2. Locate the row that contains the target task node group, and click **Scale In** in the **Operation** column.
3. Set the **Scale-In Type** to **Specific node** and select the target nodes.

   NOTE

   The target nodes need to be shut down.

4. Select **I understand the consequences of performing the scale-in operation**, and click **OK**.

11 Configuring Thresholds for Alarms

MRS clusters provide easy-to-use alarming functions with intuitive monitoring metric views. You can quickly view statistics on key performance metrics (KPIs) of a cluster and evaluate the cluster health status. MRS allows you to configure metric thresholds to stay informed of cluster health status. If a threshold value is met, the system generates and displays an alarm on the metric dashboard.

If it is **verified** that the impact of some alarms on services can be ignored or the alarm thresholds need to be adjusted, you can customize cluster metrics or mask some alarms as required.

You can set thresholds for alarms of node information metrics and cluster service metrics. For details about these metrics, their impacts on the system, and default thresholds, see **Monitoring Metric Reference**.

NOTICE

These alarms may affect cluster functions or job running. If you want to mask or modify alarm rules, evaluate operation risks in advance.

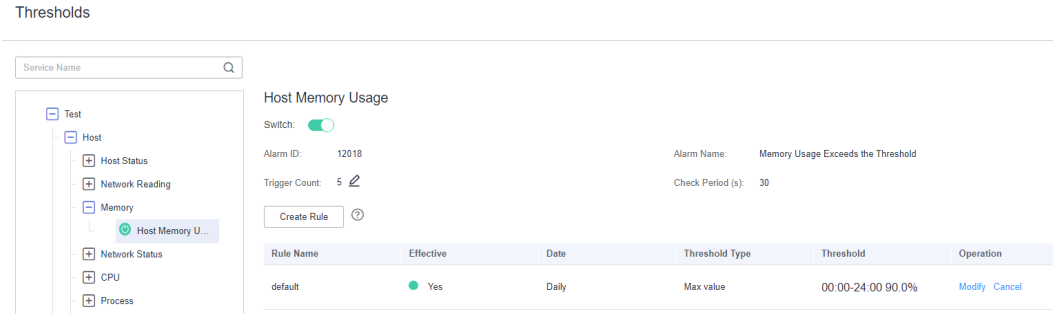
Modifying Rules for Alarms with Custom Thresholds

Step 1 Log in to FusionInsight Manager of the target MRS cluster by referring to **Accessing Log in the FusionInsight Manager (MRS 3.x or Later)**.

Step 2 Choose **O&M** > **Alarm** > **Thresholds**.

Step 3 Select a metric for a host or service in the cluster. For example, select **Host Memory Usage**.

Figure 11-1 Viewing an alarm threshold



- **Switch**: If this switch is turned on, an alarm will be triggered when the metric breaches this threshold.
- **Trigger Count**: Manager checks whether the metric meets the threshold value. If the number of consecutive checks where the metric fails equals the value of **Trigger Count**, an alarm is generated. The value can be customized. **If an alarm is frequently reported, you can set Trigger Count to a bigger value to reduce the alarming frequency.**
- **Check Period (s)**: Interval between each two checks
- The rules to trigger alarms are listed on the page.

Step 4 Modify an alarm rule.

- Add a new rule.
  a. Click **Create Rule** to add a rule that defines how an alarm will be triggered. For details, see **Table 11-1**.
  b. Click **OK** to save the rule.
  c. Locate the row that contains a rule that is in use, and click **Cancel** in the **Operation** column. If no rule is in use, skip this step.
  d. Locate the row that contains the new rule, and click **Apply** in the **Operation** column. The value of **Effective** for this rule changes to **Yes**.
- Modify an existing rule.
  a. Click **Modify** in the **Operation** column of the row that contains the target rule.
  b. Modify rule parameters by referring to **Table 11-1**.
  c. Click **OK**.

The following table lists the rule parameters you need to set for triggering an alarm of **Host Memory Usage**.

Table 11-1 Alarm rule parameters

Parameter	Description	Example Value
Rule Name	Rule name	mrs_test

Parameter	Description	Example Value
Severity	Alarm severity. The options are as follows: • **Critical** • **Major** • **Minor** • **Warning**	Major
Threshold Type	Maximum or minimum value of a metric • **Max value**: An alarm will be generated when the metric value is greater than this value. • **Min value**: An alarm will be generated when the metric value is less than this value.	Max. Value
Date	How often the rule takes effect • Daily • Weekly • Others	Daily
Add Date	Date when the rule takes effect. This parameter is available only when **Date** is set to **Others**. You can set multiple dates.	-
Thresholds	**Start and End Time**: Period when the rule takes effect.	00:00 - 23:59
	Threshold: Alarm threshold value	85

----End

Masking Specified Alarms

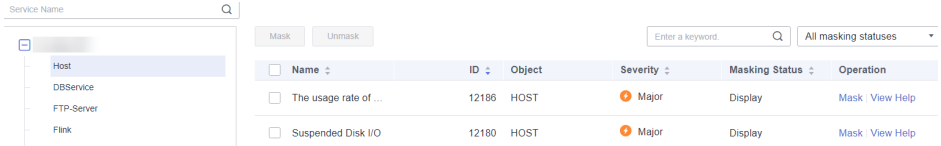
Step 1 Log in to FusionInsight Manager of the target MRS cluster by referring to **Accessing Log in the FusionInsight Manager (MRS 3.x or Later)**.

Step 2 Choose **O&M** > **Alarm** > **Masking**.

Step 3 In the list on the left of the displayed page, select the target service or module.

Step 4 Click **Mask** in the **Operation** column of the alarm you want to mask. In the dialog box that is displayed, click **OK** to change the masking status of the alarm to **Masking**.

Figure 11-2 Masking an alarm



NOTE

- You can search for specified alarms in the list.
- To cancel alarm masking, click **Unmask** in the row of the target alarm. In the dialog box that is displayed, click **OK** to change the alarm masking status to **Display**.
- If you need to perform operations on multiple alarms at a time, select the alarms and click **Mask** or **Unmask** on the top of the list.

----End

FAQ

- **How Do I View Uncleared Alarms in a Cluster?**
  a. Log in to the MRS management console.
  b. Click the name of the target cluster and click the **Alarms** tab.
  c. Click **Advanced Search**, set **Alarm Status** to **Uncleared**, and click **Search**.
  d. Uncleared alarms of the current cluster are displayed.
- **How Do I Clear a Cluster Alarm?**
  You can handle the alarms by referring to the alarm help. To view the help document, perform the following steps:
  – Console: Log in to the MRS management console, click the name of the target cluster, click the **Alarms** tab, and click **View Help** in the **Operation** column of the alarm list. Then, clear the alarm by referring to the alarm handling procedure.
  – Manager: Log in to FusionInsight Manager, choose **O&M** > **Alarm** > **Alarms**, and click **View Help** in the **Operation** column. Then, clear the alarm by referring to the alarm handling procedure.

Monitoring Metric Reference

FusionInsight Manager monitoring metrics are classified as node information metrics and cluster service metrics. **Table 11-2** lists the metrics whose thresholds can be configured a node, and **Table 11-3** lists metrics whose thresholds can be configured for a component.

Table 11-2 Node monitoring metrics

Metric Group	Metric	ID	Alarm	Impact on System	Default Threshold
CPU	Host CPU Usage	12016	CPU Usage Exceeds the Threshold	Service processes respond slowly or become unavailable.	90.0%
Disk	Disk Usage	12017	Insufficient Disk Capacity	Service processes become unavailable.	90.0%

Metric Group	Metric	ID	Alarm	Impact on System	Default Threshold
	Disk Inode Usage	12051	Disk Inode Usage Exceeds the Threshold	Data cannot be properly written to the file system.	80.0%
Memory	Host Memory Usage	12018	Memory Usage Exceeds the Threshold	Service processes respond slowly or become unavailable.	90.0%
Host Status	Host File Handle Usage	12053	Host File Handle Usage Exceeds the Threshold	The I/O operations, such as opening a file or connecting to network, cannot be performed and programs are abnormal.	80.0%
	Host PID Usage	12027	Host PID Usage Exceeds the Threshold	No PID is available for new processes and service processes are unavailable.	90%
Network Status	TCP Temporary Port Usage	12052	TCP Temporary Port Usage Exceeds the Threshold	Services on the host fail to establish connections with the external and services are interrupted.	80.0%
Network Reading	Read Packet Error Rate	12047	Read Packet Error Rate Exceeds the Threshold	The communication is intermittently interrupted, and services time out.	0.5%
	Read Packet Dropped Rate	12045	Read Packet Dropped Rate Exceeds the Threshold	The service performance deteriorates or some services time out.	0.5%
	Read Throughput Rate	12049	Read Throughput Rate Exceeds the Threshold	The service system runs abnormally or is unavailable.	80%

Metric Group	Metric	ID	Alarm	Impact on System	Default Threshold
Network Writing	Write Packet Error Rate	12048	Write Packet Error Rate Exceeds the Threshold	The communication is intermittently interrupted, and services time out.	0.5%
	Write Packet Dropped Rate	12046	Write Packet Dropped Rate Exceeds the Threshold	The service performance deteriorates or some services time out.	0.5%
	Write Throughput Rate	12050	Write Throughput Rate Exceeds the Threshold	The service system runs abnormally or is unavailable.	80%
Process	Total Number of Processes in D and Z States	12028	Number of Processes in the D State and Z State on a Host Exceeds the Threshold	Excessive system resources are used and service processes respond slowly.	0
	omm Process Usage	12061	Process Usage Exceeds the Threshold	Switch to user **omm** fails. New omm process cannot be created.	90

Table 11-3 Cluster monitoring metrics

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
DBService	Usage of the Number of Database Connections	27005	Database Connection Usage Exceeds the Threshold	Upper-layer services may fail to connect to the DBService database, affecting services.	90%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Disk Space Usage of the Data Directory	27006	Disk Space Usage of the Data Directory Exceeds the Threshold	Service processes become unavailable. When the disk space usage of the data directory exceeds 90%, the database enters the read-only mode and **Database Enters the Read-Only Mode** is generated. As a result, service data is lost.	80%
Flume	Heap Memory Resource Percentage	24006	Heap Memory Usage of Flume Server Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95.0%
	Direct Memory Usage Statistics	24007	Flume Server Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%
	Non-heap Memory Usage	24008	Flume Server Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80.0%
	Total GC Duration	24009	Flume Server GC Duration Exceeds the Threshold	Flume data transmission efficiency decreases.	12000 ms
HBase	GC Duration of Old Generation	19007	HBase GC Duration Exceeds the Threshold	If the old generation GC duration exceeds the threshold, HBase data read and write are affected.	5000ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	RegionServer Direct Memory Usage Statistics	19009	Direct Memory Usage of the HBase Process Exceeds the Threshold	If the available HBase direct memory is insufficient, a memory overflow occurs and the service breaks down.	90%
	RegionServer Heap Memory Usage Statistics	19008	Heap Memory Usage of the HBase Process Exceeds the Threshold	If the available HBase memory is insufficient, a memory overflow occurs and the service breaks down.	90%
	HMaster Direct Memory Usage	19009	Direct Memory Usage of the HBase Process Exceeds the Threshold	If the available HBase direct memory is insufficient, a memory overflow occurs and the service breaks down.	90%
	HMaster Heap Memory Usage Statistics	19008	Heap Memory Usage of the HBase Process Exceeds the Threshold	If the available HBase memory is insufficient, a memory overflow occurs and the service breaks down.	90%
	Number of Online Regions of a RegionServer	19011	Number of RegionServer Regions Exceeds the Threshold	The data read/write performance of HBase is affected when the number of regions on a RegionServer exceeds the threshold.	2000
	Region in RIT State That Reaches the Threshold Duration	19013	Duration of Regions in RIT State Exceeds the Threshold	Some data in the table is lost or becomes unavailable.	1

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Handler Usage of RegionServer	19021	Number of Active Handlers of RegionServer Exceeds the Threshold	RegionServers or HBase cannot provide services properly.	90%
	Synchronization Failures in Disaster Recovery	19006	HBase Replication Sync Failed	HBase data in a cluster fails to be synchronized to the standby cluster, causing data inconsistency between active and standby clusters.	1
	Number of Log Files to Be Synchronized in the Active Cluster	19020	Number of HBase WAL Files to Be Synchronized Exceeds the Threshold	If the number of WAL files to be synchronized by a RegionServer exceeds the threshold, the number of ZNodes used by HBase exceeds the threshold, affecting the HBase service status.	128
	Number of HFiles to Be Synchronized in the Active Cluster	19019	Number of HFiles to Be Synchronized Exceeds the Threshold	If the number of HFiles to be synchronized by a RegionServer exceeds the threshold, the number of ZNodes used by HBase exceeds the threshold, affecting the HBase service status.	128

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Compaction Queue Size	19018	HBase Compaction Queue Size Exceeds the Threshold	The cluster performance may deteriorate, affecting data read and write.	100
HDFS	Lost Blocks	14003	Number of Lost HDFS Blocks Exceeds the Threshold	Data stored in HDFS is lost. HDFS may enter the security mode and cannot provide write services. Lost block data cannot be restored.	0
	Blocks Under Replicated	14028	Number of Blocks to Be Supplemented Exceeds the Threshold	Data stored in HDFS is lost. HDFS may enter the security mode and cannot provide write services. Lost block data cannot be restored.	1000
	Average Time of Active NameNode RPC Processing	14021	Average NameNode RPC Processing Time Exceeds the Threshold	NameNode cannot process the RPC requests from HDFS clients, upper-layer services that depend on HDFS, and DataNode in a timely manner. Specifically, the services that access HDFS run slowly or the HDFS service is unavailable.	100ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Average Time of Active NameNode RPC Queuing	14022	Average NameNode RPC Queuing Time Exceeds the Threshold	NameNode cannot process the RPC requests from HDFS clients, upper-layer services that depend on HDFS, and DataNode in a timely manner. Specifically, the services that access HDFS run slowly or the HDFS service is unavailable.	200ms
	HDFS Disk Usage	14001	HDFS Disk Usage Exceeds the Threshold	The performance of writing data to HDFS is affected.	80%
	DataNode Disk Usage	14002	DataNode Disk Usage Exceeds the Threshold	Insufficient disk space will impact data write to HDFS.	80%
	Percentage of Reserved Space for Replicas of Unused Space	14023	Percentage of Total Reserved Disk Space for Replicas Exceeds the Threshold	The performance of writing data to HDFS is affected. If all unused DataNode space is reserved for replicas, writing HDFS data fails.	90%
	Total Faulty DataNodes	14009	Number of Dead DataNodes Exceeds the Threshold	Faulty DataNodes cannot provide HDFS services.	3
	NameNode Non-Heap Memory Usage Statistics	14018	NameNode Non-Heap Memory Usage Exceeds the Threshold	If the non-heap memory usage of the HDFS NameNode is too high, data read/write performance of HDFS will be affected.	90%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	NameNode Direct Memory Usage Statistics	14017	NameNode Direct Memory Usage Exceeds the Threshold	If the available direct memory of NameNode instances is insufficient, a memory overflow may occur and the service breaks down.	90%
	NameNode Heap Memory Usage Statistics	14007	NameNode Heap Memory Usage Exceeds the Threshold	If the heap memory usage of the HDFS NameNode is too high, data read/write performance of HDFS will be affected.	95%
	DataNode Direct Memory Usage Statistics	14016	DataNode Direct Memory Usage Exceeds the Threshold	If the available direct memory of DataNode instances is insufficient, a memory overflow may occur and the service breaks down.	90%
	DataNode Heap Memory Usage Statistics	14008	DataNode Heap Memory Usage Exceeds the Threshold	The HDFS DataNode heap memory usage is too high, which affects the data read/write performance of the HDFS.	95%
	DataNode Non-Heap Memory Usage Statistics	14019	DataNode Non-Heap Memory Usage Exceeds the Threshold	If the non-heap memory usage of the HDFS DataNode is too high, data read/write performance of HDFS will be affected.	90%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	NameNode GC Duration Statistics	14014	NameNode GC Duration Exceeds the Threshold	A long GC duration of the NameNode process may interrupt the services.	12000 ms
	DataNode GC Duration Statistics	14015	DataNode GC Duration Exceeds the Threshold	A long GC duration of the DataNode process may interrupt the services.	12000 ms
Hive	Hive SQL Execution Success Rate (Percentage)	16002	Hive SQL Execution Success Rate Is Lower Than the Threshold	The system configuration and performance cannot meet service processing requirements.	90.0%
	Background Thread Usage	16003	Background Thread Usage Exceeds the Threshold	There are too many background threads, so the newly submitted task cannot run in time.	90%
	Total GC Duration of MetaStore	16007	Hive GC Duration Exceeds the Threshold	If the GC duration exceeds the threshold, Hive data read and write are affected.	12000 ms
	Total GC Duration of HiveServer	16007	Hive GC Duration Exceeds the Threshold	If the GC duration exceeds the threshold, Hive data read and write are affected.	12000 ms
	Percentage of HDFS Space Used by Hive to the Available Space	16001	Hive Warehouse Space Usage Exceeds the Threshold	The system fails to write data, which causes data loss.	85.0%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	MetaStore Direct Memory Usage Statistics	16006	Direct Memory Usage of the Hive Process Exceeds the Threshold	When the direct memory usage of Hive is overhigh, the performance of Hive task operation is affected. In addition, a memory overflow may occur so that the Hive service is unavailable.	95%
	MetaStore Non-Heap Memory Usage Statistics	16008	Non-heap Memory Usage of the Hive Service Exceeds the Threshold	When the non-heap memory usage of Hive is overhigh, the performance of Hive task operation is affected. In addition, a memory overflow may occur so that the Hive service is unavailable.	95%
	MetaStore Heap Memory Usage Statistics	16005	Heap Memory Usage of the Hive Process Exceeds the Threshold	When the heap memory usage of Hive is overhigh, the performance of Hive task operation is affected. In addition, a memory overflow may occur so that the Hive service is unavailable.	95%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	HiveServer Direct Memory Usage Statistics	16006	Direct Memory Usage of the Hive Process Exceeds the Threshold	When the direct memory usage of Hive is overhigh, the performance of Hive task operation is affected. In addition, a memory overflow may occur so that the Hive service is unavailable.	95%
	HiveServer Non-Heap Memory Usage Statistics	16008	Non-heap Memory Usage of the Hive Service Exceeds the Threshold	When the non-heap memory usage of Hive is overhigh, the performance of Hive task operation is affected. In addition, a memory overflow may occur so that the Hive service is unavailable.	95%
	HiveServer Heap Memory Usage Statistics	16005	Heap Memory Usage of the Hive Process Exceeds the Threshold	When the heap memory usage of Hive is overhigh, the performance of Hive task operation is affected. In addition, a memory overflow may occur so that the Hive service is unavailable.	95%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Percentage of Sessions Connected to the HiveServer to Maximum Number of Sessions Allowed by the HiveServer	16000	Percentage of Sessions Connected to the HiveServer to Maximum Number Allowed Exceeds the Threshold	If a connection alarm is generated, too many sessions are connected to the HiveServer and new connections cannot be created.	90.0%
Kafka	Percentage of Partitions That Are Not Completely Synchronized	38006	Percentage of Kafka Partitions That Are Not Completely Synchronized Exceeds the Threshold	Too many Kafka partitions that are not completely synchronized affect service reliability. In addition, data may be lost when leaders are switched.	50%
	User Connection Usage on Broker	38011	User Connection Usage on Broker Exceeds the Threshold	If the number of connections of a user is excessive, the user cannot create new connections to the Broker.	80%
	Broker Disk Usage	38001	Insufficient Kafka Disk Capacity	Kafka data write operations fail.	80.0%
	Disk I/O Rate of a Broker	38009	Busy Broker Disk I/Os	The disk partition has frequent I/Os. Data may fail to be written to the Kafka topic for which the alarm is generated.	80%
	Broker GC Duration per Minute	38005	GC Duration of the Broker Process Exceeds the Threshold	A long GC duration of the Broker process may interrupt the services.	12000 ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Heap Memory Usage of Kafka	38002	Kafka Heap Memory Usage Exceeds the Threshold	If the available Kafka heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%
	Kafka Direct Memory Usage	38004	Kafka Direct Memory Usage Exceeds the Threshold	If the available direct memory of the Kafka service is insufficient, a memory overflow occurs and the service breaks down.	95%
Loader	Heap Memory Usage	23004	Loader Heap Memory Usage Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95%
	Direct Memory Usage Statistics	23006	Loader Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%
	Non-heap Memory Usage	23005	Loader Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80%
	Total GC Duration	23007	GC Duration of the Loader Process Exceeds the Threshold	Loader service response is slow.	12000 ms
MapReduce	GC Duration Statistics	18012	JobHistoryServer GC Duration Exceeds the Threshold	A long GC duration of the JobHistoryServer process may interrupt the services.	12000 ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	JobHistoryServer Direct Memory Usage Statistics	18015	JobHistoryServer Direct Memory Usage Exceeds the Threshold	If the available direct memory of the MapReduce service is insufficient, a memory overflow occurs and the service breaks down.	90%
	JobHistoryServer Non-Heap Memory Usage Statistics	18019	Non-Heap Memory Usage of JobHistoryServer Exceeds the Threshold	When the non-heap memory usage of MapReduce JobHistoryServer is overhigh, the performance of MapReduce task submission and operation is affected. In addition, a memory overflow may occur so that the MapReduce service is unavailable.	90%
	JobHistoryServer Heap Memory Usage Statistics	18009	Heap Memory Usage of JobHistoryServer Exceeds the Threshold	When the heap memory usage of MapReduce JobHistoryServer is overhigh, the performance of MapReduce log archiving is affected. In addition, a memory overflow may occur so that the Yarn service is unavailable.	95%
Oozie	Heap Memory Usage	17004	Oozie Heap Memory Usage Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95.0%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Direct Memory Usage	17006	Oozie Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%
	Non-heap Memory Usage	17005	Oozie Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80%
	Total GC Duration	17007	GC Duration of the Oozie Process Exceeds the Threshold	Oozie responds slowly when it is used to submit tasks.	12000 ms
Spark2x	JDBCServer2x Heap Memory Usage Statistics	43010	Heap Memory Usage of the JDBCServer2x Process Exceeds the Threshold	If available JDBCServe2x process heap memory is insufficient, a memory overflow occurs and the service breaks down	95%
	JDBCServer2x Direct Memory Usage Statistics	43012	Direct Heap Memory Usage of the JDBCServer2x Process Exceeds the Threshold	If the available JDBCServer2x Process direct heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%
	JDBCServer2x Non-Heap Memory Usage Statistics	43011	Non-Heap Memory Usage of the JDBCServer2x Process Exceeds the Threshold	If the available JDBCServer2x Process non-heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	JobHistory2x Direct Memory Usage Statistics	43008	Direct Memory Usage of the JobHistory2x Process Exceeds the Threshold	If the available JobHistory2x Process directmemory is insufficient, a memory overflow occurs and the service breaks down.	95%
	JobHistory2x Non-Heap Memory Usage Statistics	43007	Non-Heap Memory Usage of the JobHistory2x Process Exceeds the Threshold	If the available JobHistory2x Process non-heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%
	JobHistory2x Heap Memory Usage Statistics	43006	Heap Memory Usage of the JobHistory2x Process Exceeds the Threshold	If the available JobHistory2x Process heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%
	IndexServer2x Direct Memory Usage Statistics	43021	Direct Memory Usage of the IndexServer2x Process Exceeds the Threshold	If the available IndexServer2x process direct memory is insufficient, a memory overflow occurs and the service breaks down.	95%
	IndexServer2x Heap Memory Usage Statistics	43019	Heap Memory Usage of the IndexServer2x Process Exceeds the Threshold	If the available IndexServer2x process heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	IndexServer2x Non-Heap Memory Usage Statistics	43020	Non-Heap Memory Usage of the IndexServer2x Process Exceeds the Threshold	If the available IndexServer2x process non-heap memory is insufficient, a memory overflow occurs and the service breaks down.	95%
	Full GC Number of JDBCServer2x	43017	JDBCServer2x Process Full GC Number Exceeds the Threshold	The performance of the JDBCServer2x process is affected, or even the JDBCServer2x process is unavailable.	12
	Full GC Number of JobHistory2x	43018	JobHistory2x Process Full GC Number Exceeds the Threshold	The performance of the JobHistory2x process is affected, or even the JobHistory2x process is unavailable.	12
	Full GC Number of IndexServer2x	43023	IndexServer2x Process Full GC Number Exceeds the Threshold	If the GC number exceeds the threshold, IndexServer2x maybe run in low performance or even unavailable.	12
	Total GC Duration (in Milliseconds) of JDBCServer2x	43013	JDBCServer2x Process GC Duration Exceeds the Threshold	If the GC duration exceeds the threshold, JDBCServer2x maybe run in low performance.	12000 ms
	Total GC Duration (in Milliseconds) of JobHistory2x	43009	JobHistory2x Process GC Duration Exceeds the Threshold	If the GC duration exceeds the threshold, JobHistory2x may run in low performance.	12000 ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Total GC Duration (in Milliseconds) of IndexServer2x	43022	IndexServer2x Process GC Duration Exceeds the Threshold	If the GC duration exceeds the threshold, IndexServer2x may run in low performance or even unavailable.	12000 ms
Storm	Number of Available Supervisors	26052	Number of Available Supervisors of the Storm Service Is Less Than the Threshold	Existing tasks in the cluster cannot be performed. The cluster can receive new Storm tasks, but cannot perform these tasks.	1
	Slot Usage	26053	Storm Slot Usage Exceeds the Threshold	New Storm tasks cannot be performed.	80.0%
	Nimbus Heap Memory Usage	26054	Nimbus Heap Memory Usage Exceeds the Threshold	When the heap memory usage of Storm Nimbus is overhigh, frequent GCs occur. In addition, a memory overflow may occur so that the Yarn service is unavailable.	80%
Yarn	NodeManager Direct Memory Usage Statistics	18014	NodeManager Direct Memory Usage Exceeds the Threshold	If the available direct memory of NodeManager is insufficient, a memory overflow occurs and the service breaks down.	90%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	NodeManager Heap Memory Usage Statistics	18018	NodeManager Heap Memory Usage Exceeds the Threshold	When the heap memory usage of Yarn NodeManager is overhigh, the performance of Yarn task submission and operation is affected. In addition, a memory overflow may occur so that the Yarn service is unavailable.	95%
	NodeManager Non-Heap Memory Usage Statistics	18017	NodeManager Non-heap Memory Usage Exceeds the Threshold	When the heap memory usage of Yarn NodeManager is overhigh, the performance of Yarn task submission and operation is affected. In addition, a memory overflow may occur so that the Yarn service is unavailable.	90%
	ResourceManager Direct Memory Usage Statistics	18013	ResourceManager Direct Memory Usage Exceeds the Threshold	If the available direct memory of ResourceManager is insufficient, a memory overflow occurs and the service breaks down.	90%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	ResourceManager Heap Memory Usage Statistics	18008	ResourceManager Heap Memory Usage Exceeds the Threshold	When the heap memory usage of Yarn ResourceManager is overhigh, the performance of Yarn task submission and operation is affected. In addition, a memory overflow may occur so that the Yarn service is unavailable.	95%
	ResourceManager Non-Heap Memory Usage Statistics	18016	ResourceManager Non-Heap Memory Usage Exceeds the Threshold	When the non-heap memory usage of Yarn ResourceManager is overhigh, the performance of Yarn task submission and operation is affected. In addition, a memory overflow may occur so that the Yarn service is unavailable.	90%
	NodeManager GC Duration Statistics	18011	NodeManager GC Duration Exceeds the Threshold	A long GC duration of the NodeManager process may interrupt the services.	12000 ms
	ResourceManager GC Duration Statistics	18010	ResourceManager GC Duration Exceeds the Threshold	A long GC duration of the ResourceManager process may interrupt the services.	12000 ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Number of Failed Tasks in the Root Queue	18026	Number of Failed Yarn Tasks Exceeds the Threshold	A large number of application tasks fail to be executed. Failed tasks need to be submitted again.	50
	Terminated Applications of the Root Queue	18025	Number of Terminated Yarn Tasks Exceeds the Threshold	A large number of application tasks are forcibly stopped.	50
	Pending Memory	18024	Pending Yarn Memory Usage Exceeds the Threshold	It takes long time to end an application. A new application cannot run after submission.	83886080MB
	Pending Tasks	18023	Number of Pending Yarn Tasks Exceeds the Threshold	It takes long time to end an application. A new application cannot run for a long time after submission.	60
ZooKeeper	ZooKeeper Connections Usage	13001	Available ZooKeeper Connections Are Insufficient	Available ZooKeeper connections are insufficient. When the connection usage reaches 100%, external connections cannot be handled.	80%
	ZooKeeper Heap Memory Usage	13004	ZooKeeper Heap Memory Usage Exceeds the Threshold	If the available ZooKeeper memory is insufficient, a memory overflow occurs and the service breaks down.	95%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	ZooKeeper Direct Memory Usage	13002	ZooKeeper Direct Memory Usage Exceeds the Threshold	If the available ZooKeeper memory is insufficient, a memory overflow occurs and the service breaks down.	80%
	ZooKeeper GC Duration per Minute	13003	GC Duration of the ZooKeeper Process Exceeds the Threshold	A long GC duration of the ZooKeeper process may interrupt the services.	12000 ms
Ranger	UserSync GC Duration	45284	UserSync GC Duration Exceeds the Threshold	UserSync responds slowly.	12000 ms
	PolicySync GC Duration	45292	PolicySync GC Duration Exceeds the Threshold	PolicySync responds slowly.	12000 ms
	RangerAdmin GC Duration	45280	RangerAdmin GC Duration Exceeds the Threshold	RangerAdmin responds slowly.	12000 ms
	TagSync GC Duration	45288	TagSync GC Duration Exceeds the Threshold	TagSync responds slowly.	12000 ms
	UserSync Non-Heap Memory Usage	45283	UserSync Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80.0%
	UserSync Direct Memory Usage	45282	UserSync Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	UserSync Heap Memory Usage	45281	UserSync Heap Memory Usage Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95.0%
	PolicySync Direct Memory Usage	45290	PolicySync Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%
	PolicySync Heap Memory Usage	45289	PolicySync Heap Memory Usage Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95.0%
	PolicySync Non-Heap Memory Usage	45291	PolicySync Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80.0%
	RangerAdmin Non-Heap Memory Usage	45279	RangerAdmin Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80.0%
	RangerAdmin Heap Memory Usage	45277	RangerAdmin Heap Memory Usage Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95.0%
	RangerAdmin Direct Memory Usage	45278	RangerAdmin Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	TagSync Direct Memory Usage	45286	TagSync Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	80.0%
	TagSync Non-Heap Memory Usage	45287	TagSync Non-Heap Memory Usage Exceeds the Threshold	Non-heap memory overflow may cause service breakdown.	80.0%
	TagSync Heap Memory Usage	45285	TagSync Heap Memory Usage Exceeds the Threshold	Heap memory overflow may cause service breakdown.	95.0%
ClickHouse	Clickhouse Service Quantity Quota Usage in ZooKeeper	45426	ClickHouse Service Quantity Quota Usage in ZooKeeper Exceeds the Threshold	After the ZooKeeper quantity quota of the ClickHouse service exceeds the threshold, you cannot perform cluster operations on the ClickHouse service on FusionInsight Manager. As a result, the ClickHouse service cannot be used.	90%

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	ClickHouse Service Capacity Quota Usage in ZooKeeper	45427	ClickHouse Service Capacity Quota Usage in ZooKeeper Exceeds the Threshold	After the ZooKeeper capacity quota of the ClickHouse service exceeds the threshold, you cannot perform cluster operations on the ClickHouse service on FusionInsight Manager. As a result, the ClickHouse service cannot be used.	90%
IoTDB	Maximum Merge (Intra-Space Merge) Latency	45594	IoTDBServer Intra-Space Merge Duration Exceeds the Threshold	Data write is blocked and the write operation performance is affected.	300000 ms
	Maximum Merge (Flush) Latency	45593	IoTDBServer Flush Execution Duration Exceeds the Threshold	Data write is blocked and the write operation performance is affected.	300000 ms
	Maximum Merge (Cross-Space Merge) Latency	45595	IoTDBServer Cross-Space Merge Duration Exceeds the Threshold	Data write is blocked and the write operation performance is affected.	300000 ms
	Maximum RPC (executeStatement) Latency	45592	IoTDBServer RPC Execution Duration Exceeds the Threshold	Running performance of the IoTDBServer process is affected.	10000s
	Total GC Duration of IoTDBServer	45587	IoTDBServer GC Duration Exceeds the Threshold	A long GC duration of the IoTDBServer process may interrupt the services.	12000 ms

Service	Metric	ID	Alarm Name	Impact on System	Default Threshold
	Total GC Duration of ConfigNode	45590	ConfigNode GC Duration Exceeds the Threshold	A long GC duration of the ConfigNode process may interrupt services.	12000 ms
	IoTDBServer Heap Memory Usage	45586	IoTDBServer Heap Memory Usage Exceeds the Threshold	If the available IoTDBServer process heap memory is insufficient, a memory overflow occurs and the service breaks down.	90%
	IoTDBServer Direct Memory Usage	45588	IoTDBServer Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause service breakdown.	90%
	ConfigNode Heap Memory Usage	45589	ConfigNode Heap Memory Usage Exceeds the Threshold	If the available ConfigNode process heap memory is insufficient, a memory overflow occurs and the service breaks down.	90%
	ConfigNode Direct Memory Usage	45591	ConfigNode Direct Memory Usage Exceeds the Threshold	Direct memory overflow may cause the IoTDB instance to be unavailable.	90%

12 MRS Component Application Development

12.1 HBase Application Development

HBase is a column-based distributed storage system that features high reliability, performance, and scalability. It is designed to eliminate the limitations of relational databases in processing massive amounts of data.

Application scenarios of HBase have the following features:

- Massive data processing (higher than the TB or PB level)
- High throughput
- Highly efficient random read of massive data
- Excellent scalability
- Concurrent processing of structured and non-structured data

MRS provides sample application development projects based on HBase. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then conduct building and commissioning locally. In this sample project, you can create HBase tables, insert data, create indexes, and delete tables in the MRS cluster.

Creating an MRS HBase Cluster

1. Create and purchase an MRS cluster that contains HBase. For details, see **Buying a Custom Cluster**.

   NOTE

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

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

Figure 12-1 Cluster purchased

Preparing the Application Development Configuration File

Step 1 After the cluster is created, log in to FusionInsight Manager and create a cluster user for security authentication of the sample project.

1. Choose **System** > **Permission** > **User**. In the right pane, click **Create**. On the displayed page, create a human-machine user, for example, **developuser**.

   Add the **hadoop** user group to **User Group**.

   After the user is created, log in to FusionInsight Manager as **developuser** and change the initial password as prompted.
2. Log in to the Ranger web UI as the Ranger administrator **rangeradmin**.

   The default password of user **rangeradmin** is **Rangeradmin@123**. For details, see **User Account List**.
3. On the Ranger homepage, click the component plug-in name in the **HBASE** area, for example, **HBase**.
4. Click in the **Action** column of the row containing the **all - table, column-family, column** policy.
5. In the **Allow Conditions** area, add an allow condition. Select the created user for **Select User**, and select **Select/Deselect All** for **Permissions**.
6. Click **Save**.

Step 2 Log in to FusionInsight Manager as user **admin** and choose **System** > **Permission** > **User**. In the **Operation** column of **developuser**, choose **More** > **Download Authentication Credential**. Save the file and decompress it to obtain the **user.keytab** and **krb5.conf** files of the user.

Step 3 Choose **Cluster**. On the **Dashboard** tab, click **More** and select **Download Client**. In the dialog box that is displayed, set **Select Client Type** to **Configuration Files Only** and click **OK**. After the client package is generated, download the package as prompted and decompress it.

For example, if the client configuration file package is **FusionInsight_Cluster_1_Services_Client.tar**, decompress it to obtain **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar**. Then, continue to decompress this file.

1. Go to the **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles\HBase\config** directory and obtain the configuration files listed in **Table 12-1**.

Table 12-1 Configuration files

Configuration File	Description
core-site.xml	Configures Hadoop Core parameters.

Configuration File	Description
hbase-site.xml	Configures HBase parameters.
hdfs-site.xml	Configures HDFS parameters.

2. Copy all content from the **hosts** file in the decompression directory to your local **hosts** file. Ensure that the local PC can communicate with the hosts listed in the **hosts** file in the decompression directory.

   NOTE

   - In this practice, ensure that the local environment can communicate with the network plane where the MRS cluster resides. Generally, you can access the MRS cluster via an EIP..
   - If the local environment cannot communicate with nodes in the MRS cluster, you can build the sample project first and upload the JAR package to the cluster to run. .
   - **C:\WINDOWS\system32\drivers\etc\hosts** is an example directory in a Windows environment for storing the local **hosts** file.

----End

Obtaining the Sample Project

Step 1 Obtain the sample project from Huawei Mirrors.

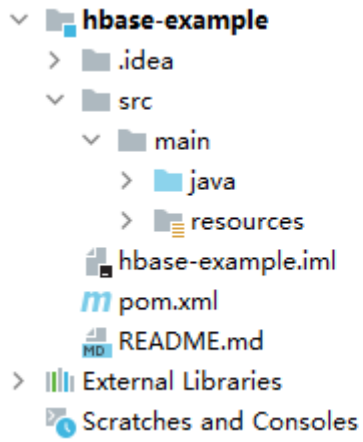
Download the Maven project source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **hbase-example**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.0/src/hbase-examples/hbase-example**.

Step 2 Use IDEA to import the sample project and wait for the Maven project to download the dependency packages. For details, see **Configuring and Importing Sample Projects**.

Figure 12-2 HBase sample project



After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages.

Step 3 Place the cluster configuration files and user authentication credentials obtained in **Preparing the Application Development Configuration File** into the **../src/main/resources/conf** directory of the sample project.

Step 4 In the **TestMain** class of the **com.huawei.bigdata.hbase.examples** package, change **userName** to the actual username, for example, **developuser**.

```
private static void login() throws IOException {
    if (User.isHBaseSecurityEnabled(conf)) {
        userName = "developuser";

        //In Windows environment
        String userdir = TestMain.class.getClassLoader().getResource("conf").getPath() + File.separator;
        //In Linux environment
        //String userdir = System.getProperty("user.dir") + File.separator + "conf" + File.separator;

        LoginUtil.setJaasConf(ZOOKEEPER_DEFAULT_LOGIN_CONTEXT_NAME, userName, userKeytabFile);
        LoginUtil.login(userName, userKeytabFile, krb5File, conf);
    }
}
```

Assume that you are developing an application to manage information about users of service A in an enterprise. The operation process is as follows.

No.	Step
1	Create a table based on existing information.
2	Import user data.
3	Add the **Education Information** column family and add the education backgrounds and titles of users to the user information table.
4	Query usernames and addresses by user ID.
5	Execute queries by username.
6	To improve query performance, create or delete secondary indexes.
7	Deregister users and delete user data from the user information table.
8	Delete the user information table after service A ends.

For example, the following code snippet executes the **testCreateTable** method in the **HBaseSample** class of the **com.huawei.bigdata.hbase.examples** package to create a user information table..

```
public void testCreateTable() {
        LOG.info("Entering testCreateTable.");
        TableDescriptorBuilder htd = TableDescriptorBuilder.newBuilder(tableName); //Create a table descriptor.
        ColumnFamilyDescriptorBuilder hcd = ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("info")); //Create a column family descriptor.
        hcd.setDataBlockEncoding(DataBlockEncoding.FAST_DIFF); //Set the encoding algorithm. HBase provides DIFF, FAST_DIFF, and PREFIX encoding algorithms.
        hcd.setCompressionType(Compression.Algorithm.SNAPPY);
```



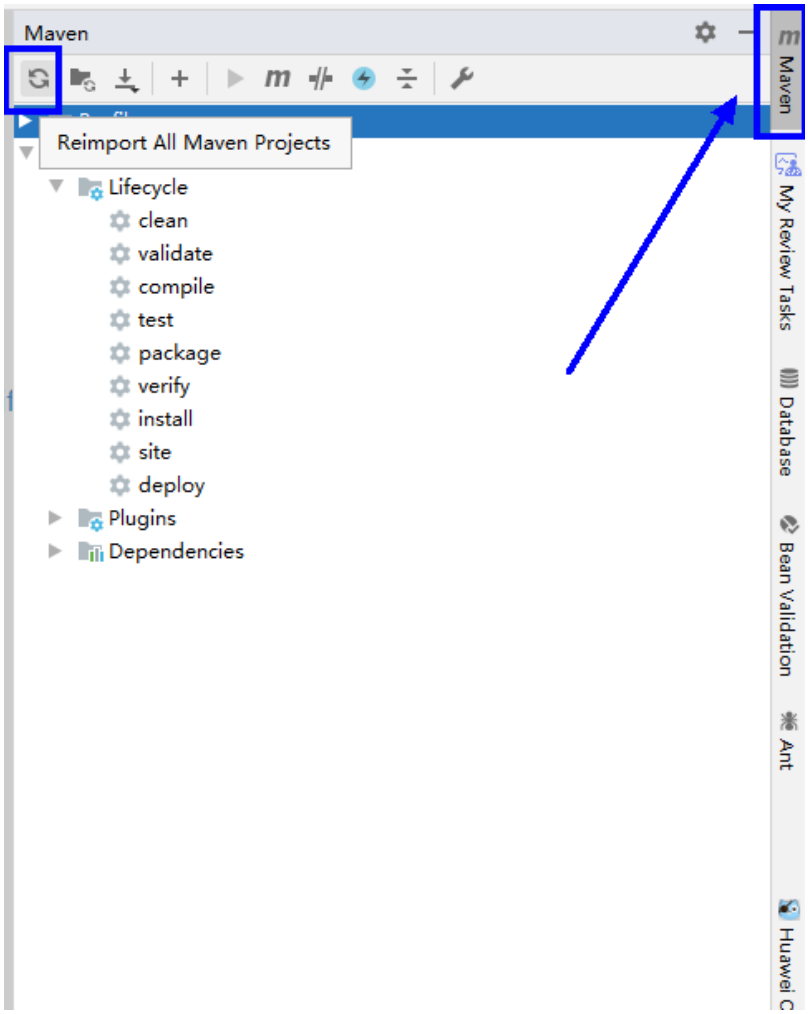
```
            htd.setColumnFamily(hcd.build());  //Add the column family descriptor to the table descriptor.
            Admin admin = null;
            try {
              admin = conn.getAdmin(); //Obtain the Admin object, which allows you to create a table, create
a column family, check whether the table exists, change the table structure and column family structure,
and delete the table.
              if (!admin.tableExists(tableName)) {
                LOG.info("Creating table...");
                admin.createTable(htd.build());//Call the createTable method of Admin.
                LOG.info(admin.getClusterMetrics().toString());
                LOG.info(admin.listNamespaceDescriptors().toString());
                LOG.info("Table created successfully.");
              } else {
                LOG.warn("table already exists");
              }
            } catch (IOException e) {
                LOG.error("Create table failed " ,e);
            } finally {
              if (admin != null) {
                try {
                  admin.close();
                } catch (IOException e) {
                  LOG.error("Failed to close admin " ,e);
                }
              }
            }
            LOG.info("Exiting testCreateTable.");
 }
```

----End

Building and Running the Application

Step 1 Click **Reimport All Maven Projects** in the Maven window on the right of IDEA to load the Maven project dependencies.

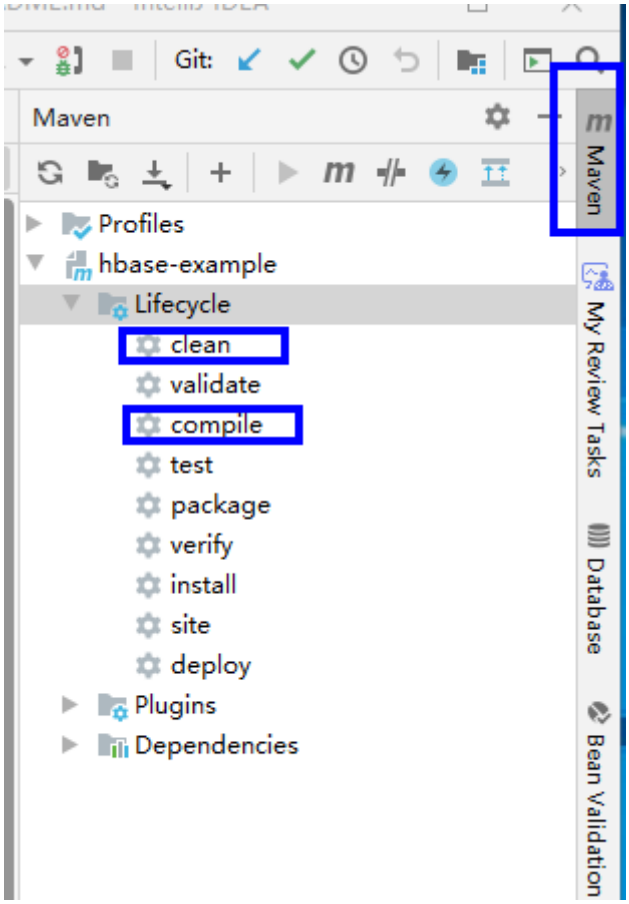
Figure 12-3 Loading a sample project



Step 2 Build the application.

1. Choose **Maven**, locate the target project name, and double-click **clean** under **Lifecycle** to run the **clean** command of Maven.
2. Choose **Maven**, locate the target project name, and double-click **compile** under **Lifecycle** to run the **compile** command of Maven.

Figure 12-4 clean and compile of Maven



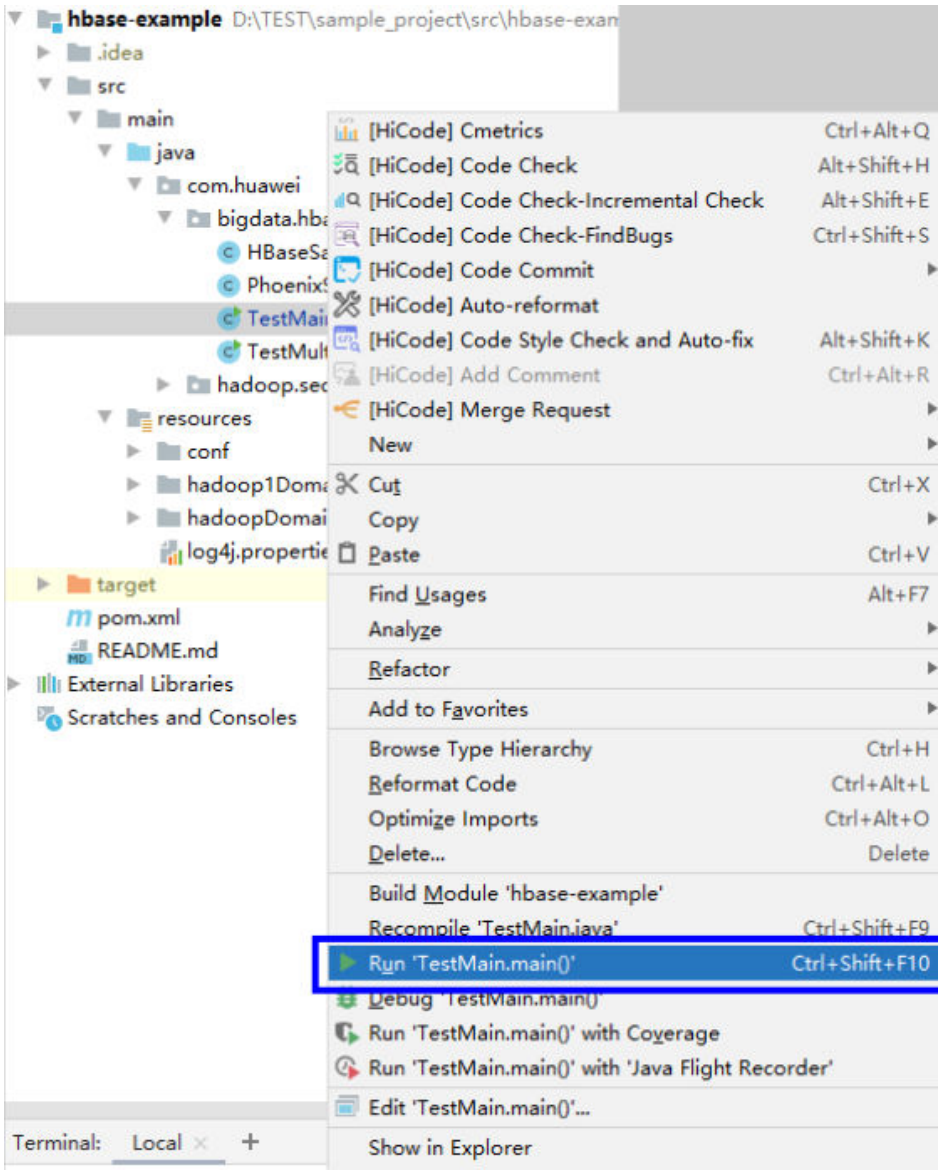
After the building is complete, message "Build Success" is displayed and the **target** directory is generated.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  21.276 s
[INFO] Finished at: 2023-05-05T14:36:39+08:00
[INFO] ------------------------------------------------------------------------
```

Step 3 Run the application.

Right-click the **TestMain.java** file and choose **Run 'TestMain.main()**.

Figure 12-5 Running the application



Step 4 Check the output information after running the **hbase-example** sample. The following information indicates that related table operations are successfully executed:

```
...
2023-05-05 15:05:27,050 INFO  [main] examples.HBaseSample: Table created successfully.
2023-05-05 15:05:27,050 INFO  [main] examples.HBaseSample: Exiting testCreateTable.
2023-05-05 15:05:27,050 INFO  [main] examples.HBaseSample: Entering testMultiSplit.
2023-05-05 15:05:31,171 INFO  [main] client.HBaseAdmin: Operation: MULTI_SPLIT_REGION, Table Name:
default:hbase_sample_table, procId: 21 completed
2023-05-05 15:05:31,171 INFO  [main] examples.HBaseSample: MultiSplit successfully.
2023-05-05 15:05:31,172 INFO  [main] examples.HBaseSample: Exiting testMultiSplit.
2023-05-05 15:05:31,172 INFO  [main] examples.HBaseSample: Entering testPut.
2023-05-05 15:05:32,862 INFO  [main] examples.HBaseSample: Put successfully.
2023-05-05 15:05:32,862 INFO  [main] examples.HBaseSample: Exiting testPut.
2023-05-05 15:05:32,862 INFO  [main] examples.HBaseSample: Entering createIndex.
2023-05-05 15:05:36,627 INFO  [main] examples.HBaseSample: Create index successfully.
2023-05-05 15:05:36,627 INFO  [main] examples.HBaseSample: Exiting createIndex.
2023-05-05 15:05:36,627 INFO  [main] examples.HBaseSample: Entering createIndex.
2023-05-05 15:05:37,912 INFO  [main] examples.HBaseSample: Successfully enable indices [index_name]
of the table hbase_sample_table
```



```
2023-05-05 15:05:37,912 INFO  [main] examples.HBaseSample: Entering testScanDataByIndex.
2023-05-05 15:05:37,915 INFO  [main] examples.HBaseSample: Scan indexed data.
2023-05-05 15:05:39,939 INFO  [main] examples.HBaseSample: Scan data by index successfully.
2023-05-05 15:05:39,939 INFO  [main] examples.HBaseSample: Exiting testScanDataByIndex.
2023-05-05 15:05:39,941 INFO  [main] examples.HBaseSample: Entering testModifyTable.
2023-05-05 15:05:40,191 INFO  [main] client.HBaseAdmin: Started disable of hbase_sample_table
2023-05-05 15:05:41,322 INFO  [main] client.HBaseAdmin: Operation: DISABLE, Table Name:
default:hbase_sample_table, procId: 53 completed
2023-05-05 15:05:42,230 INFO  [main] client.HBaseAdmin: Started enable of hbase_sample_table
2023-05-05 15:05:43,187 INFO  [main] client.HBaseAdmin: Operation: ENABLE, Table Name:
default:hbase_sample_table, procId: 65 completed
2023-05-05 15:05:43,187 INFO  [main] examples.HBaseSample: Modify table successfully.
2023-05-05 15:05:43,187 INFO  [main] examples.HBaseSample: Exiting testModifyTable.
2023-05-05 15:05:43,187 INFO  [main] examples.HBaseSample: Entering testGet.
2023-05-05 15:05:43,278 INFO  [main] examples.HBaseSample: 012005000201:info,address,Shenzhen,
Guangdong
2023-05-05 15:05:43,279 INFO  [main] examples.HBaseSample: 012005000201:info,name,Zhang San
2023-05-05 15:05:43,279 INFO  [main] examples.HBaseSample: Get data successfully.
2023-05-05 15:05:43,279 INFO  [main] examples.HBaseSample: Exiting testGet.
2023-05-05 15:05:43,279 INFO  [main] examples.HBaseSample: Entering testScanData.
2023-05-05 15:05:43,576 INFO  [main] examples.HBaseSample: 012005000201:info,name,Zhang San
2023-05-05 15:05:43,576 INFO  [main] examples.HBaseSample: 012005000202:info,name,Li Wanting
2023-05-05 15:05:43,577 INFO  [main] examples.HBaseSample: 012005000203:info,name,Wang Ming
2023-05-05 15:05:43,577 INFO  [main] examples.HBaseSample: 012005000204:info,name,Li Gang
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: 012005000205:info,name,Zhao Enru
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: 012005000206:info,name,Chen Long
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: 012005000207:info,name,Zhou Wei
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: 012005000208:info,name,Yang Yiwen
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: 012005000209:info,name,Xu Bing
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: 012005000210:info,name,Xiao Kai
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: Scan data successfully.
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: Exiting testScanData.
2023-05-05 15:05:43,578 INFO  [main] examples.HBaseSample: Entering testSingleColumnValueFilter.
2023-05-05 15:05:43,883 INFO  [main] examples.HBaseSample: Single column value filter successfully.
2023-05-05 15:05:43,883 INFO  [main] examples.HBaseSample: Exiting testSingleColumnValueFilter.
2023-05-05 15:05:43,884 INFO  [main] examples.HBaseSample: Entering testFilterList.
2023-05-05 15:05:44,388 INFO  [main] examples.HBaseSample: 012005000201:info,name,Zhang San
2023-05-05 15:05:44,388 INFO  [main] examples.HBaseSample: 012005000202:info,name,Li Wanting
2023-05-05 15:05:44,388 INFO  [main] examples.HBaseSample: 012005000203:info,name,Wang Ming
2023-05-05 15:05:44,388 INFO  [main] examples.HBaseSample: 012005000204:info,name,Li Gang
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: 012005000205:info,name,Zhao Enru
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: 012005000206:info,name,Chen Long
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: 012005000207:info,name,Zhou Wei
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: 012005000208:info,name,Yang Yiwen
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: 012005000209:info,name,Xu Bing
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: 012005000210:info,name,Xiao Kai
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: Filter list successfully.
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: Exiting testFilterList.
2023-05-05 15:05:44,389 INFO  [main] examples.HBaseSample: Entering testDelete.
2023-05-05 15:05:44,586 INFO  [main] examples.HBaseSample: Delete table successfully.
2023-05-05 15:05:44,586 INFO  [main] examples.HBaseSample: Exiting testDelete.
2023-05-05 15:05:44,586 INFO  [main] examples.HBaseSample: Entering disableIndex.
2023-05-05 15:05:45,819 INFO  [main] examples.HBaseSample: Successfully disable indices [index_name]
of the table hbase_sample_table
2023-05-05 15:05:45,819 INFO  [main] examples.HBaseSample: Entering dropIndex.
2023-05-05 15:05:48,084 INFO  [main] examples.HBaseSample: Drop index successfully.
2023-05-05 15:05:48,084 INFO  [main] examples.HBaseSample: Exiting dropIndex.
2023-05-05 15:05:48,084 INFO  [main] examples.HBaseSample: Entering dropTable.
2023-05-05 15:05:48,237 INFO  [main] client.HBaseAdmin: Started disable of hbase_sample_table
2023-05-05 15:05:49,543 INFO  [main] client.HBaseAdmin: Operation: DISABLE, Table Name:
default:hbase_sample_table, procId: 95 completed
2023-05-05 15:05:50,645 INFO  [main] client.HBaseAdmin: Operation: DELETE, Table Name:
default:hbase_sample_table, procId: 106 completed
2023-05-05 15:05:50,645 INFO  [main] examples.HBaseSample: Drop table successfully.
2023-05-05 15:05:50,645 INFO  [main] examples.HBaseSample: Exiting dropTable.
2023-05-05 15:05:50,646 INFO  [main] client.ConnectionImplementation: Closing master protocol:
MasterService
2023-05-05 15:05:50,652 INFO  [main] client.ConnectionImplementation: Connection has been closed by
main.
```



```
2023-05-05 15:05:50,654 INFO  [main] hbase.ChoreService: Chore service for: AsyncConn Chore Service had
[[ScheduledChore: Name: RefreshCredentials Period: 30000 Unit: MILLISECONDS]] on shutdown
2023-05-05 15:05:50,655 INFO  [main] examples.TestMain: ----------finish HBase ----------
...
```

----End

12.2 HDFS Application Development

Hadoop Distribute File System (HDFS) is a distributed file system that runs on universal hardware. It features high fault tolerance and supports high-throughput data access. It is suitable for processing large-scale data sets.

HDFS is suitable for the following application scenarios:

- Processing of massive amounts of data (TB or PB level and larger)
- Scenarios that require high throughput
- Scenarios that require high reliability
- Scenarios that require excellent scalability

MRS provides sample application development projects based on HBase. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then compile and debug the code locally. In this sample project, you can create HDFS directories, and write, read, and delete files.

Creating an MRS Hadoop Cluster

1. Create and purchase an MRS cluster that contains Hadoop. For details, see **Buying a Custom Cluster**.

   NOTE

   In this practice, an MRS 3.2.0-LTS.1 cluster, with Hadoop installed and with Kerberos authentication enabled, is used as an example.

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

Preparing the Application Development Configuration File

Step 1 Log in to FusionInsight Manager and create a cluster user for security authentication of the sample project.

Choose **System** > **Permission** > **User**. On the displayed page, click **Create**. On the displayed page, create a machine-machine user, for example, **developuser**.

Add the **hadoop** user group to **User Group**.

Step 2 Choose **System** > **Permission** > **User**. In the **Operation** column of **developuser**, choose **More** > **Download Authentication Credential**. Save the file and decompress it to obtain the **user.keytab** and **krb5.conf** files of the user.

Step 3 Choose **Cluster**. On the **Dashboard** tab, click **More** and select **Download Client**. In the dialog box that is displayed, set **Select Client Type** to **Configuration Files Only** and click **OK**. After the client package is generated, download the package as prompted and decompress it.

For example, if the client configuration file package is **FusionInsight_Cluster_1_Services_Client.tar**, decompress it to obtain

FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar. Then, continue to decompress this file.

1. Go to the **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles\HDFS\config** directory and obtain the configuration files listed in **Table 12-2**.

   Table 12-2 File

File	Description
core-site.xml	Hadoop Core parameters
hdfs-site.xml	HDFS parameters

2. Copy all content from the **hosts** file in the decompression directory to your local **hosts** file. Ensure that the local PC can communicate with the hosts listed in the **hosts** file in the decompression directory.

   NOTE

   - In this practice, ensure that the local environment can communicate with the network plane where the MRS cluster is deployed. Generally, you can access the MRS cluster via an EIP.
   - If the local environment cannot communicate with nodes in the MRS cluster, you can build the sample project first and upload the JAR package to the cluster to run.
   - **C:\WINDOWS\system32\drivers\etc\hosts** is an example directory in a Windows environment for storing the local **hosts** file.

----End

Obtaining the Sample Project

Step 1 Obtain the sample project from Huawei Mirrors.

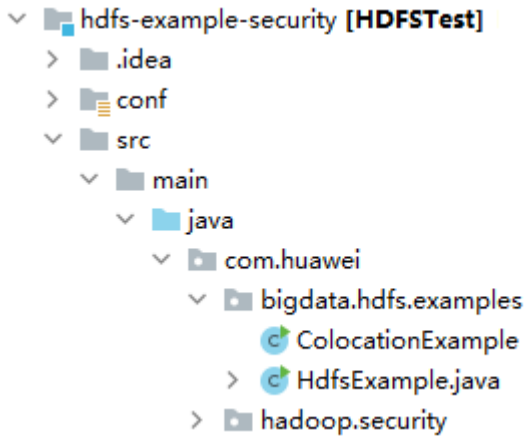
Download the source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **hdfs-example-security**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.2.0.1/src/hdfs-example-security**.

Step 2 Use IDEA to import the sample project and wait for the Maven project to download the dependency packages. For details, see **Configuring and Importing Sample Projects**.

Figure 12-6 HDFS sample project



After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages.

Step 3 Place the cluster configuration files and user authentication credentials obtained in **Preparing the Application Development Configuration File** into the **conf** directory of the sample project.

Step 4 Use the required authentication code for the HDFS sample project. Generally, there are security authentication and ZooKeeper authentication.

In this example, you do not need to access HBase or ZooKeeper. Only the basic security authentication code is required.

In the **HdfsExample** class of the **com.huawei.bigdata.hdfs.examples** package, change **PRNCIPAL_NAME** to the username you are using, for example, **developuser**.

```
private static final String PATH_TO_HDFS_SITE_XML = System.getProperty("user.dir") + File.separator +
"conf"
        + File.separator + "hdfs-site.xml";
private static final String PATH_TO_CORE_SITE_XML =  System.getProperty("user.dir") + File.separator +
"conf"
        + File.separator + "core-site.xml";
private static final String PRNCIPAL_NAME = "developuser";
private static final String PATH_TO_KEYTAB = System.getProperty("user.dir") + File.separator + "conf"
        + File.separator + "user.keytab";
private static final String PATH_TO_KRB5_CONF = System.getProperty("user.dir") + File.separator + "conf"
        + File.separator + "krb5.conf";
...
```

In this sample project, the development roadmap based on the service requirement is as follows.

The following example describes how to read, write, delete the **/user/hdfs-examples/test.txt** file in HDFS.

1. Pass the cluster security authentication.
2. Create a FileSystem object: **fSystem**
3. Call the **mkdir** API in **fSystem** to create a directory.
4. Call **create** on **fSystem** to create an FSDataOutputStream object **out**. Write data into **out** by calling **write**.
5. Call **append** on **fSystem** to create an FSDataOutputStream object **out**. Append data into **out** by calling **write**.

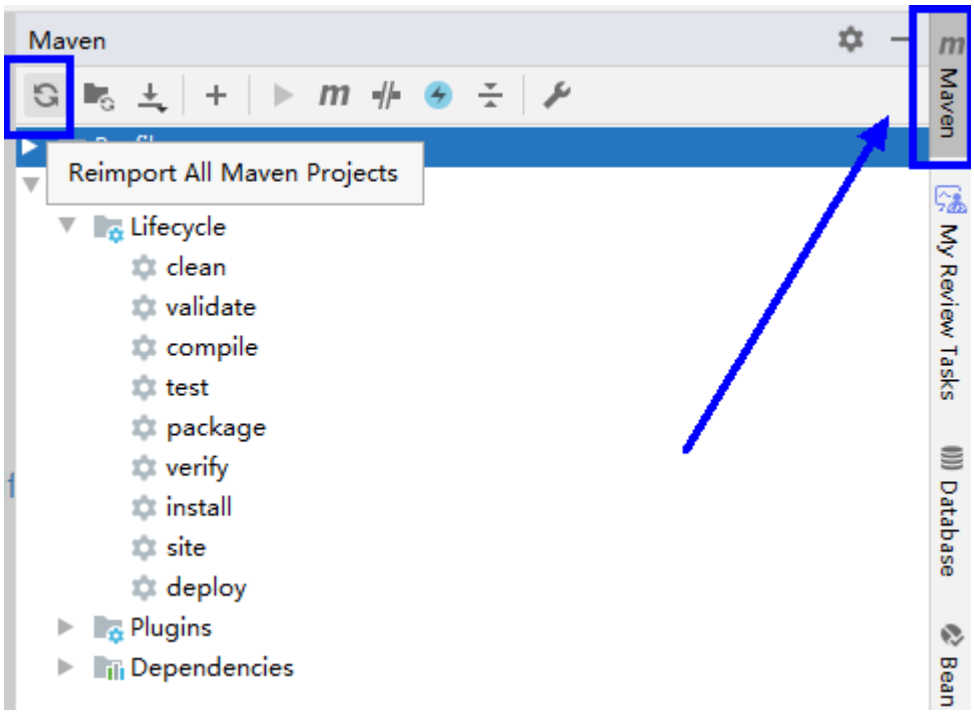
6. Call **open** on **fSystem** to create an FSDataInputStream object **in**. Read files of **in** by calling **read**.
7. Call **delete** on **fSystem** to delete a file.
8. Call **delete** on **fSystem** to delete a folder.

----End

Building and Running the Application

Step 1 Click **Reimport All Maven Projects** in the Maven window on the right of IDEA to load the Maven project dependencies.

Figure 12-7 Loading a sample project



Step 2 Compile and run the application.

1. Choose **Maven**, locate the target project name, and double-click **clean** under **Lifecycle** to run the **clean** command of Maven.
2. Choose **Maven**, locate the target project name, and double-click **compile** under **Lifecycle** to run the **compile** command of Maven.

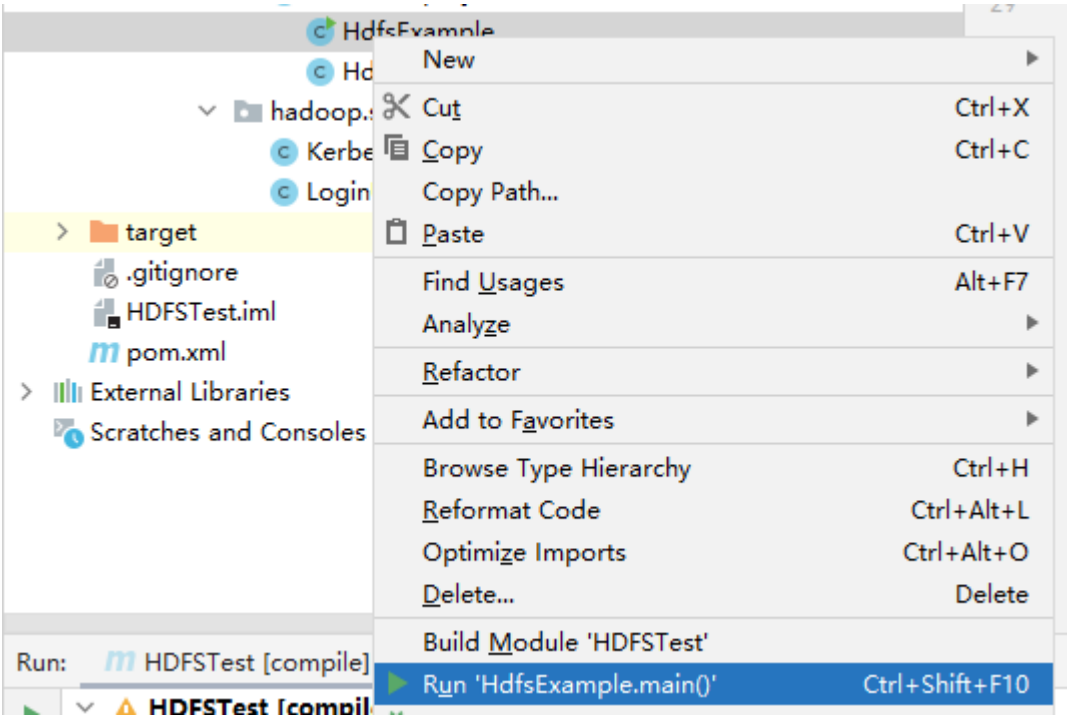
After the building is complete, message "Build Success" is displayed and the **target** directory is generated.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  21.276 s
[INFO] Finished at: 2023-05-05T14:36:39+08:00
[INFO] ------------------------------------------------------------------------
```

Step 3 Run the application.

Right-click the **HdfsExample.java** file and choose **Run 'HdfsExample.main()'** from the shortcut menu.

Figure 12-8 Running the application



Step 4 Check the output information after running the sample. The following information indicates that related file operations are successfully executed:

```
...
2217 [main] INFO  org.apache.hadoop.security.UserGroupInformation  - Login successful for user developuser using keytab file user.keytab. Keytab auto renewal enabled : false
2217 [main] INFO  com.huawei.hadoop.security.LoginUtil  - Login success!!!!!!!!!!!!!!
3529 [main] WARN  org.apache.hadoop.hdfs.shortcircuit.DomainSocketFactory  - The short-circuit local reads feature cannot be used because UNIX Domain sockets are not available on Windows.
4632 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to create path /user/hdfs-examples
5392 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to write.
8200 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to append.
9384 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - result is : hi, I am bigdata. It is successful if you can see me.I append this content.
9384 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to read.
9636 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to delete the file /user/hdfs-examples\test.txt
9860 [main] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to delete path /user/hdfs-examples
10010 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to create path /user/hdfs-examples/hdfs_example_0
10069 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to create path /user/hdfs-examples/hdfs_example_1
10553 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to write.
10607 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to write.
13356 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to append.
13469 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to append.
13784 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - result is : hi, I am bigdata. It is successful if you can see me.I append this content.
13784 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to read.
13834 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - result is : hi, I am bigdata. It is successful if you can see me.I append this content.
13834 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to read.
13837 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to delete the file /user/hdfs-examples/hdfs_example_0\test.txt
13889 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to delete the file /user/hdfs-examples/hdfs_example_1\test.txt
14003 [hdfs_example_0] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to delete path /
```



```
user/hdfs-examples/hdfs_example_0
14118 [hdfs_example_1] INFO  com.huawei.bigdata.hdfs.examples.HdfsExample  - success to delete path /
user/hdfs-examples/hdfs_example_1
...
```

----End

12.3 Hive JDBC Application Development

Hive is an open-source data warehouse framework built on Hadoop. You can use it to store structured data and analyze data with the Hive query language (HiveQL) statements. Hive converts HiveQL statements to MapReduce or Spark jobs to query and analyze massive amounts of data stored in Hadoop clusters.

You can use Hive to:

- Extract, transform, and load (ETL) data with HiveQL.
- Analyze massive amounts of structured data with HiveQL.
- Process data in a wide range of formats, such as JSON, CSV, TEXTFILE, RCFILE, ORCFILE, and SEQUENCEFILE, and customize extensions.
- Connect the client flexibly and call JDBC APIs.

Hive is good for offline massive data analysis (such as log and cluster status analysis), large-scale data mining (such as user behavior analysis, interest region analysis, and region display), and other scenarios.

MRS provides sample application development projects based on Hive. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then compile and debug the code locally. In this sample project, you can create Hive tables, insert data, and read data.

Creating an MRS Hive Cluster

1. Create and purchase an MRS cluster that contains Hive. For details, see **Buying a Custom Cluster**.

   NOTE

   In this practice, an MRS 3.1.5 cluster, with Hadoop and Hive installed and with Kerberos authentication enabled, is used as an example.

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

Preparing the Application Development Configuration File

Step 1 Log in to FusionInsight Manager and create a cluster user for security authentication of the sample project.

Choose **System** > **Permission** > **User**. On the displayed page, click **Create**. On the displayed page, create a machine-machine user, for example, **developuser**.

Add **hive** and **supergroup** to **User Group**.

Step 2 Choose **System** > **Permission** > **User**. In the **Operation** column of **developuser**, choose **More** > **Download Authentication Credential**. Save the file and decompress it to obtain the **user.keytab** and **krb5.conf** files of the user.

Step 3 Choose **Cluster**. On the **Dashboard** tab, click **More** and select **Download Client**. In the dialog box that is displayed, set **Select Client Type** to **Configuration Files Only** and click **OK**. After the client package is generated, download the package as prompted and decompress it.

For example, if the client configuration file package is **FusionInsight_Cluster_1_Services_Client.tar**, decompress it to obtain **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar**. Then, continue to decompress this file.

1. Go to the **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles\Hive\config** directory and obtain the configuration files.
2. Copy all content from the **hosts** file in the decompression directory to your local **hosts** file. Ensure that the local PC can communicate with the hosts listed in the **hosts** file in the decompression directory.

   NOTE

   - In this practice, ensure that the local environment can communicate with the network plane where the MRS cluster is deployed. Generally, you can access the MRS cluster via an EIP.
   - **C:\WINDOWS\system32\drivers\etc\hosts** is an example directory in a Windows environment for storing the local **hosts** file.

----End

Obtaining the Sample Project

Step 1 Obtain the sample project from Huawei Mirrors.

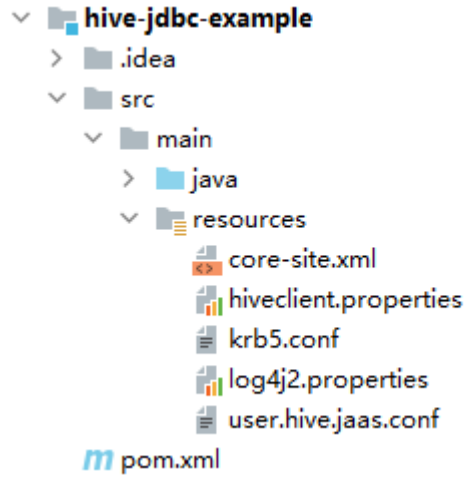
Download the source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **hive-jdbc-example**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.5/src/hive-examples/hive-jdbc-example**.

Step 2 Use IDEA to import the sample project and wait for the Maven project to download the dependency packages. For details, see **Configuring the JDBC Sample Project**.

Figure 12-9 Hive sample project



After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages.

Step 3 Place the cluster configuration files and user authentication credentials obtained in **Preparing the Application Development Configuration File** into the **resources** directory of the sample project.

Step 4 To connect to an MRS cluster with Kerberos authentication enabled, specify related authentication information in the sample code.

In the **JDBCExample** class of the **com.huawei.bigdata.hive.examples** package, change **USER_NAME** to the username you are using, for example, **developuser**.

```
KRB5_FILE = userdir + "krb5.conf";
System.setProperty("java.security.krb5.conf", KRB5_FILE);
USER_NAME = "developuser";
if ("KERBEROS".equalsIgnoreCase(auth)) {
    USER_KEYTAB_FILE = "src/main/resources/user.keytab";
    ZOOKEEPER_DEFAULT_SERVER_PRINCIPAL = "zookeeper/" + getUserRealm();
    System.setProperty(ZOOKEEPER_SERVER_PRINCIPAL_KEY, ZOOKEEPER_DEFAULT_SERVER_PRINCIPAL);
}
...
```

In this sample project, the development roadmap based on the service requirement is as follows.

1. Prepare data.
   a. Create an employee information table **employees_info**.
   b. Load employee information to **employees_info**.
2. Analyze data.
   Collect the number of records in the **employees_info** table.
3. Deletes the table.

----End

Building and Running the Application

Step 1 Compile the JDBC sample program.

Click **Terminal** in the lower left corner of the IDEA page to access the terminal. Run the **mvn clean package** command to perform compilation.

If "BUILD SUCCESS" is displayed, the compilation is successful. A JAR file containing the **-with-dependencies** field is generated in the **target** directory of the sample project.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:30 min
[INFO] Finished at: 2023-05-17T20:22:44+08:00
[INFO] ------------------------------------------------------------------------
```

Step 2 Create a directory as the runtime directory, for example, **D:\jdbc_example** in your local environment, save the generated JAR packages whose names contain the **-with-dependencies** field to the directory, and create the **src/main/resources** subdirectory in the directory. Copy all files from the **resources** directory of the sample project to this local subdirectory.

Step 3 Run the following commands in the Windows CMD environment:

cd /d d:\jdbc_example

java -jar hive-jdbc-example-*XXX*-with-dependencies.jar

Step 4 Check the output information after running the sample. The following information indicates that Hive table operations are successfully executed:

```
...
2023-05-17 20:25:09,421 INFO  HiveConnection - Login timeout is 0
2023-05-17 20:25:09,656 INFO  HiveConnection - user login success.
2023-05-17 20:25:09,685 INFO  HiveConnection - Will try to open client transport with JDBC Uri:
jdbc:hive2://192.168.64.216:21066/;principal=hive/hadoop.hadoop.com@HADOOP.COM;sasl.qop=auth-
conf;serviceDiscoveryMode=zooKeeper;auth=KERBEROS;zooKeeperNamespace=hiveserver2;user.principal=de
velopuser;user.keytab=src/main/resources/user.keytab
2023-05-17 20:25:30,294 INFO  JDBCExample - Create table success!
2023-05-17 20:26:34,032 INFO  JDBCExample - _c0
2023-05-17 20:26:34,266 INFO  JDBCExample - 0
2023-05-17 20:26:35,199 INFO  JDBCExample - Delete table success!
...
```

----End

12.4 Hive HCatalog Application Development

Hive is an open-source data warehouse framework built on Hadoop. You can use it to store structured data and analyze data with the Hive query language (HiveQL) statements. Hive converts HiveQL statements to MapReduce or Spark jobs to query and analyze massive amounts of data stored in Hadoop clusters.

You can use Hive to:

- Extract, transform, and load (ETL) data with HiveQL.
- Analyze massive amounts of structured data with HiveQL.
- Process data in a wide range of formats, such as JSON, CSV, TEXTFILE, RCFILE, ORCFILE, and SEQUENCEFILE, and customize extensions.
- Connect the client flexibly and call JDBC APIs.

Hive is good for offline massive data analysis (such as log and cluster status analysis), large-scale data mining (such as user behavior analysis, interest region analysis, and region display), and other scenarios.

MRS provides sample application development projects based on Hive. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then compile and debug the code locally. In this sample project, you can create Hive tables, insert data, and read data.

Creating an MRS Hive Cluster

1. Create and purchase an MRS cluster that contains Hive. For details, see **Buying a Custom Cluster**.

   NOTE

   In this practice, an MRS 3.1.5 cluster, with Hadoop and Hive installed and with Kerberos authentication enabled, is used as an example.

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

Preparing the Application Development Configuration File

Step 1 Log in to FusionInsight Manager to create a cluster user for creating Hive data tables and submitting the HCatalog program.

Choose **System** > **Permission** > **User**. On the displayed page, click **Create**. On the displayed page, create a machine-machine user, for example, **hiveuser**.

Add **hive** and **supergroup** to **User Group**.

Step 2 Download and install the cluster client to run the HCatalog program. For example, the installation directory is **/opt/client**.

----End

Obtaining the Sample Project

Step 1 Obtain the sample project from Huawei Mirrors.

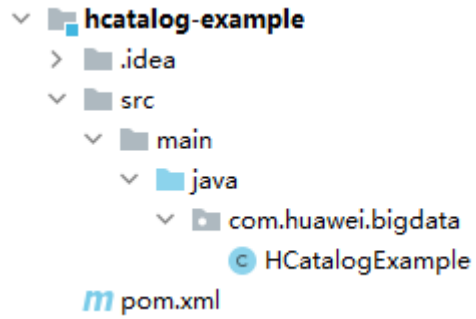
Download the source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **hcatalog-example**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.5/src/hive-examples/hcatalog-example**.

Step 2 Use IDEA to import the sample project and wait for the Maven project to download the dependency packages. For details, see **Configuring the JDBC Sample Project**.

Figure 12-10 Hive HCatalog sample project



After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages.

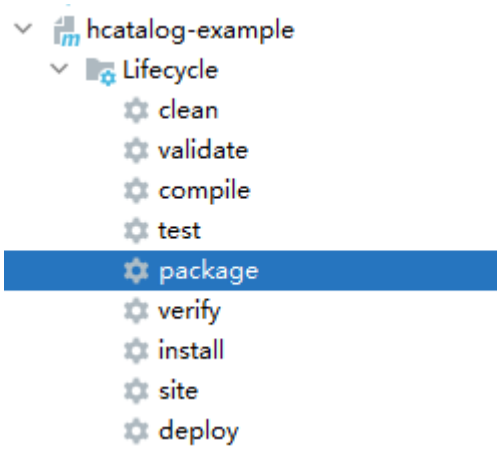
----End

Building and Running the Application

Step 1 Compile the HCatalog sample program.

1. In the Maven tool window, select **clean** from **Lifecycle** to execute the Maven building process.
2. Selecting **package** from **Lifecycle** and executing the Maven build process

Figure 12-11 Packaging the sample program



If "BUILD SUCCESS" is displayed, the compilation is successful.

The **hcatalog-example-*XXX*.jar** package is generated in the **target** directory of the sample project.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:30 min
[INFO] Finished at: 2023-05-17T20:22:44+08:00
[INFO] ------------------------------------------------------------------------
```

Step 2 Log in to the Hive Beeline CLI and create source tables and data tables for HCatalog analysis.

source /opt/client/bigdata_env

kinit hiveuser

beeline

create table t1(col1 int);

create table t2(col1 int,col2 int);

Insert test data into the source data table **t1**.

insert into table t1 select 1 union all select 1 union all select 2 union all select 2 union all select 3;

select * from t1;

```
+----------+
| t1.col1  |
+----------+
| 1        |
| 1        |
| 2        |
| 2        |
| 3        |
+----------+
```

Step 3 Upload the exported JAR package to the specified path of the Linux node where the cluster client is deployed, for example, **/opt/hive_demo**.

Step 4 To facilitate subsequent operations, configure the sample program directory and client component directory as public variables.

Exit the Beeline CLI and run the following commands:

export HCAT_CLIENT=/opt/hive_demo

export HADOOP_HOME=/opt/client/HDFS/hadoop

export HIVE_HOME=/opt/client/Hive/Beeline

export HCAT_HOME=$HIVE_HOME/../HCatalog

export LIB_JARS=$HCAT_HOME/lib/hive-hcatalog-core-*XXX*.jar,$HCAT_HOME/lib/hive-metastore-*XXX*.jar,$HCAT_HOME/lib/hive-standalone-metastore-*XXX*.jar,$HIVE_HOME/lib/hive-exec-*XXX*.jar,$HCAT_HOME/lib/libfb303-*XXX*.jar,$HCAT_HOME/lib/slf4j-api-*XXX*.jar,$HCAT_HOME/lib/jdo-api-*XXX*.jar,$HCAT_HOME/lib/antlr-runtime-*XXX*.jar,$HCAT_HOME/lib/datanucleus-api-jdo-*XXX*.jar,$HCAT_HOME/lib/datanucleus-core-*XXX*.jar,$HCAT_HOME/lib/datanucleus-rdbms-fi-*XXX*.jar,$HCAT_HOME/lib/log4j-api-*XXX*.jar,$HCAT_HOME/lib/log4j-core-*XXX*.jar,$HIVE_HOME/lib/commons-lang-*XXX*.jar,$HIVE_HOME/lib/hive-exec-*XXX*.jar

export HADOOP_CLASSPATH=$HCAT_HOME/lib/hive-hcatalog-core-*XXX*.jar:$HCAT_HOME/lib/hive-metastore-*XXX*.jar:$HCAT_HOME/lib/hive-standalone-metastore-*XXX*.jar:$HIVE_HOME/lib/hive-exec-*XXX*.jar:$HCAT_HOME/lib/libfb303-*XXX*.jar:$HADOOP_HOME/etc/hadoop:$HCAT_HOME/conf:$HCAT_HOME/lib/slf4j-api-*XXX*.jar:$HCAT_HOME/lib/jdo-api-*XXX*.jar:$HCAT_HOME/lib/antlr-runtime-*XXX*.jar:$HCAT_HOME/lib/datanucleus-api-jdo-*XXX*.jar:$HCAT_HOME/lib/datanucleus-core-*XXX*.jar:$HCAT_HOME/lib/datanucleus-rdbms-fi-

XXX.jar:$HCAT_HOME/lib/log4j-api-_XXX_.jar:$HCAT_HOME/lib/log4j-core-_XXX_.jar:$HIVE_HOME/lib/commons-lang-_XXX_.jar:$HIVE_HOME/lib/hive-exec-_XXX_.jar

NOTE

The version number *XXX* of the JAR package specified in **LIB_JARS** and **HADOOP_CLASSPATH** must be changed to the version you are using.

Step 5 Use the Yarn client to submit a task.

yarn --config $HADOOP_HOME/etc/hadoop jar $HCAT_CLIENT/hcatalog-example-_XXX_.jar com.huawei.bigdata.HCatalogExample -libjars $LIB_JARS t1 t2

```
...
2023-05-18 20:05:56,691 INFO mapreduce.Job: The url to track the job: https://host-192-168-64-122:26001/proxy/application_1683438782910_0008/
2023-05-18 20:05:56,692 INFO mapreduce.Job: Running job: job_1683438782910_0008
2023-05-18 20:06:07,250 INFO mapreduce.Job: Job job_1683438782910_0008 running in uber mode : false
2023-05-18 20:06:07,253 INFO mapreduce.Job:  map 0% reduce 0%
2023-05-18 20:06:15,362 INFO mapreduce.Job:  map 25% reduce 0%
2023-05-18 20:06:16,386 INFO mapreduce.Job:  map 50% reduce 0%
2023-05-18 20:06:35,999 INFO mapreduce.Job:  map 100% reduce 0%
2023-05-18 20:06:42,048 INFO mapreduce.Job:  map 100% reduce 100%
2023-05-18 20:06:43,136 INFO mapreduce.Job: Job job_1683438782910_0008 completed successfully
2023-05-18 20:06:44,118 INFO mapreduce.Job: Counters: 54
...
```

Step 6 After the job is complete, go to the Hive Beeline CLI, query the data in the **t2** table, and view the data analysis result.

select * from t2;

```
+----------+----------+
| t2.col1  | t2.col2  |
+----------+----------+
| 1        | 2        |
| 2        | 2        |
| 3        | 1        |
+----------+----------+
```

----End

12.5 Kafka Application Development

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
- Support for online and offline scenarios

MRS provides sample application development projects based on Kafka. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then conduct building and commissioning locally. In this sample project, you can implement processing of streaming data.

The guidelines for the sample project in this practice are as follows:

1. Create two topics on the Kafka client to serve as the input and output topics.
2. Develop Kafka Streams to count words in each message by reading messages in the input topic and to output the result in key-value pairs by consuming data in the output topic.

Creating an MRS Cluster

Step 1 Create and purchase an MRS cluster that contains Kafka. 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 client on any node in 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 packages.

Copy the Kafka JAR packages 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

Developing the Application

Step 1 Obtain the sample project from Huawei Mirrors.

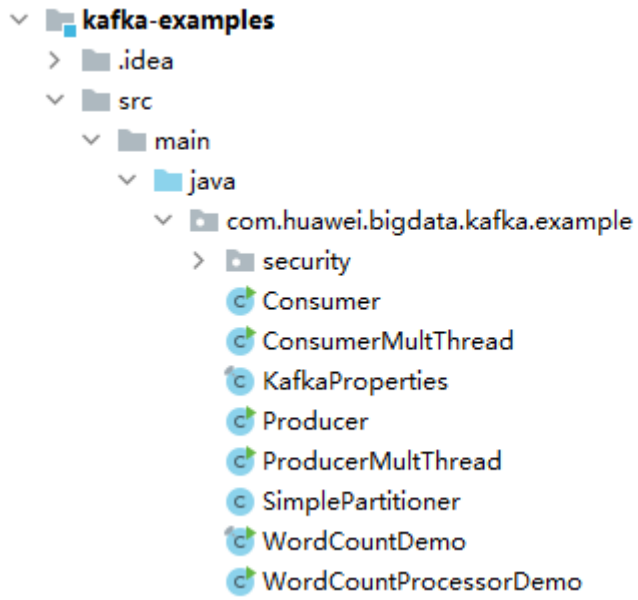
Download the Maven project source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **WordCountDemo**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.0/src/kafka-examples**.

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

After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages. For details, see **Configuring and Importing Sample Projects**.



The **WordCountDemo** sample project calls Kafka APIs to obtain and sort word records and then obtain the records of each word. The code snippet is as follows:

```
...
    static Properties getStreamsConfig() {
        final Properties props = new Properties();
        KafkaProperties kafkaProc = KafkaProperties.getInstance();
        //Set broker addresses 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"));
        //Set the protocol type, which 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 the calculation results of the key-value pairs.
        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.
```

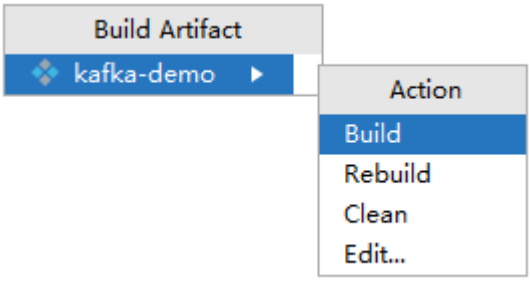
```
        counts.toStream().to(OUTPUT_TOPIC_NAME, Produced.with(Serdes.String(), Serdes.Long()));
    }
...
```

NOTE

- Set **BOOTSTRAP_SERVERS** to the host name and port number of the Kafka broker node based on site requirements. You can choose **Cluster** > **Services** > **Kafka** > **Instance** on FusionInsight Manager to view the broker instance information.
- Set **SECURITY_PROTOCOL** to the protocol for connecting to Kafka. In this example, set this parameter to **PLAINTEXT**.

Step 3 After confirming that the parameters in **WordCountDemo.java** are correct, build the project and package it into a JAR file.

For details about how to build a JAR file, see **Commissioning an Application in Linux**.



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

----End

Uploading the JAR Package and Source Data

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

NOTE

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

----End

Running a 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 obtain the ZooKeeper client connection port by querying the ZooKeeper 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 application:

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

Step 4 Open a new client 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 *IP address of the broker instance: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 window and run the following commands to use **kafka-console-consumer.sh** to consume data from the output topic and view the result:

cd /opt/client

source bigdata_env

kafka-console-consumer.sh --topic streams-wordcount-output --bootstrap-server *IP address of the broker instance: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 output is 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

12.6 Flink Application Development

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.

The Flink system consists of the following parts:

- Client

  Flink client is used to submit streaming jobs to Flink.
- TaskManager

  TaskManager is a service execution node of Flink, which executes specific tasks. There can be many TaskManagers, and they are equivalent to each other.
- JobManager

  JobManager is a management node of Flink. It manages all TaskManagers and schedules tasks submitted by users to specific TaskManagers. In high-availability (HA) mode, multiple JobManagers are deployed. Among these JobManagers, one is selected as the active JobManager, and the others are standby.

MRS provides sample application development projects based on multiple Flink components. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then conduct building and commissioning locally. In this sample project, you can implement Flink DataStream to process data.

Creating an MRS Flink Cluster

1. Create and purchase an MRS cluster that contains Hive. For details, see **Buying a Custom Cluster**.

   NOTE

   In this practice, an MRS 3.2.0-LTS.1 cluster, with Hadoop and Flink installed and with Kerberos authentication enabled, is used as an example.
2. Click **Buy Now** and wait until the MRS cluster is created.

Preparing the Cluster Configuration File

Step 1 After the cluster is created, log in to FusionInsight Manager and create a cluster user for submitting Flink jobs.

Choose **System** > **Permission** > **User**. On the displayed page, click **Create**. On the displayed page, create a machine-machine user, for example, **flinkuser**.

Add the **supergroup** user group and associate with the **System_administrator** role.

Step 2 Choose **System** > **Permission** > **User**. In the **Operation** column of **flinkuser**, choose **More** > **Download Authentication Credential**. Save the file and decompress it to obtain the **user.keytab** and **krb5.conf** files of the user.

Step 3 Choose **Cluster**. On the **Dashboard** tab, click **More** and select **Download Client**. In the dialog box that is displayed, set **Select Client Type** to **Configuration Files Only** and click **OK**. After the client package is generated, download the package as prompted and decompress it.

For example, if the client configuration file package is **FusionInsight_Cluster_1_Services_Client.tar**, decompress it to obtain **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles.tar**. Then, continue to decompress this file.

Go to the **FusionInsight_Cluster_1_Services_ClientConfig_ConfigFiles\Flink \config** directory and obtain the configuration files.

----End

Obtaining the Sample Project

Step 1 Obtain the sample project from Huawei Mirrors.

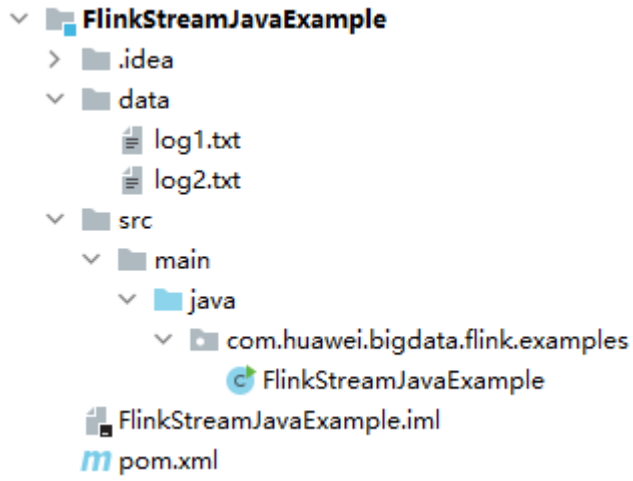
Download the source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **FlinkStreamJavaExample**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.2.0.1/src/flink-examples/flink-examples-security/FlinkStreamJavaExample**.

Step 2 Use IDEA to import the sample project and wait for the Maven project to download the dependency packages. For details, see **Configuring and Importing Sample Projects**.

Figure 12-12 Flink sample project



After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages.

Step 3 Use the Flink client to submit the developed DataStream program so security authentication is not required in the code.

Assume that there is a log file of time on site during weekends of an online shopping website. Write the DataStream program to collect real-time statistics on detailed information about the female users who spend more than 2 hours on online shopping.

The first column in the log file records names, the second column records gender, and the third column records the time on site (in minutes). Three columns are separated by commas (,).

- **log1.txt**: logs collected on Saturday

```
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,50
CaiXuyu,female,50
FangBo,female,60
```

log2.txt: logs collected on Sunday

```
LiuYang,female,20
YuanJing,male,10
CaiXuyu,female,50
FangBo,female,50
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
CaiXuyu,female,50
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
FangBo,female,50
GuoYijun,male,50
CaiXuyu,female,50
FangBo,female,60
```


The development procedure is as follows:

1. Read the text data, generate DataStreams, and parse data to generate UserRecord.
2. Search for the target data (time on site of female users).
3. Perform keyby operation based on names and genders, and calculate the total time that each female user spends online within a time window.
4. Search for users whose consecutive online duration exceeds the threshold.

```
public class FlinkStreamJavaExample {
    public static void main(String[] args) throws Exception {
        // Print the command reference for flink run.
        System.out.println("use command as: ");
        System.out.println("./bin/flink run --class
com.huawei.bigdata.flink.examples.FlinkStreamJavaExample /opt/test.jar --filePath /opt/log1.txt,/opt/
log2.txt --windowTime 2");
        System.out.println("*******************************************************************************************");
        System.out.println("<filePath> is for text file to read data, use comma to separate");
        System.out.println("<windowTime> is the width of the window, time as minutes");
        System.out.println("*******************************************************************************************");

        // Read text pathes and separate them with commas (,). If the source file is in the HDFS, set this
parameter to a specific HDFS path, for example, hdfs://hacluster/tmp/log1.txt,hdfs://hacluster/tmp/
log2.txt.
        final String[] filePaths = ParameterTool.fromArgs(args).get("filePath", "/opt/log1.txt,/opt/
log2.txt").split(",");
        assert filePaths.length > 0;

       // Set the time window. The default value is 2 minutes per time window. One time window is sufficient
to read all data in the text.
        final int windowTime = ParameterTool.fromArgs(args).getInt("windowTime", 2);

       // Construct an execution environment and use eventTime to process the data obtained in a time
window.
        final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
        env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);
        env.setParallelism(1);

       // Read the text data stream.
        DataStream<String> unionStream = env.readTextFile(filePaths[0]);
        if (filePaths.length > 1) {
            for (int i = 1; i < filePaths.length; i++) {
                unionStream = unionStream.union(env.readTextFile(filePaths[i]));
            }
        }

       // Convert the data, construct data processing logic, and calculate and print the results.
        unionStream.map(new MapFunction<String, UserRecord>() {
            @Override
            public UserRecord map(String value) throws Exception {
                return getRecord(value);
            }
        }).assignTimestampsAndWatermarks(
                new Record2TimestampExtractor()
        ).filter(new FilterFunction<UserRecord>() {
            @Override
            public boolean filter(UserRecord value) throws Exception {
                return value.sexy.equals("female");
            }
        }).keyBy(
            new UserRecordSelector()
        ).window(
            TumblingEventTimeWindows.of(Time.minutes(windowTime))
        ).reduce(new ReduceFunction<UserRecord>() {
            @Override
            public UserRecord reduce(UserRecord value1, UserRecord value2)
                    throws Exception {
```



```
            value1.shoppingTime += value2.shoppingTime;
            return value1;
        }
    }).filter(new FilterFunction<UserRecord>() {
        @Override
        public boolean filter(UserRecord value) throws Exception {
            return value.shoppingTime > 120;
        }
    }).print();

    // Call execute to trigger the execution.
    env.execute("FemaleInfoCollectionPrint java");
}

// Construct a keyBy keyword for grouping.
private static class UserRecordSelector implements KeySelector<UserRecord, Tuple2<String, String>> {
    @Override
    public Tuple2<String, String> getKey(UserRecord value) throws Exception {
        return Tuple2.of(value.name, value.sexy);
    }
}

// Resolve text line data and construct the UserRecord data structure.
private static UserRecord getRecord(String line) {
    String[] elems = line.split(",");
    assert elems.length == 3;
    return new UserRecord(elems[0], elems[1], Integer.parseInt(elems[2]));
}

// Define the UserRecord data structure and override the toString printing method.
public static class UserRecord {
    private String name;
    private String sexy;
    private int shoppingTime;

    public UserRecord(String n, String s, int t) {
        name = n;
        sexy = s;
        shoppingTime = t;
    }

    public String toString() {
        return "name: " + name + "  sexy: " + sexy + "  shoppingTime: " + shoppingTime;
    }
}

// Construct a class inherited from AssignerWithPunctuatedWatermarks to set eventTime and
waterMark.
    private static class Record2TimestampExtractor implements
AssignerWithPunctuatedWatermarks<UserRecord> {

        // add tag in the data of datastream elements
        @Override
        public long extractTimestamp(UserRecord element, long previousTimestamp) {
            return System.currentTimeMillis();
        }

        // give the watermark to trigger the window to execute, and use the value to check if the window
elements is ready
        @Override
        public Watermark checkAndGetNextWatermark(UserRecord element, long extractedTimestamp) {
            return new Watermark(extractedTimestamp - 1);
        }
    }
}
```

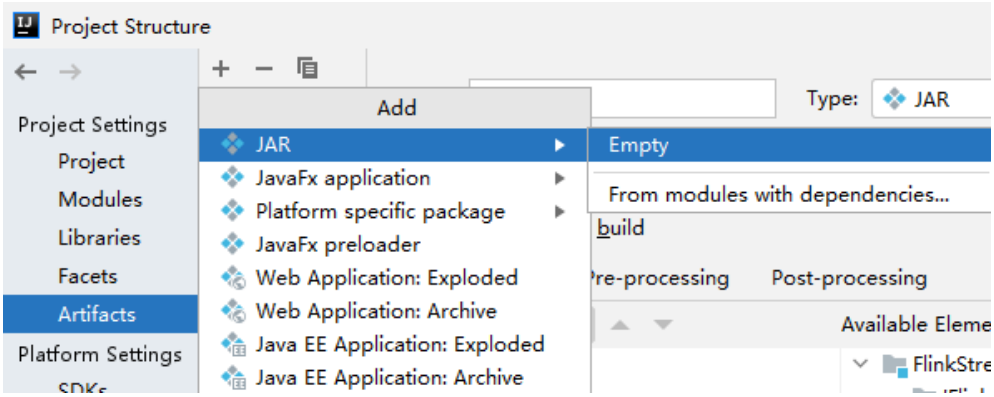
----End

Building and Running the Application

Step 1 In IntelliJ IDEA, configure the Artifacts information of the project.

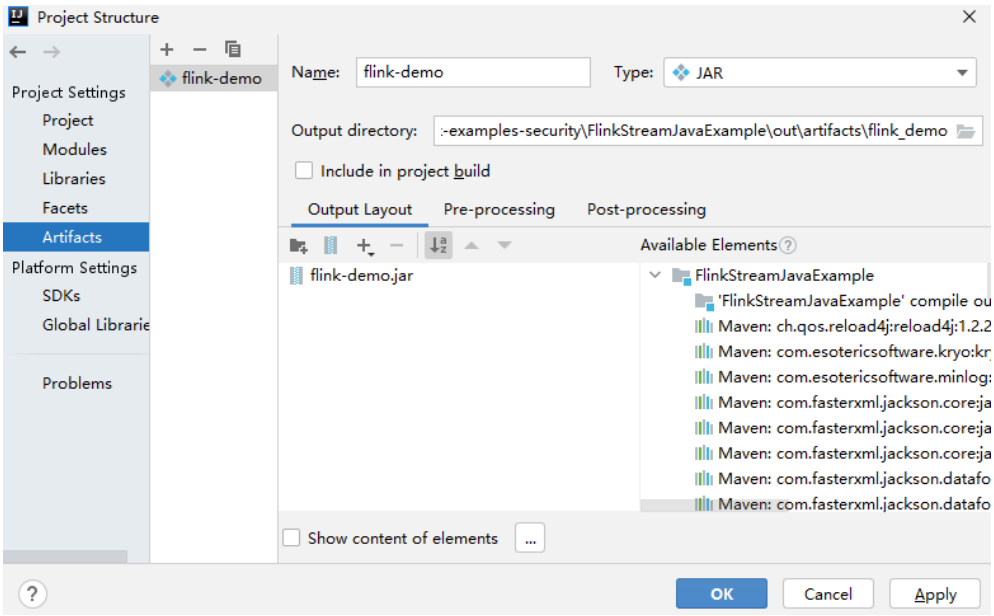
1. On the IDEA homepage, choose **File** > **Project Structures...**.
2. On the **Project Structure** page, select **Artifacts**, click **+**, and choose **JAR** > **Empty**.

Figure 12-13 Adding Artifacts



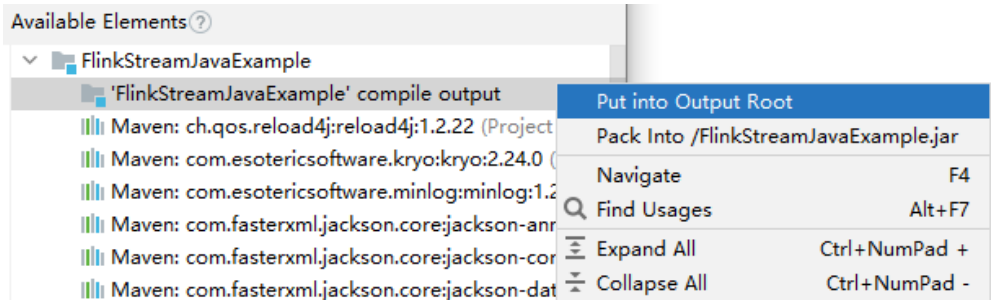
3. Set the name, type, and output path of the JAR package, for example, **flink-demo**.

Figure 12-14 Setting basic information



4. Right-click **'FlinkStreamJavaExample' compile output** and choose **Put into Output Root** from the shortcut menu. Then, click **Apply**.

Figure 12-15 Put into Output Root

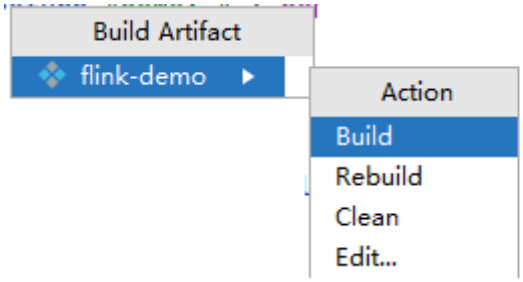


5. Click **OK**.

Step 2 Generate a JAR file.

1. On the IDEA home page, choose **Build** > **Build Artifacts...**.
2. In the displayed menu, choose **FlinkStreamJavaExample** > **Build** to generate the JAR file.

Figure 12-16 Build



3. Obtain the **flink-demo.jar** file from the path configured in **Step 1.3**.

Step 3 Install and configure the Flink client.

1. Install the MRS cluster client, for example, in **/opt/hadoopclient**.
2. Decompress the authentication credential package downloaded from **Preparing the Cluster Configuration File** and copy the obtained file to a directory on the client node, for example, **/opt/hadoopclient/Flink/flink/conf**.
3. Run the following command to set Flink client configuration parameters and save the settings:

   vi /opt/hadoopclient/Flink/flink/conf/flink-conf.yaml

   Add the service IP address of the client node and the floating IP address of FusionInsight Manager to the **jobmanager.web.allow-access-address** configuration item, and add the **keytab** path and username to the corresponding configuration items.

   ```
   ...
   jobmanager.web.allow-access-address: 192.168.64.122,192.168.64.216,192.168.64.234
   ...
   security.kerberos.login.keytab: /opt/client/Flink/flink/conf/user.keytab
   security.kerberos.login.principal: flinkuser
   ...
   ```

4. Configure security authentication.

   a. Run the following commands to generate a Flink client security authentication file:

cd /opt/hadoopclient/Flink/flink/bin

sh generate_keystore.sh

Enter a user-defined password for authentication.

b. Configure paths for the client to access the **flink.keystore** and **flink.truststore** files.

cd /opt/hadoopclient/Flink/flink/conf/

mkdir ssl

mv flink.keystore ssl/

mv flink.truststore ssl/

vi /opt/hadoopclient/Flink/flink/conf/flink-conf.yaml

Change the paths of the following two parameters to relative paths:

```
security.ssl.keystore: ssl/flink.keystore
security.ssl.truststore: ssl/flink.truststore
```

Step 4 Upload the JAR package generated in **Step 2** to the related directory on the Flink client node, for example, **/opt/hadoopclient**.

Create the **conf** directory in the directory where the JAR package is located, and upload the configuration files in **Flink/config** of the cluster client configuration file package obtained in **Preparing the Cluster Configuration File** to the **conf** directory.

Step 5 Upload the application source data files to the node where the NodeManager instance is deployed.

In this example, source data files **log1.txt** and **log2.txt** are stored on the local host. You need to upload the files to the **/opt** directory on all Yarn NodeManager nodes and set the file permission to **755**.

Step 6 On the Flink client, run the **yarn session** command to start the Flink cluster.

cd /opt/hadoopclient/Flink/flink

bin/yarn-session.sh -jm 1024 -tm 1024 -t conf/ssl/

```
...
Cluster started: Yarn cluster with application id application_1683438782910_0009
JobManager Web Interface: http://192.168.64.10:32261
```

Step 7 Open a new client connection window, go to the Flink client directory, and run the program.

source /opt/hadoopclient/bigdata_env

cd /opt/hadoopclient/Flink/flink

bin/flink run --class com.huawei.bigdata.flink.examples.FlinkStreamJavaExample /opt/hadoopclient/flink-demo.jar --filePath /opt/log1.txt,/opt/log2.txt --windowTime 2

```
...
2023-05-26 19:56:52,068 | INFO  | [main] | Found Web Interface host-192-168-64-10:32261 of application 'application_1683438782910_0009'. | org.apache.flink.yarn.YarnClusterDescriptor.setClusterEntrypointInfoToConfig(YarnClusterDescriptor.java:1854)
Job has been submitted with JobID 7647255752b09456d5a580e33a8529f5
Program execution finished
```

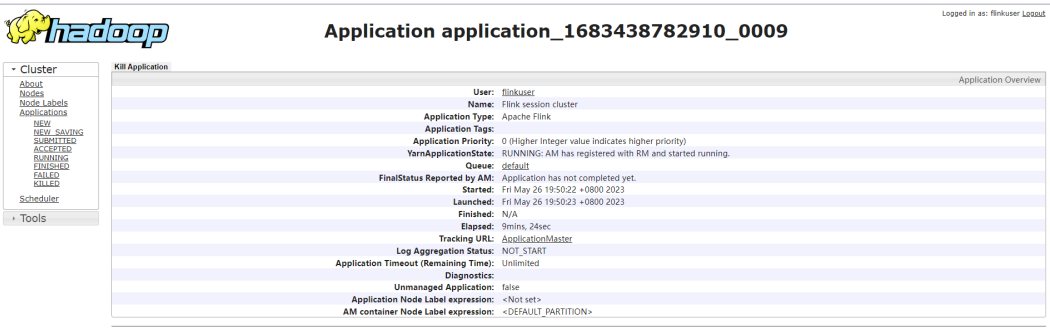


```
Job with JobID 7647255752b09456d5a580e33a8529f5 has finished.
Job Runtime: 36652 ms
```

Step 8 Check execution results.

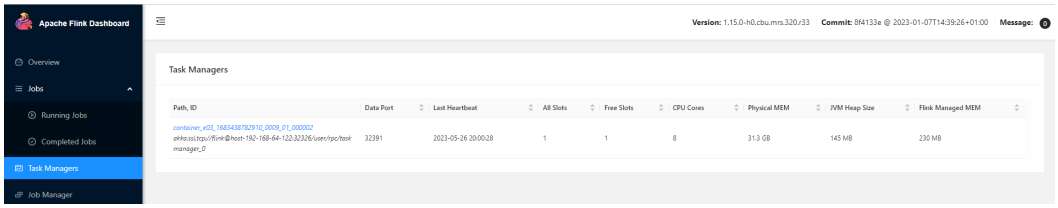
Log in to FusionInsight Manager as user **flinkuser** and choose **Cluster** > **Service** > **Yarn**. On the **Applications** page, click a job name to go to the job details page.

Figure 12-17 Viewing Yarn job details



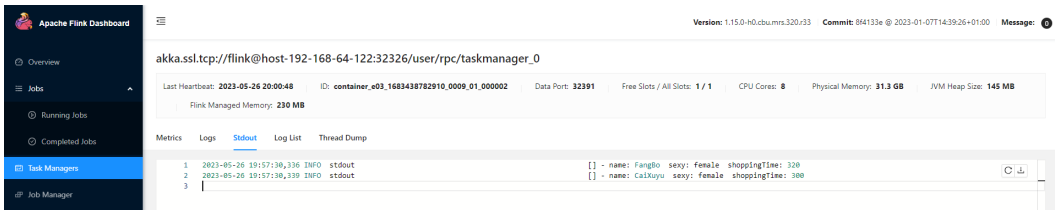
For the job submitted in a session, you can click **Tracking URL** to log in to the native Flink service page to view job information.

Figure 12-18 Viewing Flink job details



In this sample project, click **Task Managers** and view the running result in the **Stdout** tab of the job.

Figure 12-19 View application running results



----End

12.7 ClickHouse Application Development

ClickHouse is a column-oriented database for online analytical processing. It supports SQL query and provides good query performance. The aggregation analysis and query performance based on large and wide tables is excellent, which is one order of magnitude faster than other analytical databases.

ClickHouse features:

- High data compression ratio
- Multi-core parallel computing
- Vectorized computing engine
- Support for nested data structure
- Support for sparse indexes
- Support for INSERT and UPDATE

ClickHouse application scenarios:

- Real-time data warehouse

  The streaming computing engine (such as Flink) is used to write real-time data to ClickHouse. With the excellent query performance of ClickHouse, multi-dimensional and multi-mode real-time query and analysis requests can be responded within subseconds.

- Offline query

  Large-scale service data is imported to ClickHouse to construct a large wide table with hundreds of millions to tens of billions of records and hundreds of dimensions. It supports personalized statistics collection and continuous exploratory query and analysis at any time to assist business decision-making and provide excellent query experience.

MRS provides sample application development projects based on ClickHouse JDBC. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then conduct building and commissioning locally. In this sample project, you can create and delete ClickHouse tables, and insert and query data in the MRS cluster.

Creating an MRS ClickHouse Cluster

1. Create and purchase an MRS cluster that contains ClickHouse. For details, see **Buying a Custom Cluster**.

   NOTE

   In this practice, an MRS 3.2.0-LTS.1 cluster, with ClickHouse installed and with Kerberos authentication enabled, is used as an example.

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

Preparing an Application Authentication User

For an MRS cluster with Kerberos authentication enabled, prepare a user who has the operation permission on related components for application authentication.

The following ClickHouse permission configuration example is for reference only. You can modify the configuration as you need.

Step 1 After the cluster is created, log in to FusionInsight Manager.

Step 2 Choose **System** > **Permission** > **Role** and click **Create Role** in the right pane.

1. Enter a role name, for example, **developrole**, and click **OK**.
2. In **Configure Resource Permission**, select the desired cluster, choose **ClickHouse**, and select **SUPER_USER_GROUP**.

Step 3 Choose **System** > **Permission** > **User**, click **Create** in the right pane to create a human-machine user, for example, **developuser**, and add the **developrole** role.

After the user is created, log in to FusionInsight Manager as **developuser** and change the initial password as prompted.

----End

Obtaining the Sample Project

Step 1 Obtain the sample project from Huawei Mirrors.

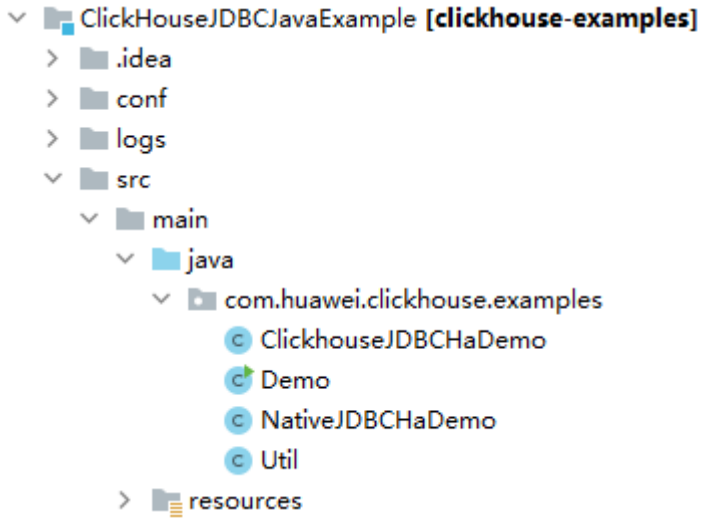
Download the Maven project source code and configuration files of the sample project, and configure related development tools on your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

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

For example, the sample project suitable for this practice is **clickhouse-examples**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.2.0.1/src/clickhouse-examples**.

Step 2 Use IDEA to import the sample project and wait for the Maven project to download the dependency packages. For details, see **Configuring and Importing Sample Projects**.

Figure 12-20 ClickHouse sample project



After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages.

Step 3 In this sample project, the application connects to the ClickHouse server through the IP address and user information in the configuration file. Therefore, after the project is imported, you need to modify the **clickhouse-example.properties** file in the **conf** directory of the sample project based on the actual environment information.

```
loadBalancerIPList=192.168.64.10,192.168.64.122
sslUsed=true
loadBalancerHttpPort=21425
```



```
loadBalancerHttpsPort=21426
CLICKHOUSE_SECURITY_ENABLED=true
user=developuser
password=Bigdata_!@#
isMachineUser=false
isSupportMachineUser=false
clusterName=default_cluster
databaseName=testdb
tableName=testtb
batchRows=10000
batchNum=10
clickhouse_dataSource_ip_list=192.168.64.10:21426,192.168.64.122:21426
native_dataSource_ip_list=192.168.64.10:21424,192.168.64.122:21424
```

Table 12-3 Configuration description

Configuration Item	Description
loadBalancerIPList	Addresses of the ClickHouseBalancer instances. To view the instance IP addresses, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click **Instance**. In this example, set this parameter to **192.168.64.10,192.168.64.122**.
sslUsed	Whether to enable SSL encryption. Set this parameter to **true** for clusters in security mode.
loadBalancerHttp-Port	HTTP and HTTPS port numbers of the load balancer. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Logical Cluster**, locate the row containing the desired logical cluster, and view **Port** and **Ssl Port** in the **HTTP Balancer Port** column.
loadBalancerHttp-sPort	
CLICKHOUSE_SECURITY_ENABLED	Whether to enable the ClickHouse security mode. In this example, set this parameter to **true**.
user	Authentication information of the development user. For a machine-machine user, leave **password** empty.
password	
isMachineUser	Whether the authentication user is a machine-machine user.
isSupportMachineUs-er	Whether to support authentication of a machine-machine user. In this example, set this parameter to **false**.
clusterName	Name of the ClickHouse logical cluster connected to the application. In this example, retain the default value **default_cluster**.
databaseName	Names of the database and table to be created in the sample project. You can change the names based on site requirements.
tableName	
batchRows	Number of data records to be written in a batch. In this example, set this parameter to **10**.

Configuration Item	Description
batchNum	Total number of batches for writing data. Retain the default value in this example.
clickhouse_dataSource_ip_list	Addresses and HTTP ports of the ClickHouseBalancer instances. Log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click **Logical Cluster**. This example uses a cluster in security mode. Therefore, locate the row containing the desired logical cluster, and view **Ssl Port** in the **HTTP Balancer Port** column. In this example, set this parameter to **192.168.64.10:21426,192.168.64.122:21426**.
native_dataSource_ip_list	Addresses and TCP ports of the ClickHouseBalancer instances. Log in to FusionInsight Manager and choose **Cluster** > **Services** > **ClickHouse**. Click **Logical Cluster**, locate the row containing the desired logical cluster, and view **Port** in the **TCP Balancer Port** column. In this example, set this parameter to **192.168.64.10:21424,192.168.64.122:21424**.

Step 4 Develop the application in this sample project through the clickhouse-jdbc API. For details about the code snippets of each function, see **ClickHouse Sample Code**.

- Setting up a connection: Set up a connection to the ClickHouse service instance.

  During connection setup, the user information configured in **Table 12-3** is passed as the authentication credential for security authentication on the server.

  ```
  clickHouseProperties.setPassword(userPass);
  clickHouseProperties.setUser(userName);
  BalancedClickhouseDataSource balancedClickhouseDataSource = new
  BalancedClickhouseDataSource(JDBC_PREFIX + UriList, clickHouseProperties);
  ```

- Creating a database: Create a ClickHouse database.

  Execute the **on cluster** statement to create a database in the cluster.

  ```
  private void createDatabase(String databaseName, String clusterName) throws Exception  {
      String createDbSql = "create database if not exists " + databaseName + " on cluster " +
  clusterName;
      util.exeSql(createDbSql);
  }
  ```

- Creating a table: Create a table in the ClickHouse database.

  Execute the **on cluster** statement to create a **ReplicatedMerge** table and a **Distributed** table in the cluster.

  ```
  private void createTable(String databaseName, String tableName, String clusterName) throws
  Exception {
     String createSql = "create table " + databaseName + "." + tableName + " on cluster " +
  clusterName  + " (name String, age UInt8, date Date)engine=ReplicatedMergeTree('/clickhouse/tables/
  {shard}/" + databaseName + "." + tableName + "'," + "'{replica}') partition by toYYYYMM(date) order
  by age";
     String createDisSql = "create table " + databaseName + "." + tableName + "_all" + " on cluster " +
  clusterName + " as " + databaseName + "." + tableName + " ENGINE = Distributed(default_cluster," +
  ```



```
databaseName + "," + tableName + ", rand());";    ArrayList<String> sqlList = new
ArrayList<String>();
  sqlList.add(createSql);
  sqlList.add(createDisSql);
  util.exeSql(sqlList);
}
```

- Inserting data: Insert data into the ClickHouse table.

  Insert data into the created table. The table created in this example has three columns, **String**, **UInt8**, and **Date**.

```
String insertSql = "insert into " + databaseName + "." + tableName + " values (?,?,?)";
PreparedStatement preparedStatement = connection.prepareStatement(insertSql);
long allBatchBegin = System.currentTimeMillis();
for (int j = 0; j < batchNum; j++) {
    for (int i = 0; i < batchRows; i++) {
      preparedStatement.setString(1, "huawei_" + (i + j * 10));
      preparedStatement.setInt(2, ((int) (Math.random() * 100)));
      preparedStatement.setDate(3, generateRandomDate("2018-01-01", "2021-12-31"));
      preparedStatement.addBatch();
    }
  long begin = System.currentTimeMillis();
  preparedStatement.executeBatch();
  long end = System.currentTimeMillis();
  log.info("Inert batch time is {} ms", end - begin);
}
long allBatchEnd = System.currentTimeMillis();
log.info("Inert all batch time is {} ms", allBatchEnd - allBatchBegin);
```

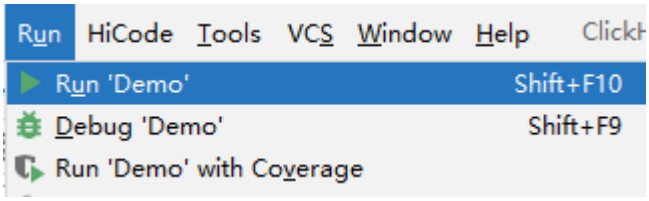
----End

Building and Running the Application

If you can access the MRS cluster from your local PC, you can commission and run the application locally.

Step 1 In the **clickhouse-examples** project of IntelliJ IDEA, click **Run 'Demo'** to run the application project.

Figure 12-21 Running the ClickHouse Demo application



Step 2 View the output on the console, as shown in the following figure. You can see that the ClickHouse table is created and data is inserted successfully.

```
...
2023-06-03 11:30:27,127 | INFO  | main | Execute query:create table testdb.testtb on cluster default_cluster
(name String, age UInt8, date Date)engine=ReplicatedMergeTree('/clickhouse/tables/{shard}/
testdb.testtb','{replica}') partition by toYYYYMM(date) order by age |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:68)
2023-06-03 11:30:27,412 | INFO  | main | Execute time is 284 ms |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:72)
2023-06-03 11:30:27,412 | INFO  | main | Current load balancer is 192.168.64.10:21426 |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:63)
2023-06-03 11:30:28,426 | INFO  | main | Execute query:create table testdb.testtb_all on cluster
default_cluster as testdb.testtb ENGINE = Distributed(default_cluster,testdb,testtb, rand()); |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:68)
2023-06-03 11:30:28,686 | INFO  | main | Execute time is 259 ms |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:72)
2023-06-03 11:30:28,686 | INFO  | main | Current load balancer is 192.168.64.10:21426 |
```



```
com.huawei.clickhouse.examples.Util.insertData(Util.java:137)
2023-06-03 11:30:29,784 | INFO  | main | Insert batch time is 227 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:31,490 | INFO  | main | Insert batch time is 200 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:33,337 | INFO  | main | Insert batch time is 335 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:35,295 | INFO  | main | Insert batch time is 454 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:37,077 | INFO  | main | Insert batch time is 275 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:38,811 | INFO  | main | Insert batch time is 218 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:40,468 | INFO  | main | Insert batch time is 144 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:42,216 | INFO  | main | Insert batch time is 238 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:43,977 | INFO  | main | Insert batch time is 257 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:45,756 | INFO  | main | Insert batch time is 277 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:154)
2023-06-03 11:30:47,270 | INFO  | main | Inert all batch time is 17720 ms |
com.huawei.clickhouse.examples.Util.insertData(Util.java:158)
2023-06-03 11:30:47,271 | INFO  | main | Current load balancer is 192.168.64.10:21426 |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:63)
2023-06-03 11:30:47,828 | INFO  | main | Execute query:select * from testdb.testtb_all order by age limit 10 |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:68)
2023-06-03 11:30:47,917 | INFO  | main | Execute time is 89 ms |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:72)
2023-06-03 11:30:47,918 | INFO  | main | Current load balancer is 192.168.64.10:21426 |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:63)
2023-06-03 11:30:48,580 | INFO  | main | Execute query:select toYYYYMM(date),count(1) from
testdb.testtb_all group by toYYYYMM(date) order by count(1) DESC limit 10 |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:68)
2023-06-03 11:30:48,680 | INFO  | main | Execute time is 99 ms |
com.huawei.clickhouse.examples.Util.exeSql(Util.java:72)
2023-06-03 11:30:48,682 | INFO  | main | name    age    date    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,682 | INFO  | main | huawei_89    3    2021-02-21    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,682 | INFO  | main | huawei_81    3    2020-05-27    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,682 | INFO  | main | huawei_70    4    2021-10-28    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,682 | INFO  | main | huawei_73    4    2020-03-23    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | huawei_44    5    2020-12-10    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | huawei_29    6    2021-10-12    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | huawei_74    6    2021-03-03    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | huawei_38    7    2020-05-30    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | huawei_57    8    2020-09-27    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | huawei_23    8    2020-08-08    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,683 | INFO  | main | toYYYYMM(date)    count()    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,684 | INFO  | main | 202005    8    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,684 | INFO  | main | 202007    7    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,684 | INFO  | main | 202004    6    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,684 | INFO  | main | 202009    6    |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,684 | INFO  | main | 202103    6    |
```



```
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,685 | INFO  | main | 202012    6     |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,685 | INFO  | main | 202010    5     |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,685 | INFO  | main | 202112    5     |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,685 | INFO  | main | 202003    5     |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,685 | INFO  | main | 202104    4     |
com.huawei.clickhouse.examples.Demo.queryData(Demo.java:159)
2023-06-03 11:30:48,689 | INFO  | main | Use HA module. |
ru.yandex.clickhouse.BalancedClickhouseDataSource.<init>(BalancedClickhouseDataSource.java:122)
2023-06-03 11:30:51,651 | INFO  | main | Name is: huawei_89, age is: 3 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,651 | INFO  | main | Name is: huawei_81, age is: 3 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,651 | INFO  | main | Name is: huawei_70, age is: 4 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,651 | INFO  | main | Name is: huawei_73, age is: 4 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,652 | INFO  | main | Name is: huawei_44, age is: 5 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,652 | INFO  | main | Name is: huawei_29, age is: 6 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,652 | INFO  | main | Name is: huawei_74, age is: 6 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,652 | INFO  | main | Name is: huawei_38, age is: 7 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,654 | INFO  | main | Name is: huawei_57, age is: 8 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
2023-06-03 11:30:51,654 | INFO  | main | Name is: huawei_23, age is: 8 |
com.huawei.clickhouse.examples.ClickhouseJDBCHaDemo.queryData(ClickhouseJDBCHaDemo.java:73)
...
```

Step 3 Install the MRS cluster client and log in to the ClickHouse client.

For example, if the client installation directory is **/opt/client**, log in to the node where the client is installed as the client installation user.

cd /opt/client

source bigdata_env

kinit developuser

Step 4 Run the following command to connect to the ClickHouse server:

clickhouse client --host *IP address of the ClickHouseServer instance* **--port** *Connection port* **--secure**

To obtain the IP address of the ClickHouse instance, log in to FusionInsight Manager, choose **Cluster** > **Services** > **ClickHouse**, and click the **Instance** tab. You can obtain the connection port by searching for the **tcp_port_secure** parameter in the ClickHouse service configurations.

For example, run the following command:

clickhouse client --host 192.168.64.10 --port 21427 --secure

Step 5 Run the following command to view the table content created by the application:

select * from testdb.testtb;

```
┌─name─────┬─age─┬─────date─┐
│ huawei_70 │   4 │ 2021-10-28 │
│ huawei_29 │   6 │ 2021-10-12 │
```



```
| huawei_16 | 28 | 2021-10-04 |
| huawei_15 | 29 | 2021-10-03 |
+-----------+----+------------+
...
```

----End

12.8 Spark Application Development

Spark is a distributed batch processing framework. It provides analysis and mining and iterative memory computation capabilities and supports application development in multiple programming languages. It 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 keep up with the multi-step data processing logic.
- Data mining: Based on massive data, Spark can handle complex data mining and analysis and supports multiple data mining and machine learning algorithms.
- Streaming processing: Spark supports streaming processing with only a seconds-level latency and supports multiple external data sources.
- Query analysis: Spark supports standard SQL query analysis, provides the DSL (DataFrame), and supports multiple external inputs.

MRS provides sample application development projects based on Spark. This practice provides guidance for you to obtain and import a sample project after creating an MRS cluster and then conduct building and commissioning locally. In this sample project, you can read data from Hive tables and re-write the data to HBase tables.

The guidelines for the sample project in this practice are as follows:

1. Query data in a specified Hive table.
2. Query data in a specified HBase table based on the key of the data in the Hive table.
3. Add related data records and write them to the HBase table.

Creating an MRS Cluster

Step 1 Create and purchase an MRS cluster that contains Spark. For details, see **Buying a Custom Cluster**.

NOTE

In this practice, an MRS 3.1.5 cluster, with Spark2x, Hive, and HBase installed and with Kerberos authentication enabled, is used as an example.

Step 2 After the cluster is purchased, install the client on any node in 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.

----End

Preparing the Cluster Configuration File

Step 1 After the cluster is created, log in to FusionInsight Manager and create a cluster user for submitting Flink jobs.

Choose **System** > **Permission** > **User**. In the right pane, click **Create**. On the displayed page, create a human-machine user, for example, **sparkuser**.

Add the **supergroup** user group and associate with the **System_administrator** role.

Step 2 Log in to FusionInsight Manager as the new user and change the initial password as prompted.

Step 3 Choose **System** > **Permission** > **User**. In the **Operation** column of **sparkuser**, choose **More** > **Download Authentication Credential**. Save the file and decompress it to obtain the **user.keytab** and **krb5.conf** files of the user.

----End

Developing the Application

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 your local PC. For details, see **Obtaining Sample Projects from Huawei Mirrors**.

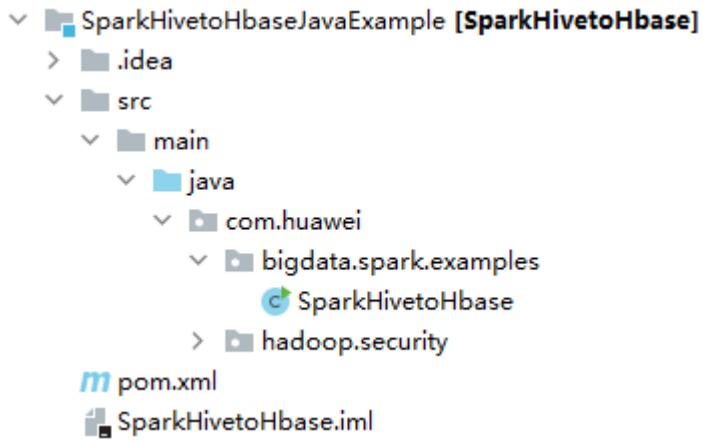
Select a branch based on the cluster version and download the required MRS sample project.

For example, the sample project suitable for this practice is **SparkHivetoHbase**, which can be obtained at **https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-3.1.5/src/spark-examples/sparksecurity-examples/SparkHivetoHbaseJavaExample**.

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

After you configure Maven and SDK parameters on the local PC, the sample project automatically loads related dependency packages. For details, see **Configuring and Importing Sample Projects**.

Figure 12-22 Spark Hive to HBase sample project



The **SparkHivetoHbase** class in the sample project uses Spark to call Hive APIs to operate a Hive table, obtain the corresponding record from an HBase table based on the key, perform operations on the two data records, and update the data to the HBase table.

The code snippet is as follows:

```
...
public class SparkHivetoHbase {
    public static void main(String[] args) throws Exception {
        String userPrincipal = "sparkuser";     //Specifies the cluster user information and keytab file address used for authentication.
        String userKeytabPath = "/opt/client/user.keytab";
        String krb5ConfPath = "/opt/client/krb5.conf";
        Configuration hadoopConf = new Configuration();
        LoginUtil.login(userPrincipal, userKeytabPath, krb5ConfPath, hadoopConf);
        //Calls the Spark API to obtain table data.
        SparkConf conf = new SparkConf().setAppName("SparkHivetoHbase");
        JavaSparkContext jsc = new JavaSparkContext(conf);
        HiveContext sqlContext = new org.apache.spark.sql.hive.HiveContext(jsc);
        Dataset<Row> dataFrame = sqlContext.sql("select name, account from person");
        //Traverses partitions in the Hive table and updates the partitions to the HBase table.
            dataFrame
            .toJavaRDD()
            .foreachPartition(
                new VoidFunction<Iterator<Row>>() {
                    public void call(Iterator<Row> iterator) throws Exception {
                        hBaseWriter(iterator);
                    }
                });
        jsc.stop();
    }
    //Updates records in the HBase table on the executor.
    private static void hBaseWriter(Iterator<Row> iterator) throws IOException {
        //Reads the HBase table.
        String tableName = "table2";
        String columnFamily = "cf";
        Configuration conf = HBaseConfiguration.create();
        Connection connection = ConnectionFactory.createConnection(conf);
        Table table = connection.getTable(TableName.valueOf(tableName));
        try {
            connection = ConnectionFactory.createConnection(conf);
            table = connection.getTable(TableName.valueOf(tableName));
            List<Row> table1List = new ArrayList<Row>();
            List<Get> rowList = new ArrayList<Get>();
            while (iterator.hasNext()) {
                Row item = iterator.next();
                Get get = new Get(item.getString(0).getBytes());
                table1List.add(item);
                rowList.add(get);
            }
            //Obtains the records in the HBase table.
            Result[] resultDataBuffer = table.get(rowList);
            //Modifies records in the HBase table.
            List<Put> putList = new ArrayList<Put>();
            for (int i = 0; i < resultDataBuffer.length; i++) {
                Result resultData = resultDataBuffer[i];
                if (!resultData.isEmpty()) {
                    int hiveValue = table1List.get(i).getInt(1);
                    String hbaseValue = Bytes.toString(resultData.getValue(columnFamily.getBytes(), "cid".getBytes()));
                    Put put = new Put(table1List.get(i).getString(0).getBytes());
                    //Calculates the result.
                    int resultValue = hiveValue + Integer.valueOf(hbaseValue);
                    put.addColumn(
                            Bytes.toBytes(columnFamily),
                            Bytes.toBytes("cid"),
                            Bytes.toBytes(String.valueOf(resultValue)));
                    putList.add(put);
```



```
            }
        }
        if (putList.size() > 0) {
            table.put(putList);
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (table != null) {
            try {
                table.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (connection != null) {
            try {
                //Closes the HBase connection.
                connection.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
...
```

NOTE

For an MRS cluster with Kerberos authentication enabled, the application needs to perform user authentication on the server. In this sample project, configure authentication information in code. Set **userPrincipal** to the username for authentication and change **userKeytabPath** and **krb5ConfPath** to the actual file paths on the client server.

Step 3 After confirming that the parameters in the project are correct, build the project and package it into a JAR file.

In the Maven window, select **clean** from **Lifecycle** to execute the Maven building process. Select **package** and obtain the JAR package from the **target** directory.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:36 min
[INFO] Finished at: 2023-06-12T20:46:24+08:00
[INFO] ------------------------------------------------------------------------
```

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

----End

Uploading the JAR Package and Preparing Source Data

Step 1 Upload the JAR package to a directory, for example, **/opt/client/sparkdemo**, on the client node.

NOTE

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

Step 2 Upload the keytab file used for authentication to the specified location in the code, for example, **/opt/client**.

Step 3 Log in to the node where the cluster client is installed as user **root**.

cd /opt/client

source bigdata_env

kinit sparkuser

Step 4 Create a Hive table and insert data into the table.

beeline

In the Hive Beeline, run the following commands to create a table and insert data:

create table person (name STRING, account INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ESCAPED BY '\\' STORED AS TEXTFILE;

insert into table person(name,account) values("1","100");

select * from person;

```
+--------------+-----------------+
| person.name  | person.account  |
+--------------+-----------------+
| 1            | 100             |
+--------------+-----------------+
```

Step 5 Create an HBase table and insert data into the table.

Exit the Hive Beeline, run the **spark-beeline** command, and run the following command to create an HBase table:

create table table2 (key string, cid string) using org.apache.spark.sql.hbase.HBaseSource options(hbaseTableName "table2", keyCols "key", colsMapping "cid=cf.cid");

Exit the Spark Beeline, run the **hbase shell** command to go to the HBase Shell, and run the following commands to insert data:

put 'table2', '1', 'cf:cid', '1000'

scan 'table2'

```
ROW                                   COLUMN
+CELL
 1                                    column=cf:cid, timestamp=2023-06-12T21:12:50.711,
value=1000
1 row(s)
```

----End

Running the Application and Viewing the Result

Step 1 On the node where the cluster client is installed, run the following commands to execute the JAR package exported from the sample project:

cd /opt/client

source bigdata_env

cd Spark2x/spark

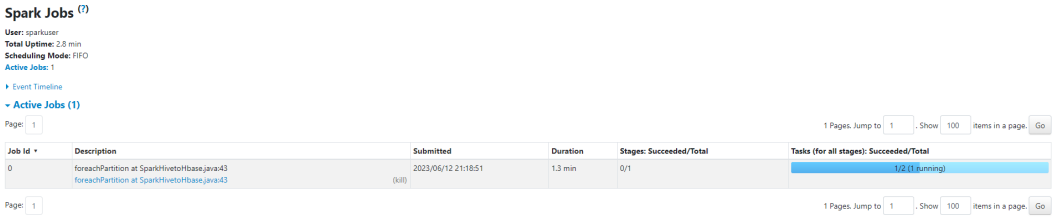
vi conf/spark-defaults.conf

Change the value of **spark.yarn.security.credentials.hbase.enabled** to **true**.

bin/spark-submit --class com.huawei.bigdata.spark.examples.SparkHivetoHbase --master yarn --deploy-mode client /opt/client/sparkdemo/SparkHivetoHbase-1.0.jar

Step 2 After the task is submitted, log in to FusionInsight Manager as user **sparkuser**, choose **Cluster** > **Services** > **Yarn**, and link to the ResourceManager web UI. Then locate the Spark application job information and click **ApplicationMaster** in the last column of the application information to go to the Spark UI and view details.

Figure 12-23 Viewing Spark task details



Step 3 After the task is complete, query the HBase table content in the HBase shell. You can see that the records have been updated.

cd /opt/client

source bigdata_env

hbase shell

scan 'table2'

```
ROW                                   COLUMN
+CELL
 1                                    column=cf:cid, timestamp=2023-06-12T21:22:50.711,
value=1100
1 row(s)
```

----End

13 Practices

After an MRS cluster is deployed, you can try some practices provided by MRS to meet your service requirements.

Table 13-1 Best practices

Practice		Description
Data analytics	**Using Spark2x to Analyze IoV Drivers' Driving Behavior**	This practice describes how to use Spark to analyze driving behavior. You can get familiar with basic functions of MRS by using the Spark2x component to analyze and collect statistics on driving behavior, obtain the analysis result, and collect statistics on the number of violations such as sudden acceleration and deceleration, coasting, speeding, and fatigue driving in a specified period.
	Using Hive to Load HDFS Data and Analyze Book Scores	This practice describes how to use Hive to import and analyze raw data and how to build elastic and affordable offline big data analytics. In this practice, reading 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.
	Using Hive to Load OBS Data and Analyze Enterprise Employee Information	This practice describes how to use Hive to import and analyze raw data from OBS and how to build elastic and affordable big data analytics based on decoupled storage and compute resources. 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.

Practice		Description
	Using Flink Jobs to Process OBS Data	This practice describes how to use the built-in Flink WordCount program of an MRS cluster to analyze the source data stored in the OBS file system and calculate the number of occurrences of specified words in the data source. 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.
Data migration	**Data Migration Solution**	This practice describes how to migrate HDFS, HBase, and Hive data to an MRS cluster in different scenarios. You will try to prepare for data migration, export metadata, copy data, and restore data.
	Migrating Data from Hadoop to MRS	In this practice, CDM is used to migrate data (dozens of terabytes or less) from Hadoop clusters to MRS.
	Migrating Data from HBase to MRS	In this practice, CDM is used to migrate data (dozens of terabytes or less) from HBase clusters to MRS. HBase stores data in HDFS, including HFile and WAL files. The **hbase.rootdir** configuration item specifies the HDFS path. By default, data is stored in the **/hbase** folder on MRS. Some mechanisms and tool commands of HBase can also be used to migrate data. For example, you can migrate data by exporting snapshots, exporting/importing data, and CopyTable.
	Migrating Data from Hive to MRS	In this practice, CDM is used to migrate data (dozens of terabytes or less) from Hive clusters to MRS. Hive data migration consists of two parts: • Hive metadata, which is stored in the databases such as MySQL. By default, the metadata of the MRS Hive cluster is stored in MRS DBService (Huawei GaussDB database). You can also use RDS for MySQL as the external metadata database. • Hive service data, which is stored in HDFS or OBS

Practice		Description
	Migrating Data from MySQL to an MRS Hive Partitioned Table	This practice demonstrates how to use CDM to import MySQL data to the Hive partition table in an MRS cluster. Hive supports SQL to help you perform extraction, transformation, and loading (ETL) operations on large-scale data sets. Queries on large-scale data sets take 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.
	Migrating Data from MRS HDFS to OBS	This practice demonstrates how to migrate file data from MRS HDFS to OBS using CDM.
System Interconnection	**Using DBeaver to Access Phoenix**	This practice describes how to use DBeaver to access Phoenix.
	Using DBeaver to Access HetuEngine	This practice describes how to use DBeaver to access HetuEngine.
	Interconnecting Hive with External Self-Built Relational Databases	This practice describes how to use Hive to connect to open-source MySQL and Postgres databases. After an external metadata database is deployed in a cluster that has 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.
	Interconnecting Hive with CSS	This practice describes how to use Hive to interconnect with CSS Elasticsearch. In this practice, you will 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.

